Ключевые процессы тестирования

Планирование, подготовка, проведение, совершенствование

Рекс Блэк

Critical Testing Processes

Plan, Prepare, Perform, Perfect

Rex Black

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Ключевые процессы тестирования

Планирование, подготовка, проведение, совершенствование

Рекс Блэк

Издательство «Лори»

Critical Testing Processes
Plan, Prepare, Perform, Perfect

Rex Black

Copyright © 2004 by Addison-Wesley
All rights reserved

ISBN 0-201-74868-1

Ключевые процессы тестирования
Планирование, подготовка, проведение, совершенствование

Рекс Блэк

Переводчик *Павлов М.*
Научный редактор *А. Головко*
Корректор *Килимник Т.*
Верстка *А. Деевой*

© Издательство "Лори", 2006

Изд. № : OAI (03)
ЛР № : 07612 30.09.97 г.
ISBN 5-85582-239-7

Подписано в печать 11.07.2006 Формат 70 × 100/16
Бумага газетная Гарнитура Баскервиль Печать офсетная
Печ. л. 36 Тираж 1500 Заказ № 28
Цена договорная

Издательство "Лори"
123100, Москва, Шмитовский пр., д. 13/6, стр. 1 (пом. ТАРП ЦАО)
Телефон для оптовых покупателей: (495)256-09-83
Размещение рекламы: (495)259-01-62
WWW.LORY-PRESS.RU

Отпечатано в типографии ООО "Тиль-2004"
Москва, Каширский пр., д. 23

Моей жене Лорель, дочерям Шарлотте и Эмме
и двум собакам Максу и Космо
с благодарностью за любовь, поддержку и дружбу.

СОДЕРЖАНИЕ

Предисловие 8
Благодарности 10
Вступление 12
Введение 15
Основные темы книги 15
Предполагаемый круг читателей 17
Снимем с процессов покров тайны 18
Виды ключевых процессов тестирования и их взаимосвязь 19
Ключевые процессы тестирования, взятые в контексте 22
Гипотетический проект "Суматра" 23
Терминология 24
Относительно использования данной книги 24
Построение успешного процесса 24
Решение проблем 26
Внедрение усовершенствований 26
Успехов на ниве тестирования! 27

ЧАСТЬ I. ПЛАНИРОВАНИЕ 1

Глава 1. Панорамный обзор: роль тестирования в более широком контексте 3
Процесс тестирования 8
Понимание функционального и организационного контекста 11
Представление проекта "Суматра" и группы тестирования приложения SpeedyWriter 14
Место тестирования в моделях жизненного цикла разработки 19
Организация тестировщиков и тестирования 24
Измерения управления 31
За рамками контекста процесса тестирования 33

Глава 2. Основа успеха: анализ рисков качества 35
Процесс анализа рисков качества 36
Группа тестирования Джамала проводит анализ рисков качества 36
Построение успешного процесса анализа рисков качества 43
Привлечение подходящих участников 45
Выбор подходящей методики 48
Отделение немногих важных аспектов от множества тривиальных 52
Поиск возможных дефектов до начала тестирования 52
Своевременное начало 54

Готовность к изменениям . . . 54
Решение проблем . . . 56
Реальная оценка рисков . . . 56
Многообразие источников рисков качества . . . 58
Кто управляет анализом рисков качества? . . . 59
Исключение магии из количественных оценок рисков . . . 60
Как работать с вариациями значений оценок . . . 61
Как убедить заинтересованных лиц в принятии реальности . . . 63
Внедрение усовершенствований . . . 63

Глава 3. Внутри хрустального шара: оценка предстоящей работы . . . 65
Процесс оценки затрат . . . 67
Джамал предсказывает будущее . . . 70
Использование эмпирических приемов для грубой оценки . . . 81
Объем, продолжительность и зависимости работ . . . 83
Планирование . . . 83
Подбор персонала . . . 86
Разработка тестов . . . 87
Настройка тестовой среды . . . 91
Выполнение тестов . . . 93
Переход на доллары и центы . . . 96

Глава 4. Не сколько стоит, а сколько экономит: бюджет и возврат инвестиций . . . 97
Анализ возврата инвестиций в тестирование . . . 98
Исправленные и предотвращенные ошибки . . . 99
Неисправленные, но известные ошибки . . . 106
Тесты, снижающие риски . . . 107
Ведение проекта . . . 110
Заключительное замечание об оценках возврата вложений . . . 112
Джамал готовится выставить счет и его обоснование . . . 113
Двигаясь вперед, оглянись назад . . . 120

Глава 5. От оценки затрат к базовой версии: получение одобрения реалистичных, работоспособных и правдивых оценок . . . 121
«Продажа» оценки затрат . . . 121
Джамал готовит свой вариант . . . 126
Построение успешного процесса оценки затрат . . . 129
Определение факторов, влияющих на оценку . . . 129
Представление реалистичной картины будущего проекта . . . 132
Назначение ответственных за выполнение . . . 134
Честные прогнозы . . . 135
Быстрая оценка затрат . . . 137
Решение проблем . . . 137
Тестирование как объект инвестиции . . . 137
Возможно, это похоже на чудо, но это только оценка . . . 144
Правило девяти месяцев . . . 145
Осторожно! Незаинтересованные лица! . . . 146
Когда «продают» неудачные оценки . . . 148
Системы с повышенными требованиями к безопасности или к выполняемой целевой задаче . . . 149
Внедрение усовершенствований . . . 150

Глава 6. Изучение и обсуждение содержания проекта: планирование тестирования . . . 153
Процесс планирования тестирования . . . 154
У Джамала есть план . . . 155
За рамками процесса: основные соображения по поводу плана тестирования . . . 170
Выбор стратегий тестирования . . . 171

Особый случай: стратегии управления рисками регрессионной зависимости . . . 176
Определение ресурсов среды для проведения тестирования . . . 181
Управление системами тестов и конфигурациями тестовой среды . . . 183
Риски подпроекта тестирования . . . 186
Свистать всех наверх . . . 187

Глава 7. От предложения к утверждению: поиск поддержки качественных планов тестирования у заинтересованных лиц . . . 189
Джамал выполняет подачу . . . 190
Построение грамотного процесса планирования тестирования . . . 193
Установка четких критериев для фаз . . . 194
Достижение консенсуса, согласование ожиданий, получение одобрения . . . 196
Проведение планирования в разумное время . . . 199
Разумная гибкость и творчество . . . 200
Подготовка надлежащей документации . . . 202
Предоставление возможности для обнаружения ошибок . . . 204
Использование синергетики в планах проекта, разработки, сборки и интеграции . . . 205
Решение проблем . . . 208
Выход за пределы шаблона . . . 208
Планирование для аутсорсинга . . . 208
Если ключевые участники проекта не поддерживают план тестирования . . . 210
Подчинение закону и следование правилам . . . 213
Внедрение усовершенствований . . . 214

ЧАСТЬ II. ПОДГОТОВКА . . . 217

Глава 8. Поиск искусных тестировщиков: как и кого нанимать . . . 219
Процесс создания команды . . . 220
Разновидности процесса создания команды . . . 222
Увеличение группы тестирования проекта "Суматра" . . . 223
Выявление ключевых навыков тестировщика и управление ими . . . 234
О младших тестировщиках . . . 237
Варианты состава группы тестирования: временное назначение, ротация, команда-питомник и тихая заводь . . . 240
Образование, дополнительное обучение, сертификация и превращение тестирования в профессию . . . 243
Отношение к делу . . . 247
По ту сторону очевидных вопросов в ходе интервью . . . 249
От увеличения группы к росту квалификации . . . 252

Глава 9. Подготовка превосходной группы тестирования: навыки, отношение к делу и пути развития карьеры . . . 253
Джамал и Лин Цу обсуждают вопросы повышения квалификации . . . 253
Оценка навыков как инструмент управления карьерным ростом группы тестирования . . . 256
Построение успешного процесса создания группы тестирования . . . 258
Применение философии "выигрывают обе стороны" . . . 258
Четкое определение должностных обязанностей . . . 260
Предоставление сотруднику долгосрочного пути развития карьеры . . . 265
Выплата сотрудникам справедливой заработной платы, соответствующей их квалификации . . . 266

Поддержка сотрудников с достаточным образованием, опытом, умениями и навыками . . . 268
Подходящая специализация . . . 269
Решение проблем . . . 271
Что должны уметь тестировщики и что вы можете получить? . . . 271
Белые слоники . . . 272
Использование контрактников и консультантов . . . 273
Внедрение усовершенствований . . . 276

Глава 10. Архимедова ванна: проектирование и разработка системы тестов . . . 278
Процесс проектирования и разработки системы тестов . . . 280
Эмма создает напряжение в работе . . . 280
Три решающих соображения . . . 291
Тестовые оракулы . . . 291
Подходящая документация . . . 293
Комбинаторные взрывы . . . 299
От технического взгляда к точке зрения менеджмента . . . 302

Глава 11. Заполнение ванны: тестовое покрытие и качество системы тестов . . . 303
Краткий обзор методов анализа тестового покрытия . . . 303
Джамал оценивает покрытие . . . 307
Построение успешного процесса проектирования и разработки системы тестов . . . 311
Предоставление работоспособных инструментов для оценки качества . . . 311
Эффективная реализация, применение, повторное использование и сопровождение системы тестов . . . 313
Выбор подходящих методов . . . 317
Предотвращение дефектов . . . 320
Решение проблем . . . 321
Сколько условий в тестовом сценарии? . . . 321
Перетестирование сторонних компонентов . . . 322
Неясно сформулированные требования, профили пользователей и описания конфигураций среды . . . 324
Уравновешивание покрытия, сроков и бюджета . . . 325
Внедрение усовершенствований . . . 326

ЧАСТЬ III. ПРОВЕДЕНИЕ . . . 329

Глава 12. Передача материалов: управление тестовыми версиями . . . 331
Процесс сборки тестовых версий . . . 332
Большая сборка поступает в тестовую лабораторию . . . 333
Построение успешного процесса подготовки тестовых версий . . . 339
Подготовка инсталлируемых, усовершенствованных и стабильных тестовых версий . . . 339
Подготовка предсказуемых, своевременных и регулярных тестовых версий 342
Использование четко определенных процессов инсталляции и деинсталляции, похожих на те, что применяет пользователь . . . 345
Присвоение имен тестовым версиям . . . 347
Подготовка сборки из тщательно отобранных, скоординированных и документированных компонентов . . . 348
Решение проблем . . . 352
Кто отвечает за процессы подготовки тестовых версий? . . . 352
Сложность системы, предназначенной для тестирования, и другие вопросы . . . 353
Неполные и нереалистичные тестовые версии для системного тестирования и приемо-сдаточных испытаний . . . 355
Внедрение усовершенствований . . . 357

Глава 13. Оценка качества: выполнение тестовых сценариев 361
Процесс выполнения тестов 361
Таблицы отслеживания тестов 363
Атака на большую сборку 366
Построение успешного процесса выполнения тестов 373
Выявление опасных мест 373
Генерация, сбор и распространение значимой информации 375
Правильная интерпретация результатов тестирования 377
Демонстрация надлежащего отношения 380
Эффективная работа 382
Решение проблем 384
Работа с дополнительными сменами и субподрядчиками, проводящими тестирование 384
Учет праздников, отпусков и культурных традиций 387
Сбор данных о проекте тестирования на месте 387
Внедрение усовершенствований 388

ЧАСТЬ IV. СОВЕРШЕНСТВОВАНИЕ 391

Глава 14. Когда качество не оправдывает ожиданий: подготовка отчетов об ошибках 393
Процесс подготовки отчетов об ошибках 394
Большая ошибка в большой сборке 398
За рамками описания дефекта 405
Построение успешного процесса подготовки отчетов об ошибках 408
Эффективная передача полезной информации 408
Включение одного симптома в каждый отчет и подготовка одного отчета на каждый симптом 412
Установка четкой границы между тестированием и отладкой 413
Решение проблем 415
Разделение ошибок, свойств и ответов "ну, и что? 415
Обработка случайно исправленных ошибок и невоспроизводимых симптомов 417
Удаление шума из сигнала 418
Создание доверительных отношений с программистами 419
Выбор инструмента управления дефектами 421
Внедрение усовершенствований 422

Глава 15. Четвертый элемент: подготовка отчетов о результатах тестирования . . . 427
Процесс подготовки отчетов о результатах тестирования 428
Джамал определяет содержимое инструментальной панели и отчеты о результатах тестирования большой сборки 430
Построение успешного процесса подготовки отчетов о результатах тестирования 447
Своевременная передача полезной, заслуживающей доверия информации 447
Связь состояния тестирования с ходом проекта и его влияние на будущее проекта 450
Демонстрация тенденций развития качества и возможных результатов . . 453
Применение подходящих механизмов и документов 454
Передача информации в удобной для каждого потребителя форме 456
Решение проблем 457
Подготовка презентации 457
Носитель плохих новостей 459
Постоянно меняющаяся сложная история 463
Сочетание переживания за качество с бесстрастным представлением результатов 464
Внедрение усовершенствований 464

Глава 16. Увеличение возможностей для обучения: управление изменениями и их влиянием на тестирование 467
Процесс управления изменениями 468
Джамал готовит и согласовывает план 470
Взаимосвязанный процесс 479
Построение успешного процесса управления изменениями 480
Выбор нужных изменений в нужном порядке 481
Уравновешивание мнений по поводу свойств, сроков, бюджета и качества 482
Решение проблем 483
Сложная природа воздействия изменений 483
Волнообразный эффект изменений 485
Действительное или кажущееся препятствие? 486
Внедрение усовершенствований 487

Глава 17. Снова к общей картине: совершенствование процесса тестирования 489
Определение грамотного процесса тестирования 489
Предоставление полезных и недорогих услуг 489
Проникновение в проект 491
Использование зрелого процесса 494
Применение подходящего уровня формализации и стандартизации 495
Использование подходящих стратегий 501
Успешное внедрение инструментов 504
Получайте удовольствие от вдохновения и от возможности непрерывно учиться 505
Решение проблем 507
Как измерить то, чему мы не в силах дать определение? 507
Незрелые процессы разработки и сопровождения 508
Нереалистичные ожидания 509
Взаимоотношения 511
"Привычка тестировать" 512
Постепенное совершенствование процесса 514
Точная регулировка больших процессов 514
Выявление и устранение крупных препятствий для процессов 516
Внедрение усовершенствований 518
Джамал Браун оглядывается назад и смотрит вперед 520
Заключение 523

Глоссарий 525

Библиография 538

ПРЕДИСЛОВИЕ

Рекс Блэк написал новую книгу! Читатели его первой книги "*Managing the Testing Process*", несомненно, возлагали большие надежды на будущие произведения Рекса. И они не будут разочарованы его новой работой "*Ключевые процессы тестирования*". Он обратился к сложной теме и описал ее в доступной форме, чутко реагирующей на изменения контекста. Абстрактные на первый взгляд процессы тестирования оживают, и читатели могут разобраться в них и применить в собственных проектах.

Некоторое время назад мы вместе с Рексом работали консультантами и читали цикл лекций. Приятно проводя время за ужином, мы нередко беседовали о том, насколько близки наши взгляды по поводу того, как следует проводить тестирование, что действительно имеет значение, а что нет. Когда пишешь книгу, легко увязнуть, пытаясь выделить все идеи, которые представляются полезными для читателя. Порой при стремлении рассказать "все, что мы знаем" действительно важные мысли теряются в общей массе. Рекс успешно избежал этого, в чем и состоит прелесть данной книги. Рекс рассмотрел весь имеющийся в его распоряжении значительный и разнообразный опыт и описал 12 ключевых процессов.

Эти процессы показаны достаточно подробно для того, чтобы пользователь мог понять их, а при необходимости видоизменить и внедрить в своей компании. Рекс разъясняет процессы в эффективной, но необычной манере. Сначала он демонстрирует их применение на примере действий вымышленной (но правдоподобной) проектной группы. Затем приводятся общие положения каждого из этих процессов и даются советы, как их применять при смешении различных культур и условий. И наконец, каждый процесс "оживает" через личный опыт Рекса. Для того чтобы помочь читателю получить четкое представление о принципах тести-

рования, часто используются средства визуализации: графики, таблицы и примеры.

Читателям вышедшей недавно книги "*Systematic Software Testing*", написанной мной в соавторстве со Стефаном Яскилем, известно, что несмотря на значимость инструментов, измерений и технических процессов, в конечном счете, по нашему мнению, качество программного продукта зависит от людей. В "*Ключевых процессах тестирования*" особенно хорошо освещено значение человеческого фактора, часто игнорируемого при создании программного продукта, включая взаимодействие тестировщиков друг с другом и с группами и людьми, внешними для группы тестирования. Рекс не рассматривает тестирование как обособленный вид деятельности, он показывает, как группа тестирования и отдельные тестировщики работают в рамках процесса создания программного продукта в целом.

Я работаю в сфере тестирования больше 20 лет и считаю это занятие благодарным и увлекательным. Но чтение книг на эту тему не всегда бывает столь же захватывающим. Однако неформальный, легкий для восприятия стиль изложения Рекса превращает изучение ключевых процессов тестирования в несложное, полезное и даже веселое занятие! "Ключевые процессы тестирования" – одна из тех книг, которую тестировщики читают от корки до корки, а затем на нее часто ссылаются. Этой книге суждено стать классикой в области тестирования программного обеспечения.

Рик Крэйг
Консультант компании
Software Quality Engineering
Автор книги "Систематическое
тестирование программного обеспечения"
Тампа, штат Флорида

БЛАГОДАРНОСТИ

Эта книга зародилась, когда Мэгди Ханна (Magdy Hanna) предложила написать для журнала "*Journal of Software Testing Professionals*" серию статей о процессах тестирования. Помимо этого журнала, части книги выходили в виде статей в журналах "*Professional Tester*", "*Software Dioxide*" и "*Software Testing and Quality Engineering*"; в электронных журналах Stickyminds.com и Test and Measurement.com; и в виде главы книги Эрика Ван Веенендааля (Erik van Veenendaal) "*The Testing Practitioner*". За рецензирование этих ранних фрагментов книги я благодарю Роберта Кутрэ (Robert Coutre), Эстер Дерби (Esther Derby), Брайана Хамблинга (Brian Hambling), Мэгди Ханну (Magdy Hanna), Линду Хайес (Linda Hayes), Брайана Лоуренса (Brian Lawrence), Брайана Марика (Brian Marick), Дениса Мередита (Denis Meredith), Джеми Митчелла (Jamie Mitchell), Каролайн Квентин (Caroline Quentin), Лесли Сигала (Leslie Segal), Ребекку Треджер (Rebecca Traeger), Эрика Ван Веенендааля (Erik van Veenendal) и Элина Уомбика (Alyn Wambeke). Я также выражаю благодарность читателям этих отрывков и моей предыдущей книги "*Managing the Testing Project*", которые не пожалели времени, чтобы прислать мне комментарии и вопросы.

Кроме того, в написание данной книги внесли свой вклад замечательные люди из разных стран. Кто-то редактировал главы, по мере того как я их писал. Кто-то подавал идеи, приводил цитаты и примеры из жизни. Кто-то присылал заметки на полях. Хочу поблагодарить Сью Бартлет (Sue Bartlett), Ренделла Беккера (Randall Becker), Бориса Бейзера (Boris Beizer), Росса Колларда (Ross Collard), Ли Копленда (Lee Copeland), Рика Крейга (Rick Craig), Гэри Доусона (Gary Dowson), Харви Дойча (Harvey Deutsch), Тима Дайеса (Tim Dyes), Криса де Нардиса (Chris DeNardis), Роджера Драбика (Rodger Drabick), Дэнни Фота (Danny Faught), Ларри Фэллоуза (Larry Fellows), Пэгги Футс (Peggy Fouts), Нормана Хайнза

(Norman Hines), Кэти Айберли (Kathy Iberle), Якоба Йенсена (Jakob Jensen), Сема Канера (Cem Kaner), Вайпула Кохера (Vipul Kocher), Тима Коомена, Филиппа Крушена (Philippe Kruchten), Маркуса Манляйтнера (Markus Mannleitner), Дарси Мартина (Darcy Martin), Жозе Мата (Jose Mata), Деб Маккэндлз (Deb McCandless), Дэйла Пэрри (Dale Perry), Эрика Петерсона (Erik Peterson), Джоанну Ротман, Барбару Руф (Barbara Ruf), Роберта Сабурина, Грега Скала (Greg Scala), Ребекку Совада (Rebecca Sowada), Стива Сплейна (Steve Splain), Сербана Теодореску (Serban Teodorescu), Анни Цу (Anne Tsou), Эрика Ван Веенендааля и Эда Уеллера (Ed Weller).

Наряду с работой над проектами и консультированием я занимался преподаванием ряда обучающих курсов. Многие рассматриваемые в книге вопросы были темами этих курсов. Я призывал слушателей активно участвовать в занятиях, что они обычно и делали, делясь собственным опытом и описывая забавные случаи. Всех, посещавших мои курсы, я также благодарю за вклад в эту книгу.

До написания первой книги я даже не представлял, сколько работы в издательском бизнесе ведется за кулисами. Поскольку это моя третья книга, я значительно лучше понимаю положение дел, и тем не менее уверен, что с некоторыми людьми, помогавшими донести до вас мои размышления, я незнаком. Поэтому благодарю всех работников компании Addison-Wesley, особенно Кима Эрни Малкачи (Kim Arney Mulcahy), Марси Барнс-Хэнри (Marcy Barnes-Henrie), Бернарда Гафни (Bernard Gaffney), Питера Гордона (Peter Gordon) и Дэбби Лафферти (Debbie Lafferty).

Наконец, последнее и самое главное. Я бесконечно благодарен своей семье. Процесс написания книги требует времени, не обходится без стрессов, а порой доводит до отчаяния. Мои родители, моя семья и родственники жены поддерживали и подбадривали меня. Дочки, Эмма и Шарлота, вместе с нашими собаками, Максом и Космо, часто наблюдали за тем, как я писал эту книгу дома, и отпускали веселые комментарии. Благодаря им, между очередными порциями ударов по клавиатуре я веселился, что было так важно! Последней (по порядку упоминания, конечно, а не по значимости) назову мою жену Лорель, которая не только мужественно переносит все тяготы положения жены "странствующего" консультанта, но и уже трижды пережила процесс создания книги. Лорель, ты лучше всех!

Я благодарен всем, кто своим участием помог мне сделать книгу лучше. Если бы ни помощь названных мной людей, в книге было бы хороших идей меньше, а плохих больше. Конечно, автор этой книги — я, и ответственность за нее лежит исключительно на мне. Ошибки есть, и это мои ошибки. Направляйте, пожалуйста, комментарии, вопросы и критические замечания по поводу этой книги на адрес Rex_Black@RexBlackConsulting.com. Спасибо за чтение и удачи в ключевых процессах тестирования.

ВСТУПЛЕНИЕ

Более 20 лет я занимаюсь тестированием программного и аппаратного обеспечения. Став тестировщиком, первые несколько лет я провел в борьбе за то, чтобы просто "удержаться на плаву". В конце концов, я освоил некоторые основные инструменты и методы.

По мере более глубокого погружения в тестирование я начал замечать некоторые общие проблемы. Ряд из них был связан с событиями (хорошими и плохими), повторяющимися вновь и вновь в проектах по созданию программ, аппаратуры и систем. При этом я обнаружил, что некоторым командам удавалось навести порядок в своих проектах. Такие команды лучше справлялись с типичными происшествиями, чем команды, постоянно испытывающие кризисы, находящиеся в вечной борьбе, погруженные в хаос. Успешные команды использовали хорошие процессы.

Некоторые успешные проектные группы внедряли процессы, зафиксированные в письменной форме, сотрудники других удачливых команд держали "институциональные" знания в мудрых (порой поседевших раньше времени) головах. Не имея ничего против распространения корпоративных культур, считаю, что трудно пользоваться уже освоенными процессами, если они не отражены в письменном виде, формально или в виде перечня контрольных вопросов. В этой книге избран неформальный путь. Я описал 12 отдельных процессов тестирования, используя перечни контрольных вопросов. Каждый процесс является ключевым для успешной работы группы тестирования.

Я описываю эти процессы в хронологическом порядке. Прежде всего мы планируем работы по тестированию. Затем ведем подготовку к тестированию. Далее проводим тесты. И наконец совершенствуем систему, предназначенную для тестирования, и сами процедуры тестирования.

Во многих других книгах подробно рассматриваются вопросы подготовки и выполнения тестов. Потому, вместо того чтобы переформулировать уже известное, я концентрирую внимание на возможностях усовершенствования процессов. Одиннадцать из семнадцати глав посвящены вопросам планирования и совершенствования. Безусловно, именно с этими направлениями связано большинство проблем, с которыми мы, тестировщики, сталкиваемся. Это особенно актуально в условиях сложных и критических проектов.

Куда заведет вас эта книга? В начале колонизации Америки, в 1540-х, Франциско Васкес де Коронадо искал в пустынях на территории современной Аризоны и Нью-Мексико страну Семи городов под названием Сибола, включая Эльдорадо — город, улицы которого считались мощеными золотом. Жуан Понс де Леон вел поиски эликсира вечной молодости. В 1911 г. Фредерик Уинслоу Тейлор, один из первых специалистов по менеджменту, написал книгу под названием "*Принципы научного менеджмента*". Тейлор придерживался идеи о существовании единственного лучшего способа — идеального процесса — для каждого вида производственной деятельности. Ни один из этих троих не нашел желаемого: улиц из золота, эликсира молодости, идеальных процессов.

В этой книге нет поиска идеала. Не существует золотых улиц, которые позволят нам обогатиться без усилий. Мы не в силах преодолеть ограничения, наложенные человеческой природой. Я не предлагаю безупречных процессов. Как написал Фредерик Брукс в книге "*Мифический человеко-месяц*", у нас нет серебряных пуль, чтобы уничтожить монстров наших системных проектов, включая тех, что водятся в обеспечении качества и в тестировании. Я узнал много способов, с помощью которых тестировщик может передать важную информацию и предоставить услуги рабочей группе проекта, и каждый из этих способов имеет свои сильные и слабые стороны.

Процессы, описанные в книге, могут отличаться от тех, с которыми вы сейчас имеете дело. Может быть, основываясь на своем удачном опыте, вы придете к выводу, что уже хорошо делаете свое дело. Тем не менее в некоторых случаях вам, возможно, захочется внести усовершенствования. Я описываю конкретные способы, позволяющие это сделать, а две темы, касающиеся изменения процесса, рассматриваются на протяжении всей книги.

Во-первых, полезно лишь изменение того, что уже идет наперекосяк. Изменение процессов ради самого изменения или совершенствование и без того хороших процессов зачастую не помогает группе тестирования или компании. На самом деле это может привести к серьезным отклонениям от того, что действительно важно.

Во-вторых, изменения нередко переносятся легче, если делать их поэтапно, там, где возможно. Изменения должны быть как можно менее болезненными. Все процессы, описанные в этой книге, были созданы благодаря последовательным изменениям по мере того, как я понимал, что более удобный способ выполнения конкретной работы позволит существенно увеличить вклад моей группы в общее дело, и точно настраивал процессы для достижения этой цели.

Процессы, описанные в книге, это не "журавль в небе"; они, скорее, выросли из моего опыта работы в качестве тестировщика, ведущего тестировщика и тест-менеджера. Ваш опыт и проблемы, с которыми вы сталкиваетесь, отличны от моих, потому будьте готовы адаптировать мой процесс или полностью придумать заново свой собственный и не пытайтесь надеть на корову седло. Применение хороших процессов позволит вам освободиться от некоторых механических видов работы и сосредоточиться на интересных, приятных и творческих. Если привычные процессы перестают решать ключевые проблемы, если меняется ситуация и необходимы преобразования, значит нужно пересмотреть способы, с помощью которых вы делаете свое дело. Обсуждаемые здесь процессы предстают в форме списка контрольных вопросов (то, что я хочу не забыть сделать), а не бюрократических правил (то, что я должен проделать, поскольку кто-то сказал мне об этом).

Надеюсь, что, прочитав эту книгу, вы начнете задавать себе вопросы: "Как выполнять работы по тестированию наилучшим образом каждый день и в каждом проекте? Используем ли мы сведения о том, как мы делаем свое дело? Даже с учетом разнообразного опыта, разработана ли у нас совокупность мероприятий, способная при перенесении ее на критические процессы тестирования предопределить успешную работу?" Данная книга является кратким руководством по испытанным процессам тестирования, которое призвано помочь запустить главнейший из ключевых процессов тестирования — мыслительный процесс.

ВВЕДЕНИЕ

В первом предложении "*Анны Карениной*" Лев Толстой пишет: "Все счастливые семьи похожи друг на друга, каждая несчастливая семья несчастлива по-своему". Бесспорно, Толстой создал великое произведение, написанное сильным языком и не оставляющее равнодушным, однако здесь он был не прав. Частности, касающиеся любой семьи, счастливой или несчастной, могут быть совершенно разными, но есть и некоторые общие причины несчастья. "*Анна Каренина*" – это история нелюбимого мужа и жены-обманщицы. Это две распространенные и взаимосвязанные беды, с которыми сталкиваются семьи во всем мире. Для семейного счастья тоже существуют общие предпосылки, например здоровые дети, финансовое благополучие, любовь, взаимопонимание и общность интересов.

Подобно семьям, компании представляют собой социальные группы; они также бывают счастливыми или несчастливыми, подчиняясь общим закономерностям. Сотрудники успешных компаний среди прочего довольны своей работой, стабильностью и возможностями карьерного роста. Работники неуспешной компании становятся несчастными, прилагая героические усилия и не получая достаточной отдачи, особенно когда результатом становится неудачный продукт. Одним из секретов успешной работы компании, занятой в области системных разработок и сопровождения продуктов, является грамотное применение хороших процессов, которые преобразуются с учетом конкретных условий успешными и компетентными специалистами и менеджерами. В книге "*Ключевые процессы тестирования*" я буду вашим проводником по 12-ти таким процессам для группы тестирования.

Основные темы книги

Что такое *ключевой процесс тестирования*? **Процесс – это совокупность действий, наблюдений и решений, используемых для достижения желаемого результата.** Некоторые работы можно проводить одновременно, другие нужно осуществлять последовательно. Определенные виды деятельности могут иметь единую цель, но применяемые в них технологии и используемые навыки необязательно будут общими. Например, один из рассматриваемых ниже процессов – процесс управления сборкой тестовой версии программы – включает в себя выбор компонентов сборки, внесение в нее изменений, создание сборки, подготовку установочного диска данной версии, установку новой тестовой версии и проведение приемочного тестирования.

Процесс является ключевым, если он соответствует одному из следующих критериев.

- Группа тестирования, выполняя свою работу, вновь и вновь прибегает к данному процессу. Ясно определенный подход к повторяющимся процессам ведет к эффективному и последовательному выполнению повседневных обязанностей в сфере тестирования. Процесс составления отчета об ошибках, например, проводится снова и снова после выполнения тестов.
- Процесс влияет на способность тестировщиков работать сообща, особенно если нарушение процесса может разрушить сплоченную, дружную команду. К примеру, если группа тестирования выполняет набор тестов, процесс должен поддерживать распределение конкретных заданий, предотвращая дублирование работ.
- В процессе участвуют равные по должности или равные и вышестоящие сотрудники. Группа тестирования, должным образом работающая с процессами, находящимися на виду, приобретает репутацию надежной и компетентной. Например, при составлении отчетов о состоянии тестирования тестировщики должны предоставить руководству и сотрудникам своего уровня сведения о проделанной работе таким образом, чтобы наиболее эффективно проинформировать их о проведенном тестировании, выявленных проблемах и дать понять, что тестирование ценно как таковое.
- Неудачное применение процесса способно повлечь за собой серьезные, возможно, длительные негативные последствия. Рассмотрим процесс отбора параметров конкретного продукта, подлежащих тестированию. Пропустив в результате неверно нацеленного тестирования основной вид критических для заказчика дефектов, получим, как минимум, неразбериху внутри компании и увеличение затрат, а для некоторых видов систем возможно появление даже физической опасности для заказчика.

Короче говоря, ключевые процессы непосредственно и существенно влияют на способность группы тестирования предоставлять значимые с организационной, операционной и технологической точек зрения информацию и услуги.

Вообразите, что вы готовите торт. У вас есть ингредиенты (входные данные процесса), планируемая вечеринка (проект), голодные гости (проектная группа и другие заинтересованные стороны), кухня (контекст тестирования), рецепт (процесс тестирования). Торт, который вы хотите испечь, — это ожидаемый результат. Пропустите пару ингредиентов, решите, что одно из действий не существенно, легкомысленно отнеситесь к последовательности выполнения задачи, не придавайте значения тому, какие именно торты нравятся гостям. И в итоге на вечеринке появится совсем не то кондитерское изделие.

Не все действия в рамках проекта по созданию программного продукта представляют собой процесс. Рассмотрим устранение ошибок — мероприятие по ликвидации ошибки, описанной в отчете о найденных дефектах. Программист воспроизводит ситуацию, при которой происходит отказ, а далее для устранения дефекта не существует определенного процесса. Имеются типичные инструменты, такие как отладчики и синтаксические анализаторы. Есть эвристики, применяемые программистами на основе собственного опыта в зависимости от симптомов отказа. Имеется также информация технического плана о типах ошибок, связанная с выбранным языком программирования, целевым окружением, компилятором и т.д. Но инструменты, эвристики и навыки не являются процессами. Устранение ошибок, скорее, напоминает поход за покупками, нежели приготовление блюда: вам надо лишь подвести к кассе тележку, в которой удобно расположены все товары из заранее составленного списка, оплатить покупки и отправиться домой.

Я намеренно выбрал эти два примера. Считается, что эти процессы скучные. Что они требуются только людям, не обладающим необходимыми навыками. Что в них нет творческого начала. Я с этим не согласен. Что может быть более созидательным, чем кулинария или выбор покупок? С чем справиться труднее? Что приносит больше удовлетворения? Точно так же я нахожу процесс составления отчетов об ошибках интересным, требующим полной отдачи творческим делом, занимаясь которым я вспоминаю, почему принял решение стать тестировщиком. Напротив, когда я пишу программу, устранение ошибок всегда заставляет вновь и вновь почувствовать утомление и неудовлетворенность от кодирования. Именно это и подтолкнуло меня оставить программирование и стать тестировщиком.

Предполагаемый круг читателей

Эта книга для всех тестировщиков, которые стремятся улучшить свою работу. Это книга для ведущих тестировщиков, тест-менеджеров, менеджеров по созданию программного продукта, отвечающих за тестирование, и для всех, в чьи обязанности временно или постоянно входит обеспечение того, чтобы такой элемент проекта по созданию или сопровождению системы, как тестирование, велся успешно. Она предназначена для людей, ответственных за повышение квалификации тестировщиков или улучшение работы группы тестирования в целом. Для всех, кто связан с тестированием, для организаций, уже внедривших ряд основных процессов тестирования (возможно, на базе идей, предложенных в моей преды-

дущей книге "*Managing the Testing Process*"), а теперь ищущих пути их совершенствования. Если эта книга найдет отклик на уровне высшего звена менеджмента или дирекции, то она способна вывести вашу группу тестирования в авангард сообщества тестировщиков и принесет значительную отдачу на вложенные в тестирование средства.

В отличие от большинства других книг по тестированию программного обеспечения, которые рассматривают прежде всего технические аспекты выполнения конкретных задач, эта книга, кроме всего прочего, поможет вам понять более широкий контекст, в котором и для которого осуществляется тестирование. Эта книга не является продолжением моей первой книги "*Управление процессом тестирования*". Там я рассматривал процесс в целом, его ключевые задачи и процедуры с технической точки зрения. "*Managing the Testing Process*" отличается от прочих книг, посвященных этой теме, поскольку она писалась с учетом потребностей тест-менеджера, но, как и они, в первую очередь направлена на решение тактических задач. Подобные, посвященные тактике, книги весьма полезны при выполнении конкретных задач, но нуждаются в рассмотрении более широкой перспективы. Именно это вы и получаете при чтении "*Ключевых процессов тестирования*". В данной книге рассматривается процесс тестирования в целом и каждый входящий в него ключевой процесс. В центре внимания находятся процессы (конкретные способы, с помощью которых тестировщик предоставляет полезную информацию и услуги в рамках проекта), связывающие воедино конкретные задачи и действия, которые мы выполняем повседневно. Независимо от позиции, которую вы занимаете в сфере тестирования: от инженера-тестировщика до руководителя сотнями тестировщиков, — "Ключевые процессы тестирования" помогут вам получить новое представление о внутренних аспектах вашей деятельности, о ее значении и о том, как работать лучше.

Снимем с процессов покров тайны

Я прочел несколько книг, посвященных совершенствованию процесса создания программного продукта, которые, честно говоря, меня озадачили. Я читал и читал, но по завершении чтения не имел ни малейшего представления, как применить идеи, с которыми только что познакомился. Эта книга, в отличие от других книг, посвященных процессам, не запутает вас. Рассказывая о том или ином процессе, я даю пример гипотетической учебной ситуации, которая рассматривается на протяжении всей книги. Затем я даю некоторые советы по поводу поведения и достижений групп тестирования, использующих хорошие процессы, по поводу того, как справляться с трудностями в процессе и как внедрить усовершенствования. Эти общие идеи я "приправляю" конкретными примерами из собственного опыта. Так что процессы обретают плоть и кровь.

Чтобы не усложнять объяснения, я представляю процессы в виде списков контрольных вопросов. Это полезный и легкий способ организации тестирования, ориентированного на процесс. Кроме того, вопросы позволят вам собирать данные и совершенствовать процессы, поскольку вы начнете контролировать происходящее. Чтобы связать обсуждение кон-

кретных и общих вопросов с частными элементами каждого процесса, о котором идет речь, я буду использовать таблицы, аналогичные таблице для процесса 1.

Процесс 1. Этап 1 процесса тестирования

№ этапа	Этап	Выполнено?
1.	Планирование: понимание места тестирования.	☐
1.A	Определить функциональный (система, проект и процесс) и организационный контексты, в рамках которых выполняется тестирование.	☐
1.B	Определить и расставить приоритеты рисков качества системы, достичь консенсуса заинтересованных сторон проекта относительно возможностей тестирования по смягчению этих рисков.	☐
1.C	Оценить временные, ресурсные и финансовые затраты, необходимые для выполнения тестирования и согласованные с пунктом 1.B, получить поддержку оценки у руководства.	☐
1.D	Запланировать задачи, зависимости и состав участников, которые необходимы для смягчения рисков качества системы, и получить поддержку плана у заинтересованных сторон проекта.	☐

Процессы имеют сквозную нумерацию по всей книге. Хотя вам, несомненно, придется перекраивать эти списки контрольных вопросов для нужд тестирования в своих проектах, вы можете получить их вместе с другой вспомогательной документацией бесплатно на сайте www.rexblackconsulting.com. Загрузите их и переделайте, как вам угодно, в соответствии с лицензией, которая загрузится вместе с этими документами[1].

Виды ключевых процессов тестирования и их взаимосвязь

Некоторые ключевые процессы тестирования являются внутренними для группы тестирования, проводятся исключительно силами тестировщиков и влияют только на них. Другие процессы — совместные, в них включены другие группы компании. Я использую слово "совместные", а не "внешние", поскольку эффективно взаимодействующие группы предварительно принимают работу друг друга. Если группы перебрасывают неполные и недоброкачественные объекты через организационные "стены", это является отличительным признаком слабой командной работы. Совместный процесс может включать в себя прием или предоставление

[1] Не волнуйтесь! Для чтения этой книги вам вовсе не обязательно быть гуру в области SEI CMM, ISO 9000, RUP и других процессов. Если вы в состоянии справиться с инструкцией по сборке трехколесного велосипеда или приготовить цыпленка в тандуре, то для вас не составит труда понять идеи книги, опирающиеся на списки контрольных вопросов и оживленные описанием практических ситуаций, независимо от того, знакомы вам или нет подобные подходы к работе.

передаваемых объектов или и то и другое. При совместном процессе может появиться совокупность передаваемых объектов, связывающих множество команд.

Любой ключевой процесс зависит от контекста, в рамках которого он проходит. То есть процесс должен соответствовать участвующим в нем людям, системе, проекту и компании. В ситуации, зависящей от контекста, едва уловимые факторы могут повлечь за собой последствия, желательные в одном случае и нежелательные в другом. Например, бросать еду через плечо, как правило, не стоит, однако это может оказаться вполне приемлемым, если в качестве еды выступает просыпанная соль, а вокруг вас — суеверные люди. Рассмотрим пример окружения процесса тестирования. Предоставление отчетов о результатах тестирования группе управления проектом требует хороших навыков в общении, ведь нужно чувствовать, что, когда и кому сказать.

Тестирование в ракурсе проекта связано с множеством различных видов деятельности и групп. На рис. I.1 упрощенно показаны ряд процессов (стрелки) и групп (круги), участвующих в процессе разработки программного обеспечения в ходе выполнения тестов.

Выполнение тестов является здесь внутренним процессом, поскольку отслеживает результаты цикла тестирования. Предоставление отчетов об ошибках является совместным процессом, в ходе которого группа тестирования передает группе разработчиков информацию, получаемую в процессе проведения тестов. Аналогично, отправка перечня изменений для конкретной тестовой версии представляет собой совместный процесс передачи группой разработчиков группе тестирования информации, требующейся в процессе проведения тестов. В виду того, что процесс подготовки отчетов о состоянии тестирования зависит от контекста, тест-менеджер сообщает группе управления проектом о продвижении работ по тестированию и о том, как идет оценка качества программного продукта, прибегая к графикам и измерениям, понятным данной группе и подходящим для нужд процесса разработки. И наконец, обратите внимание на многоступенчатый совместный процесс передачи версии программного продукта группе тестирования, который был разбит мной на совместные подпроцессы, показанные в круглых скобках.

Многие передаваемые объекты и зависимости должны по времени и по форме аккуратнейшим образом сходиться в одной точке — на группе тестирования. Только в этом случае она сможет своевременно предоставлять точную, полную и надежную информацию и соответствующие услуги по тестированию. Подобные совместные процессы относятся к наиболее критическим. Точно так же влияние контекста часто скрыто и трудно различимо. Прежде чем принять решение по поводу надлежащего процесса, тестировщики должны внимательно изучить происходящее, поскольку реальное положение дел может существенно отличаться от того, что говорят или думают об этом другие. Контекстно-зависимый и совместный характер многих процессов усложняет их проведение, поскольку в группах возникает необходимость адаптировать и перестраивать процессы для того, чтобы удовлетворить взаимные потребности или чтобы подстроиться под изменившуюся ситуацию, а это требует использования дипломатии, проведения переговоров и координации усилий.

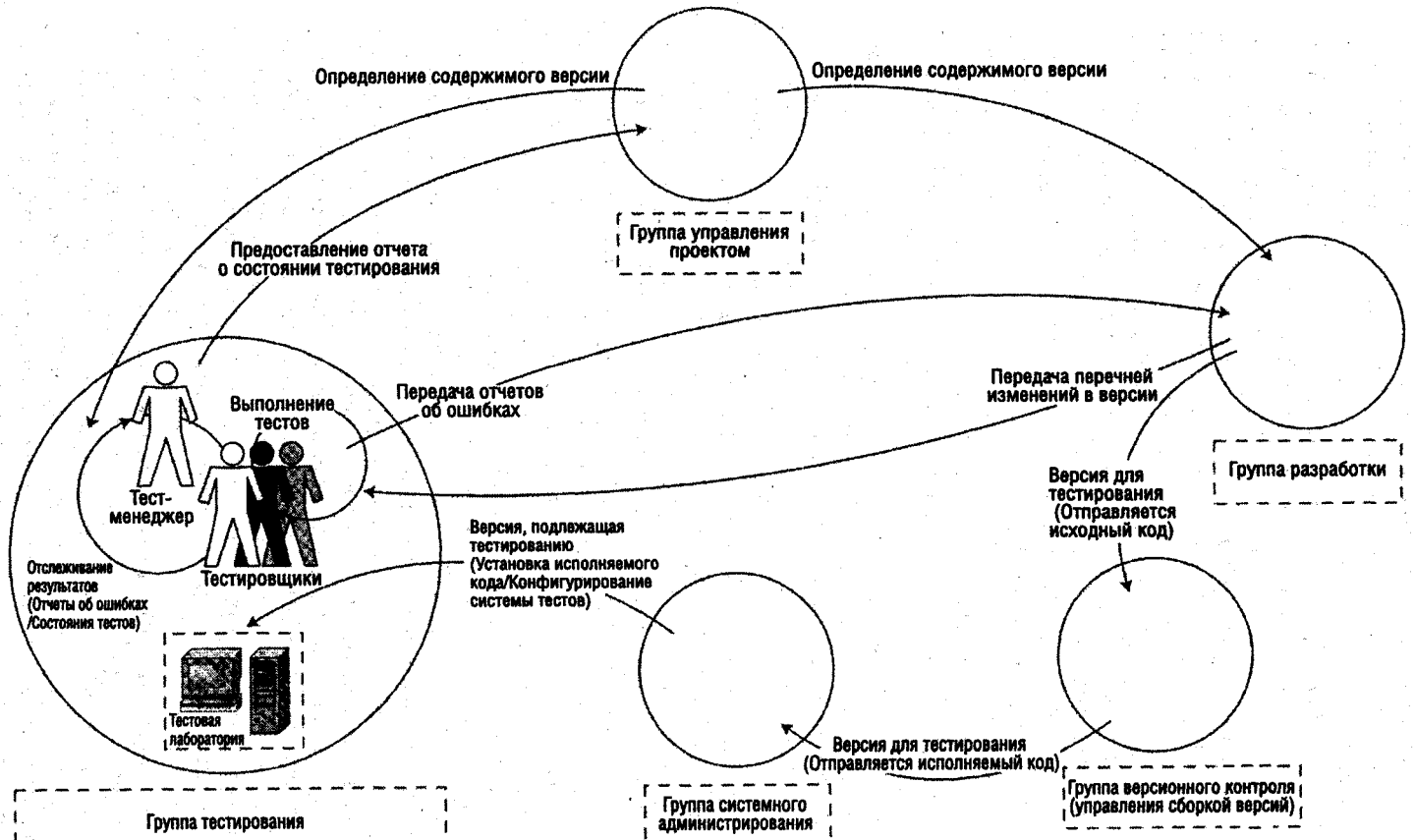


Рис. I.1. Поиск ошибок и их устранение с точки зрения процесса

Кроме того, тесная связь между самими процессами тестирования и между процессами тестирования и другими процессами в проекте порождает взаимозависимость. Каждый процесс влияет на ход проведения и результат всех остальных процессов. В итоге устанавливаются петли обратной связи. Проницательные тестировщики могут их использовать для расширения своего влияния. Приведу пример.

На рис. I.1 видно, что качество процесса подготовки отчетов об ошибках влияет на качество отчетов о состоянии тестирования, передаваемых руководству. Проектная группа оценивает качество отчетов о состоянии тестирования часто на основании того, содержат ли данные отчеты полезную для нее информацию. При этом имеет значение не только объективная "полезность", но и ее восприятие. Если отчеты о состоянии тестирования *действительно полезны* и *воспринимаются как полезные*, то складывается мнение, что в тестирование выгодно вкладывать средства. Подобное отношение, в свою очередь, укрепляет доверие к группе тестирования, в результате чего она получает больше необходимых для работы ресурсов. Эти ресурсы являются предпосылкой для хорошего проведения процессов тестирования, включая подготовку отчетов об ошибках.

Ключевые процессы тестирования, взятые в контексте

Ключевые процессы тестирования протекают в контексте проекта тестирования, который в свою очередь входит в объемлющий проект разработки, сопровождения, интеграции и приемо-сдаточных испытаний системы. Выявление и изучение ключевых процессов тестирования позволяет наилучшим образом выполнять проекты тестирования. По мере того как мы будем исследовать процессы в контексте подпроекта тестирования, не забывайте, что мы наблюдаем за совокупностью активностей, которые со временем завершаются и предоставляют проектной группе на протяжении всего проекта информацию и полезные услуги тестирования. Поэтому я буду исследовать ключевые процессы тестирования в хронологическом порядке: планирование, подготовка, проведение и совершенствование.

Часть I — Планирование — описывает эти процессы.

1. Вникните в операционный (система, проект, процесс) и организационный контексты предстоящего тестирования.
2. Определите риски качества системы и проранжируйте их; добейтесь, чтобы все заинтересованные стороны пришли к единому мнению по поводу объема и охвата тестирования, это позволит снизить риски.
3. Оцените потребности во временных и людских ресурсах и в денежных средствах на проведение тестирования в оговоренных ранее рамках и заручитесь поддержкой руководства.
4. Разработайте план мероприятий, состав участников, определите взаимозависимости, необходимые для проведения тестирования, и добейтесь поддержки этого плана у заинтересованных сторон.

Часть II – Подготовка – исследует данные процессы.

5. Привлеките в команду новых тестировщиков и добейтесь повышения квалификации группы тестирования. То есть посредством найма и обучения обеспечьте наличие необходимых навыков у ее участников, добейтесь надлежащего отношения к делу и мотивации.
6. Спланируйте, создайте, проверьте и утвердите систему тестирования (тестовые сценарии, тестовые данные, инструменты тестирования, среда тестирования, процедуры проведения тестов и т.д.), которой будет придерживаться группа тестирования для оценки качества тестируемой системы.

В части III – Проведение – вы столкнетесь со следующими процессами.

7. Получите и установите версию для тестирования, содержащую некоторые или все компоненты тестируемой системы.
8. Определите, проведите, последите и осуществите руководство комплектом тестов, предназначенным к исполнению в каждой тестовой версии.

Часть IV – Совершенствование – обсуждает остальные процессы.

9. Документируйте ошибки, выявленные в ходе проведения тестирования.
10. Проинформируйте основные заинтересованные стороны о результатах тестирования.
11. Подстройтесь под изменения контекста проекта и усовершенствуйте процесс тестирования.

Гипотетический проект "Суматра"

Для иллюстрации процессов на протяжении всей книги используется выдуманный проект. Данный проект представляет собой новую версию SpeedyWriter – приложения по обработке текстов на основе браузера, которое работает в основных web-браузерах и поддерживается операционными системами Windows, Linux, Solaris и Mac. Приложение реализовано на Java, использует web-сервер, сервер приложения, а также файловый сервер и сервер баз данных. Разработчиком SpeedyWriter является Software Cafeteria – средних размеров компания, которая занимается разработкой программного обеспечения и располагается в Остине, штат Техас. Software Cafeteria работает над версией 3.1 данного продукта, получившего кодовое название "*Суматра*" (Sumatra). Выпуск версии на рынок США по графику намечен на начало следующего года, в последующие 3 – 6 месяцев планируется выпустить версии для основных иностранных рынков[1].

---

[1] Имена и фамилии участников учебной ситуации отражают соотношение половой и национальной принадлежности, обычное для мира высоких технологий. Я придумывал участников, диалоги и рассказы, исходя из моего профессионального опыта.

Я стремлюсь следовать структурированному, хорошо документированному подходу к тестированию в большинстве проектов. Выбранный мной в качестве примера проект иллюстрируется различными цифрами и таблицами, взятыми из документов. Эти документы можно бесплатно скачать с моего сайта. Таким образом, изучаемый пример является одновременно и рассказом-описанием, и комплектом передаваемых документов тестирования. Вы имеете возможность ознакомиться с учебной ситуацией, принять решение об использовании предложенного мной процесса и получить некоторые инструменты для претворения данного процесса в жизнь. Я приложил много сил, чтобы эти документы были качественными и могли бы стать для вас важным подспорьем[1].

Терминология

Читая данную книгу и изучая документы, вы можете встретить незнакомые термины, а также знакомые термины, употребляемые мной в значении, отличном от того, к которому вы привыкли. В мире тестировщиков существует много диалектов. У нас нет общепринятого словаря, не оговорен и минимально необходимый набор знаний, что приводит к трудностям при обсуждении того, чем мы занимаемся. У бухгалтеров есть общеизвестные бухгалтерские проводки и операции. Доктора пишут диагнозы в медицинских картах пациентов, а другие врачи понимают сделанные записи и используют те же названия для обозначения соответствующих проблем. А мы, тестировщики, пока не имеем такой возможности.

Поэтому каждый раз, прибегая к термину, который, как мне кажется, может внести неразбериху, я формулирую, что под ним понимается. Я не пытаюсь создавать *определения*, а лишь хочу познакомить вас с *моими определениями*. Они представлены в виде врезок обособленного текста недалеко от того места, где тот или иной термин впервые встречается в книге. Для примера здесь приводится определение термина "система". Все определения собраны в глоссарии, приведенном в конце книги.

Относительно использования данной книги

В процессе чтения этой книги вы познакомитесь с процессами, изучите учебную ситуацию и примеры и развеете все неясности по поводу используемых мною терминов. Значит ли это, что, завершив чтение, вы будете готовы к тому, чтобы немедленно начать стремительные изменения в процессах тестирования, как в ваших личных, так и для всей группы? Не совсем. Не так быстро. Как уже говорилось, многие процессы тестирования, в том числе ключевые, о которых я буду рассказывать, являются контекстно-зависимыми. Стремясь помочь вам проложить курс на улуч-

---

[1] Замечу, что эти документы относятся к вымышленной учебной ситуации. Данные и измерения разумны, но, возможно, ваши измерения будут существенно отличаться от приведенных в книге. Кроме того, данные, используемые мной в различных примерах, приведены в объеме, необходимом лишь для пояснения положений, обсуждаемых в книге, и не более того.

Система, система, предназначенная для тестирования, разрабатываемая система, система на сопровождении (System, System under Test, System under Development, System Being Maintained)

Цельная система (программное обеспечение, аппаратное обеспечение, аппаратно-программное обеспечение и т.д.) с точки зрения тестирования. Часто состоит не только из очевидных компонентов. Например, в случае офисных приложений в дополнение к исполняемым модулям на рабочем месте пользователя могут прибавиться прикладные части, выполняемые на сервере, которые могут иметь разнообразные интерфейсы с продуктами третьих сторон, а также процессы документирования, поставки и обновления, онлайновые базы данных и средства поддержки, компоненты аппаратного обеспечения, встроенные программы и даже инфраструктура, включая LAN, WAN и Интернет. Слово "продукт" до некоторой степени более общий, однако может быть некорректно понят как исключительно прикладная программа, предназначенная для массового рынка.

Система может быть также *системой систем*, где каждая система взаимодействует с другой и зависит от одной и более систем, и все системы существенны для работы системы систем. Например, приложение на основе браузера вместе с клиентскими программами, web-сервером, сервером приложений, сервером баз данных есть система систем.

Наконец, система может быть *семейством систем*, где каждая система взаимодействует с одной или несколькими системами, но зависимости ограничены, и система может применяться как самостоятельно, так и в интеграции с другими системами. Например, Microsoft Office — это семейство систем программного обеспечения.

шение процесса, я завершаю обсуждение каждого процесса тремя основными разделами.

Построение успешного процесса

Как определить успешные процессы? Можно выявить поведение группы тестирования и ее достижения, указывающие на степень успешности процесса. Иными словами, что делает группа тестирования, когда применяет успешные процессы, и какие преимущества она от этого получает? Если, к примеру, вы начнете заниматься физическими упражнениями по определенной программе, то, достигнув успеха в этом деле, вы получите увеличение мышечной массы, снижение жировой прослойки и станете выносливее. Для закрепления и углубления полученных результатов вы, возможно, начнете заниматься спортом на регулярной основе, правильно питаться и бросите курить. В тех 12 процессах, что нам предстоит рассмотреть, подобные действия и достижения находятся во взаимодействии и подпитывают друг друга.

Как правило, существует целый спектр подобных действий и достижений. Например, занимаясь по оздоровительной программе около 8 лет,

я продолжал курить. При этом я действительно добился поставленных целей. Однако, беспокоясь о влиянии курения на здоровье в долгосрочной перспективе и будучи несколько удивлен низкой выносливостью при беге, я бросил курить. В результате всего через пару месяцев я стал пробегать милю на минуту быстрее.

Как понять, как работают ваши текущие процессы? Возможно, в изменениях нет необходимости. Как узнать, позволит ли контекст успешно внедрить предполагаемые изменения? В этом вам помогут разделы, посвященные построению успешных процессов.

Решение проблем

Во многих случаях складываются ситуации, способные создать преграды и помехи для конкретных процессов. Типичным примером являются нереалистичные ожидания со стороны руководства. Некоторые считают, что тестировщики прячут в рукавах коробочки с волшебным порошком. Стоит только посыпать этим порошком систему, и все ошибки вылезут наружу. Кто-то из руководителей примет реалистичное объяснение, каким на самом деле может быть вклад тестирования в проект. Другие будут настаивать на том, что хорошо справляющиеся со своим делом тестировщики способны устранить все проблемы. Подобные ожидания чрезвычайно осложняют переговоры с руководством по поводу принятия им реалистичных графика и бюджета тестирования. Если вы намерены внести изменения в организацию и в способ проведения тестирования, очень важно понять эти трудности и знать, как с ними бороться.

Внедрение усовершенствований

Улучшение процесса нередко является сложной задачей. Например, внедрение выстроенного процесса управления дефектами, включая грамотный выбор подходящего инструмента по отслеживанию ошибок, может потребовать лично от вас нескольких недель, а в целом занять больше месяца. Кроме того, наряду с практическими действиями требуется еще и умение убеждать. Порой получить согласие на внесение изменений в процесс сложнее, чем изменить его. Часто подавляющее большинство ратует за сохранение статус-кво и не поддерживает ваших предложений, какими бы хорошими они ни были. Поэтому, поставив цель что-то изменить, вам придется идти на риск, обосновывать изменения и прикладывать много сил, чтобы дело не застопорилось. Наконец, процесс усовершенствования требует наличия продуманного плана действий. Существует даже процесс для достижения изменений в процессе! Я не могу дать рецепт, который сработает в любых обстоятельствах, но постараюсь изложить некоторые идеи, над которыми стоит поразмышлять, планируя преобразования.

Процессы не гарантируют успеха, а, скорее, подготавливают для этого почву, при условии, что они подготовлены должным образом. Неверно разработанные процессы могут стать обременительными и потребовать много сил и времени на бумажную волокиту. К примеру, после переезда я попытался сменить мой рабочий адрес у моего бывшего регистратора

домена URL. Чтобы добиться желаемого, я в течение полугода отправил более двадцати факсов и электронных писем. В конце концов, я сдался и сменил регистратора. Процесс, существующий у нового регистратора, соответствовал моему представлению о качественном обслуживании. Разумеется, мы не хотим чинить препятствия на пути к предоставлению качественных услуг по тестированию. Идея состоит в том, чтобы и процесс, и структура присутствовали в оптимальном объеме, ни больше, ни меньше, выступая основой для действительно блестящей работы и подпитывая ее.

Но наряду с использованием верного подхода к изменению процесса вам потребуется выбирать, к чему прикладывать силы. Если, прочитав последнюю страницу, вы решите немедленно изменить каждый из обсуждаемых в книге процессов, вас ждет поражение. Что еще хуже, эта неудача приведет к путанице в тех делах, которые прежде шли хорошо, а это, вероятно, будет стоить вам "потери лица". Поэтому в конце книги я расскажу о том, как планировать надлежащие изменения в процессах, более всего в них нуждающихся.

Успехов на ниве тестирования!

Если предметом данной книги являются процессы тестирования, то в качестве цели чтения выступает понимание того, как добиться больших успехов в выполнении проектов тестирования. Процессы являются всего лишь средством для достижения этой цели. Так, большинство людей собирают кулинарные книги не для того, чтобы они пылились на полках. Они покупают эти книги, чтобы успешно приготовить то или иное вкусное блюдо. Заглянув ко мне на кухню, вы легко догадаетесь, какие из кулинарных книг мои любимые. Они пахнут специями, забрызганы растительным маслом и потрепаны от частого использования. Они сразу откроются на страницах с моими любимыми рецептами, по которым я люблю готовить снова и снова.

Шеф-повара ценят не за следование процессам приготовления блюд. Его мастерство не измеряется строгостью следования рецептам. Шеф-повар окажется мастером своего дела, если вновь и вновь будет преподносить гостям лакомые блюда в оговоренное время и за разумные деньги, потраченные на качественные продукты.

Точно так же данная книга призвана помочь вам добиться успеха в роли тестировщика. Надеюсь, что с годами она также обзаведется пометками, потрепанными страницами, а возможно, от частого использования кое-где порвется. Надеюсь, что приведенные мной списки контрольных вопросов, образцы и примеры станут вашими излюбленными "кухонными принадлежностями" для приготовления успешного проекта тестирования. Я буду считать, что написал удачную книгу, если она поможет вам стать более уверенными и успешными профессионалами в области тестирования.

Часть I

Планирование

В первой части книги мы исследуем процессы, связанные с пониманием того, какие работы по тестированию нужно выполнить. Это самая большая по объему часть книги, поскольку на этом этапе процесса тестирования мы закладываем основы успеха. Сначала мы исследуем место работ по тестированию в проекте, затем изучим риски, которые тестирование позволяет снизить. Заложив фундамент, мы исследуем еще два процесса: оценку и планирование. Разобравшись с графиками и затратами и оставив позади детали стратегии и тактики, мы перейдем к следующей части книги — подготовке успешного тестирования.

№ этапа	Этап	Выполнено?
1.	Планирование: понимание места тестирования.	☐
1.A	Определить функциональный (система, проект и процесс) и организационный контексты, в рамках которых выполняется тестирование.	☐
1.B	Определить и расставить приоритеты рисков качества системы, достичь консенсуса заинтересованных сторон проекта относительно возможностей тестирования по смягчению этих рисков.	☐
1.C	Оценить временные, ресурсные и финансовые затраты, необходимые для выполнения тестирования и согласованные с пунктом 1.B, получить поддержку оценки у руководства.	☐
1.D	Запланировать задачи, зависимости и состав участников, которые необходимы для смягчения рисков качества системы, и получить поддержку плана у заинтересованных сторон проекта.	☐

ГЛАВА 1

Панорамный обзор: роль тестирования в более широком контексте

Зачем мы здесь? Нет, я не спрашиваю, зачем люди живут на Земле; я спрашиваю, зачем компании финансируют проекты, которые требуют тестирования?

Компании разрабатывают, сопровождают или приобретают системы, чтобы поставлять какие-либо значимые продукты, услуги или возможности для некоторой группы клиентов, пользователей и других заинтересованных лиц. Передаваемые ценности приобретают различные формы: от развлечений (видеоигры) до производства (офисные приложения), от обороны (системы вооружений) до борьбы с заболеваниями (медицинские устройства). Заказчики, пользователи и другие заинтересованные лица ожидают (обоснованно и не очень) получить систему определенного качества. Масштаб этих ожиданий простирается от "хотя бы как-то работало" до "полное отсутствие дефектов". В целом заказчики, пользователи и другие заинтересованные лица рассчитывают, что системы будут демонстрировать преимущественно удовлетворительную работу.

Даже искренне добиваясь высокого качества в ходе разработки системы, ее сопровождения и поставки, люди совершают ошибки. Эти ошибки снижают уровень качества системы относительно ожиданий. Разница между ожидаемым качеством и получаемым качеством проявляется в рисках для компании-разработчика, а также для заказчиков, пользователей и других заинтересованных лиц.

Эти риски могут мотивировать компанию на инвестиции, связанные с качеством, включая инвестиции в качество. В частности, организации могут направить усилия на:

- Уменьшение общих затрат, связанных с проблемами качества
- Повышение доверия к функциям системы
- Снижение вероятности событий, угрожающих прибыли, достижению цели и даже жизни

Заказчики, спонсоры (Customers, Sponsors)

Люди, финансирующие создание системы любым из следующих способов: непосредственная оплата разработки системы, участие в финансировании системы, финансовая поддержка производителя системы, приобретение продукта или услуги, содержащей или использующей систему, оплата налогов, прочие прямые и косвенные способы финансирования. Заказчик — это организация, приобретающая и/или поставляющая систему для пользователей, являющихся сотрудниками организации, или частное лицо, приобретающее систему для личного пользования.

Пользователи (Users)

Люди, которые работали и, возможно, будут работать с системой, а также люди, применяющие результаты работы системы. Иногда пользователи являются заказчиками системы, но не всегда.

Заинтересованные (в системе) стороны (Stakeholders (in the system))

Лица как внутри компании, так и вне ее, на которых влияют в настоящем, повлияют или могут повлиять в будущем ход проекта, разрабатываемая система и ее качество.

Качество (Quality)

1. Пригодность для использования. Наличие свойств, атрибутов и функций, удовлетворяющих ожиданиям заказчиков и пользователей, и отсутствие свойств, атрибутов и функций, неудовлетворяющих ожиданиям пользователя.
2. Соответствие требованиям. Наличие свойств, атрибутов и функций, удовлетворяющих всем установленным требованиям, и отсутствие свойств, атрибутов и функций, отклоняющихся от требований. (Данное определение применимо в конкретном контексте, в особенности для проектов, выполняемых по контракту.)[1]

- Повышение репутации компании касательно качества самой системы и качества продуктов и услуг, которые она поддерживает
- Получение повторных и связанных заказов от удовлетворенных работой компании заказчиков
- Снижение числа судебных исков из-за дефектов системы (для компаний-разработчиков) и исков заказчиков (для контролирующих компаний)

---

[1] Первое определение взято из книги Дж. М. Джурана (J. M. Juran) "*Juran on Planning for Quality*". Второе определение позаимствовано из книги Филиппа Кросби (Philip Crosby) "*Quality Is Free*". Определение Кросби неявно следует из определения Джурана в случае, если процесс сбора требований организован и выполнен верно. Если в ходе процесса сбора требований были допущены отклонения, второе определение менее полезно. Однако в случае разработки заказных проектов имеет больший смысл рассматривать качество как соответствие требованиям и увязывать контракт, требования и тестирование с учетом данного определения.

- Соблюдение планов-графиков, бюджетов или достижение целей, связанных с промышленными стандартами, для рабочей версии или поставки системы
- Повышение качества планирования, графиков и бюджетов для будущих проектов
- Получение финансов для текущих и будущих проектов
- Снижение вероятности несдачи системы заказчику, что может негативно сказаться на конкурентоспособности компании

Мотивация может зависеть от того, говорим мы о тестировании силами компании-разработчика, создающей систему, или о тестировании силами внешней компании, рассматривающей вопрос о приобретении, интегрировании или поставке системы[1].

Хотя данные предыдущих проектов и модели могут оказать помощь в прогнозировании числа дефектов (см. главы 3 и 4), точное число, рамки, серьезность и локализация проблем качества не известны заранее. Таким образом, эти потенциальные проблемы являются рисками (вероятными негативными результатами или событиями). Риски подвергают опасности качество системы. Эффективно организованное тестирование предоставляет услуги и информацию, способствующие управлению рисками качества разрабатываемой, сопровождаемой или поставляемой системы.

Для предоставления компании услуг тестирования и информации служат процессы тестирования в рамках проектов разработки, сопровожде-

Ожидание качества (Expectation of Quality)

Убежденность заказчика/пользователя в уровне качества, который должна обеспечивать система. В идеале заказчики и пользователи должны иметь обоснованные ожидания качества системы.

Необоснованные ожидания обычно вызывают остановку таких действий, как сбор требований, бизнес-анализ и управление изменениями. Часто невозможно и, как правило, нежелательно пытаться следовать необоснованным ожиданиям качества при проведении тестирования.

Опыт качества (Experience of Quality)

Мнения о качестве системы вместе с общим уровнем удовлетворенности пользователей/заказчиков по мере получения опыта работы с системой. Когда уровень опыта качества совпадает с ожиданием качества, пользователь/заказчик обычно удовлетворен. Если опыт качества превышает ожидания, пользователь/заказчик восхищается. Если опыт качества ниже ожиданий, пользователь/заказчик разочарован.

---

[1] Индивидуальные покупатели программных средств редко обладают правом в рамках соглашения о приобретении проводить приемочные тесты любого вида. В ходе разработки большинство контрактов на приобретение прикладных программ явно исключают подобные тесты. Для изучения этой и других диспропорций в распределении стоимости продуктов низкого качества на рынке прикладных программ я рекомендую просмотреть некоторые из работ Сема Канера (Cem Kaner), в том числе "*Bad Software*", и его статьи на сайте www.kaner.com в разделе UCITA (Uniform Computer Information Transactions Act).

Жизненный цикл системы (System Lifecycle)

Все множество действий, которые имеют место в ходе разработки, поставки, использования и устаревания системы, начиная с исходной спецификации и проектирования, далее включая реализацию, тестирование, поддержку, сопровождение и заканчивая остановкой и выводом системы из эксплуатации.

Жизненный цикл разработки системы (System Development Lifecycle)

Часть жизненного цикла системы, связанная с начальной разработкой системы, начиная со спецификации и проектирования и заканчивая реализацией, тестированием и первичной поставкой.

ния или поставки системы. (В компаниях, в которых есть отдельно работающие группы тестирования, некоторые активности выступают в качестве отдельных подпроектов, выполняемых такими группами.) Процессы тестирования — это последовательности действий, работ и наблюдений, предпринимаемых для достижения цели. На самом верхнем уровне процесс тестирования состоит из четырех шагов, которые могут накладываться друг на друга.

1. *Планирование* — понимание места тестирования.
2. *Подготовка* — сбор людей и тестов.
3. *Проведение* — выполнение тестов и сбор результатов.
4. *Совершенствование* — предоставление информации о проекте и его улучшение.

В идеале процесс тестирования начинается одновременно с возникновением замысла системы и продолжается до конца жизненного цикла системы (т.е. до списания, завершения существования или окончания поддержки системы).

Процесс тестирования представлен на рис. 1.1. Если вы изучали тотальное управление качеством (total quality management), то, вероятно, узнаете показанный на рисунке цикл как разновидность процесса "план, осуществление, проверка, действие" для непрерывного совершенствования качества, описанного Э. Демингом[1]. При грамотном проведении тестирования в ходе всего жизненного цикла системы не только непрерывно улучшается система, но в качестве желаемого побочного эффекта процесс тестирования поставляет информацию о других процессах жизненного цикла. Проектная группа может применять эту информацию для улучшения соответствующих процессов.

Ранее я перечислил несколько возможных мотивов проведения тестирования. Как именно процесс тестирования обеспечивает возврат вложе-

---

[1] См. Мэри Уолтон (Mary Walton) "*The Deming Management Method*".

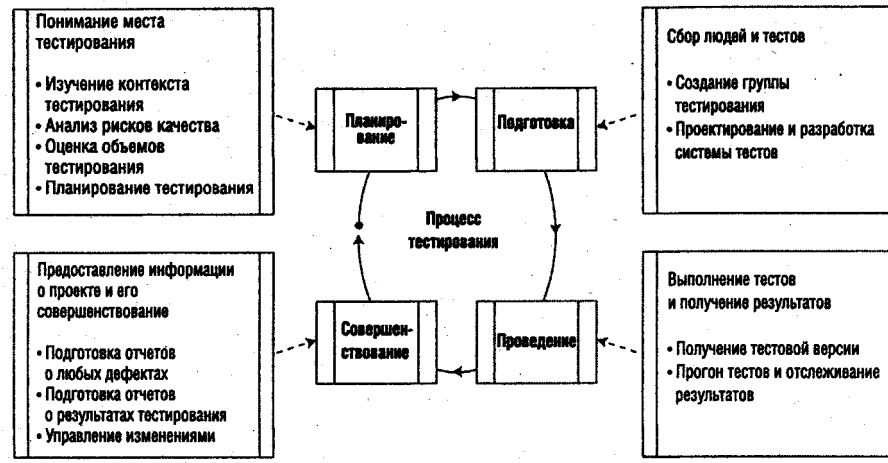


Рис. 1.1. Процесс тестирования

ний в него? В большинстве проектов тестирования это происходит четырьмя основными способами:

1. Обнаружение ошибок, которые будут устранены, или даже предотвращение их внесения.
2. Обнаружение ошибок, которые не будут устранены, но о которых становится известно.
3. Проведение тестов, которые снижают (потенциально затратные) риски.
4. Обеспечение проекта своевременной, точной и заслуживающей доверия информацией.

Группа тестирования привносит в проект эти преимущества в основном на четвертом шаге процесса (Совершенствование) за счет обсуждения результатов тестирования. Предыдущие шаги процесса имеют существенное значение для получения заслуживающей доверия, точной, своевременной и исчерпывающей информации для ключевых заинтересованных лиц, но сами эти действия не приносят компании прибыли. Уникальность группы тестирования во многих компаниях состоит в том, что ее деятельность направлена на получение и обмен информацией, предназначенной для использования только внутри компании, т.е. группа тестирования обеспечивает обратную связь с информацией о качестве системы и о проекте в целом[1].

[1] Это в равной степени верно и в случае, когда группа тестирования работает в рамках компании по разработке и сопровождению систем, и когда группа тестирования является частью компании, получающей программный продукт от внешних производителей или консалтинговых фирм. Однако в последнем случае ценность тестирования состоит практически исключительно в снижении рисков и обеспечении информацией о проекте (пп. 3 и 4).

Ошибка, дефект (Bug, Defect)

Проблема, которая вызывает или может вызвать неадекватное выполнение системой одного или нескольких обоснованных ожиданий качества заказчика/пользователя.

Поскольку процесс тестирования происходит в рамках проекта разработки, сопровождения или поставки, процесс тестирования встраивается в общий контекст проекта. Но это более органичная и подвижная операция в отличие, скажем, от включения некоторого устройства в штепсельную розетку. Наличие процесса тестирования в жизненном цикле системы изменяет жизненный цикл системы, а также проекты, которые следуют такому жизненному циклу. Процесс тестирования влияет сам и подвергается ответному влиянию со стороны проекта и системы, которые разрабатываются или сопровождаются в рамках проекта. Проектная группа использует информацию, полученную в ходе тестирования, в целях корректировки проекта и системы. Персонал компании отвечает за выполнение проекта, и некоторые из этих людей отвечают за процесс тестирования. Руководство определяет правила, оказывающие влияние на то, как группа тестирования взаимодействует с другими участниками проекта.

Процесс тестирования

На рис. 1.1 графически представлен процесс тестирования, но, по моему мнению, как показано в описании Процесса 1, более полезным представлением является список контрольных вопросов. Во многих проектах тестирования, в которых мне довелось работать, допускались значительные наложения и итерации в этом процессе. Не во всяком проекте тестирования есть каждый этап или подэтап. Однако этот общий процесс можно подстроить под конкретный проект. Рассматриваемая контрольная таблица процесса должна предоставить подходящую точку отсчета в случае, если планируется составить перечень настраиваемых процессов.

Если в одном проекте на различных фазах тестирования работают разные группы (что является типичным для больших проектов), можно рассматривать несколько вариантов процесса в ходе проекта на различных

Тестирование (Testing)

Функционально это процесс оценки качества системы, предоставления услуг и информации, позволяющий компании управлять рисками качества системы. Организационно это группа специалистов, предоставляющая данные услуги и информационные продукты для проекта, в котором она занята.

Процесс 1. Процесс тестирования

№ этапа	Этап	Выполнено?
1.	Планирование: понимание места тестирования.	☐
1.A	Определить функциональный (система, проект и процесс) и организационный контексты, в рамках которых выполняется тестирование.	☐
1.B	Определить и расставить приоритеты рисков качества системы, достичь консенсуса заинтересованных сторон проекта относительно возможностей тестирования по смягчению этих рисков.	☐
1.C	Оценить временные, ресурсные и финансовые затраты, необходимые для выполнения тестирования и согласованные с пунктом 1.B, получить поддержку оценки у руководства.	☐
1.D	Запланировать задачи, зависимости и состав участников, которые необходимы для смягчения рисков качества системы, и получить поддержку плана у заинтересованных сторон проекта.	☐
2.	Подготовка: сбор людей и подготовка тестов.	☐
2.A	Создать команду специалистов-тестировщиков, обладающих соответствующими знаниями, навыками, отношением к делу и мотивацией, посредством поиска персонала и его обучения.	☐
2.B	Спроектировать, разработать, получить и верифицировать систему тестов, которую группа тестирования будет использовать для оценки качества системы, предназначенной для тестирования.	☐
3.	Проведение: выполнение тестов и сбор результатов.	☐
3.A	Получить и установить тестовую версию, состоящую из всех или нескольких компонентов системы, предназначенной для тестирования.	☐
3.B	Определить, отслеживать и управлять комплектом тестов, с помощью которого планируется проверять каждую из тестовых версий.	☐
4.	Совершенствование: предоставление информации о проекте и его улучшение.	☐
4.A	Документировать ошибки, обнаруженные в ходе выполнения тестов.	☐
4.B	Обсудить результаты тестирования с ключевыми заинтересованными лицами.	☐
4.C	Управлять изменениями и уточнить процесс тестирования.	☐

уровнях формализации. Например, при разработке одной системы для массового рынка группа разработки проводила неформальное модульное и компонентное тестирование, группа маркетинга следила за ходом альфа- и бета-тестирования, а группа тестирования под моим руководством проводила формальное комплексное и системное тестирование. Еще пример: мы работали в проекте по разработке системы для внутреннего примене-

Модульное тестирование (Unit Testing)

Тестирование небольшого элемента или модуля системы.

Компонентное тестирование, тестирование подсистем (Component Testing, Subsystem Testing)

Фаза тестирования, ориентированная на проверку компонентов системы и/или ее подсистем по отдельности.

Альфа-тестирование, бета-тестирование (Alpha Testing, Beta Testing)

Фазы тестирования, в ходе которых система подвергается проверке со стороны внутренних (альфа) или внешних (бета) заказчиков и пользователей в реальных условиях, на настоящих данных и в реальной среде.

Комплексное тестирование, тестирование продукта (Integration Testing, Product testing)

Фаза тестирования, в ходе которой проверяются взаимосвязи и интерфейсы между парами и группами компонентов или подсистем системы.

Системное тестирование (System Testing)

Фаза тестирования, на которой проверяется поведение системы как в целом, так и в частностях, и ее функциональность.

Приемо-сдаточные испытания, пользовательское приемочное тестирование (Acceptance Testing, User Acceptance Testing)

Фаза тестирования, целью которой является проверка того, что система удовлетворяет требованиям. Название "пользовательское приемочное тестирование" может указывать на то, что тестирование проводится силами пользователя.

Пилотное тестирование (Pilot Testing)

Фаза тестирования, на которой система подвергается опытной эксплуатации под тщательным наблюдением. Пилотные тесты аппаратной части показывают возможности сборочной линии по массовому производству системы. Пилотные тесты программной части показывают возможности системы по обработке типичных операций реальных пользователей на реальной аппаратуре.

ния в большом финансовом учреждении. Разработчики проводили формальное компонентное тестирование, группа тестирования выполняла формальное комплексное и системное тестирование, бизнес-аналитики и пользователи отвечали за неформальные приемо-сдаточные испытания, а сетевые операции подвергались пилотному тестированию.

В этой главе мы изучим первую часть процесса — понимание контекста. Понимание контекста очень важно, особенно для тестирования, а именно для предоставления услуг и своевременного информирования

участников проекта. Работа тестировщиков должна соответствовать более широкому контексту: проекту и проектным процессам до мельчайших подробностей. Если мы хотим завоевать доверие и иметь перспективы на будущее в компании, то должны быть компетентными тестировщиками, которые знают, как сделать все возможное для компании, на которую мы работаем.

Примером того, что невозможно управлять ситуацией, не зная контекста, являются последние дни генерала армии США Армстронга Кастера. Посмертную славу принесла Кастеру в основном его роль в войне между Соединенными Штатами и индейцами в конце XIX века. Его лебединой песней в этот печальный период американской истории стало участие в битве при реке Литтл-Бигхорн (Battle of Little Big Horn), когда он был уверен, что преследует небольшой отряд индейцев Сиу. На самом деле он шел по следу большого войска Сиу под командованием Сидящего Быка и Неистового Коня. Кроме того, к большому огорчению Кастера, Сиу собрались на общий совет, где решался вопрос о противостоянии продолжающимся попыткам Соединенных Штатов насильственно отправить их в резервации. Нетрудно понять, что тема (контекст) совета уже соответствующим образом настроила его участников против непрошенных гостей. Кастер и 264 его подчиненных были убиты в последовавшем сражении, к которому их принудили индейцы, полностью окружив на холме. Ошибка в выборе места, ошибка во времени, ошибка в выборе цели, ошибка в подборе людей. Незнание ситуации ведет к неверным выводам, а неверные выводы могут быть очень опасны[1].

Понимание функционального и организационного контекста

Некоторые тестировщики и тест-менеджеры уже назначены на свои должности, когда организуется компания. Они начинают с того, что определяют организационный и функциональный контекст тестирования с нуля. Однако многие специалисты-тестировщики присоединяются к уже существующей компании или подразделению либо для организации нового процесса тестирования, либо для работы в существующем процессе. Помимо праздных дискуссий о лучшем организационном и функциональном контексте, наиболее приближенным к практике вопросом является следующий: каким образом организационный и функциональный контекст для тестирования и тестировщики становятся такими, какими они должны быть? Контекст тестирования обычно существует недолго, по-

[1] Битва при Литтл-Бигхорн – не единственный эпизод неустрашимости Кастера. В битве при Геттисберге он возглавил кавалерийское подразделение против войск Конфедерации: его лошадь была убита, а сам он по счастливой случайности едва избежал смерти. Смелость всегда помогала Кастеру, который уже был (самым молодым на тот момент) генералом армии США. Битва при Литтл-Бигхорн стала последним и, вероятно, наиболее опрометчивым деянием в его жизни, полной риска. Эта история преподносит еще один урок профессионалам-тестировщикам: рано или поздно привычка к большому риску приводит к большим потерям. Об удивительном характере и жизни генерала Кастера можно прочитать в биографической книге Джеффри Верта "*Кастер*".

этому хороший тестировщик или тест-менеджер должен понимать его и уметь работать в нем.

Группы, отвечающие за проведение тестирования, возникают внутри группы разработки или сопровождения по множеству причин. Иногда просвещенный менеджмент понимает с самого начала значение эффективного и качественного тестирования. В некоторых случаях заказчик, рынок, правовые и регулирующие требования вынуждают компанию принять некоторую стандартизованную форму тестирования. В других случаях компания, прошедшая период игры в рулетку качества, поставлявшая заказчикам системы либо с минимальным тестированием силами программистов, либо вообще без оного, наконец получает убийственный опыт, когда плохая версия становится предметом судебного иска, теряется важный заказчик, появляются плохие отзывы в прессе либо уходят наиболее ценные сотрудники.

Понимание таких историй — важная часть формирования процесса тестирования, тщательно встроенного в проект. Чтобы увидеть причины, сравним тестировщиков с другими участниками проекта. Что делают проектировщики, бизнес-аналитики, программисты и другие специалисты в проекте? Создают систему. Система не появится без этих людей. Даже если компания выберет оффшорную разработку, систему должны создать где-то в другом месте другие люди. Аутсорсинг не меняет способа, каким разработчики системы создают стоимость. Он только усложняет процесс и в некоторых случаях приводит к тому, что создаются более дешевые или более качественные системы... или наоборот!

Ранее были выделены четыре основных наиболее ценных типа информации и услуг, поставляемых тестировщиками. Толковые менеджеры проектов могут использовать эти услуги и информацию для снижения общего риска неудачи проекта. Однако в некоторых компаниях менеджеры, вероятно, не видят этих преимуществ либо, по ошибке, верят, что тестировщики могут предоставить другие ценные преимущества, например, волшебным образом заставят исчезнуть ошибки.

Читатель, как и я, возможно, время от времени с немалым удивлением обнаруживает непонимание того, что на самом деле могут тестировщики. Иногда бывалые тестировщики, читая книги и статьи о "правильном тестировании", разводят руками, говоря: "Не могу поверить, что эти люди делают всю эту чушь!" По поводу понимания возможных проблемных ситуаций Джеральд Вайнберг прозорливо писал в *"The Secrets of Consulting"*: "Вещи таковы, поскольку они стали такими... В свое время были серьезные и достаточно обоснованные причины для принятия решений, которые сейчас кажутся нелепыми". Высказывание Вайнберга применимо ко всем, кто впервые попадает в существующую сферу деятельности с намерением оказать влияние, изменить или просто понять, как она работает.

Итак, что можно сделать в случае, если вы рассматриваете ситуацию, которая кажется рациональной или наоборот неразумной, чтобы направить ваши усилия в верном направлении? Процесс 2 — это процесс исследования контекста.

Считаю, что этапы этого процесса идут параллельно и что каждый этап обычно нужно проходить несколько раз. Кроме того, полагаю, что у меня острый, восприимчивый ум. Мне нравится, когда ситуация в том

виде, как она представляется моим коллегам, становится ясной мне за два-три дня. Я стараюсь не высказывать своего мнения как можно дольше, пока я впитываю новую информацию о том, как вещи стали такими, как они есть, и что люди думают о том, что я могу сделать в этой ситуации.

Процесс 2. Процесс исследования контекста

№ этапа	Этап	Выполнено?
1.	Понять процессы жизненного цикла, применяемые в вашей компании для разработки, сопровождения или поставки системы, включая любые планируемые или осуществляемые инициативы по совершенствованию процессов.	☐
2.	Изучить всю документацию, относящуюся к тестированию и качеству, данные, метрики и измерения, в том числе планы тестирования, базы обнаруженных дефектов, данные об ошибках, найденных при эксплуатации, системы тестов и т.п. Понять, кто разработал все эти артефакты, почему они это сделали, почему они это сделали таким образом и что привело к таким решениям и таким результатам в компании.	☐
3.	Обсудить с другими участниками проекта, в каких действиях тестирования они участвуют и что они собираются делать дальше с учетом того, что вы тоже участвуете в работе группы тестирования.	☐
4.	Определить заинтересованных лиц своего уровня в процессе тестирования. Провести с ними переговоры для понимания взаимоотношений с теми, кто сейчас проводит тестирование, их ожиданий от процесса тестирования, предыдущих разочарований от процесса тестирования, конфликтов с теми, кто участвует в тестировании и т.п.	☐
5.	Уточнить с непосредственными руководителями и старшими менеджерами их ожидания от процесса тестирования и от вашего участия в тестировании и в оценке качества.	☐

Оценка качества (*в отличие от* тестирования)
(Quality Assurance (*contrasted with* testing))

В соответствии со стандартом IEEE 610.12-1990 стандартный глоссарий терминологии по программированию ("IEEE Standard Glossary of Software Engineering Terminology") говорит: оценка качества (QA) включает в себя "всякую деятельность, необходимую для обеспечения достаточной уверенности" в качестве системы и оценку "процесса, с помощью которого [система] разрабатывалась", тогда как тестирование состоит из действий, необходимых для "обнаружения различий между существующими и требуемыми условиями (ошибок) и оценки... свойств". Другими словами оценка качества фокусируется на корректности процесса, а тестирование — на установлении качества[1].

[1] См. глоссарий, определяющий эти и другие стандартные термины по программированию в "*IEEE Software Engineering Standards Collection: 1997 Edition*", Standard 610.

Присоединение к любой из существующих групп людей: клубу, религиозной конфессии или компании — это исключительно социальный акт. Личность оказывает влияние на то, каким образом люди социализируются. Некоторые люди используют в качестве метода предприимчивость. Вероятно, мой процесс слишком осторожный, чтобы его рассматривали смельчаки. В прошлом я был предприимчивее, но большая часть моих попыток проявить бесстрашие оказалась неудачной. Чем внимательнее я слушаю, тем лучше я делаю. На этой ноте приступим к первой части учебного примера — проекта "Суматра".

Представление проекта "Суматра" и группы тестирования приложения SpeedyWriter

Джамал Браун — менеджер группы комплексного и системного тестирования компании Software Cafeteria. Software Cafeteria представляет на рынке интегрированный пакет офисных приложений на базе web-технологий (OfficeArrow), состоящий из текстового процессора (SpeedyWriter), средств обработки таблиц (JetCalc), инструмента подготовки презентаций (ZippyShow) и системы управления реляционными базами данных (LightningData). Группа Джамала предоставляет услуги тестирования для всей линии продуктов. Он отвечает за тестирование как новых, так и сопровождаемых версий; тестирование последних происходит как для ежемесячно выходящих пакетов, так и при выпуске аварийных патчей.

Группа комплексного и системного тестирования, которой руководит Джамал, входит в состав отдела поддержки программных разработок. Помимо тестирования, отдел предоставляет различные услуги группе системной разработки, в частности подготовку технической документации, сборку версий, поддержку внутренних систем, инфраструктуры и сети, поддержку пользователей. Группа системной разработки отвечает за разработку и сопровождение линии продуктов OfficeArrow. Также есть отдельные группы маркетинга, продаж и финансовая группа. Структура компании Software Cafeteria показана на рис. 1.2.

2 сентября 2002 г. группа комплексного и системного тестирования участвовала в общем собрании сотрудников Software Cafeteria. В ходе обсуждения будущих планов по развитию продукта Кейт Эрнандес, менеджер продукта SpeedyWriter, объявила, что основные работы по подготовке версии 3.1 с кодовым наименованием "Суматра" начнутся в ближайшие недели. В Суматру, пояснила она, должны войти несколько инкрементных улучшений функций обработки таблиц, импорта файлов и колонтитулов. В качестве основной новинки в нее войдет в качестве опции подсистема документооборота.

Затем она представила менеджера проекта "Суматра" Дженни Кауфман, которая приобрела авторитет в качестве ведущего программиста в ходе разработки версии 3.0. Дженни пояснила, что, поскольку ключевым моментом является поставка Суматры на рынок в первом квартале 2003 года, в проекте будет использоваться инкрементная модель разработки. Группа программистов в ходе нескольких итераций разработает функциональное ядро (подсистему документооборота), а затем добавит оставшиеся функции. Версии, получаемые по завершении каждой из итераций, должны работать стабиль-

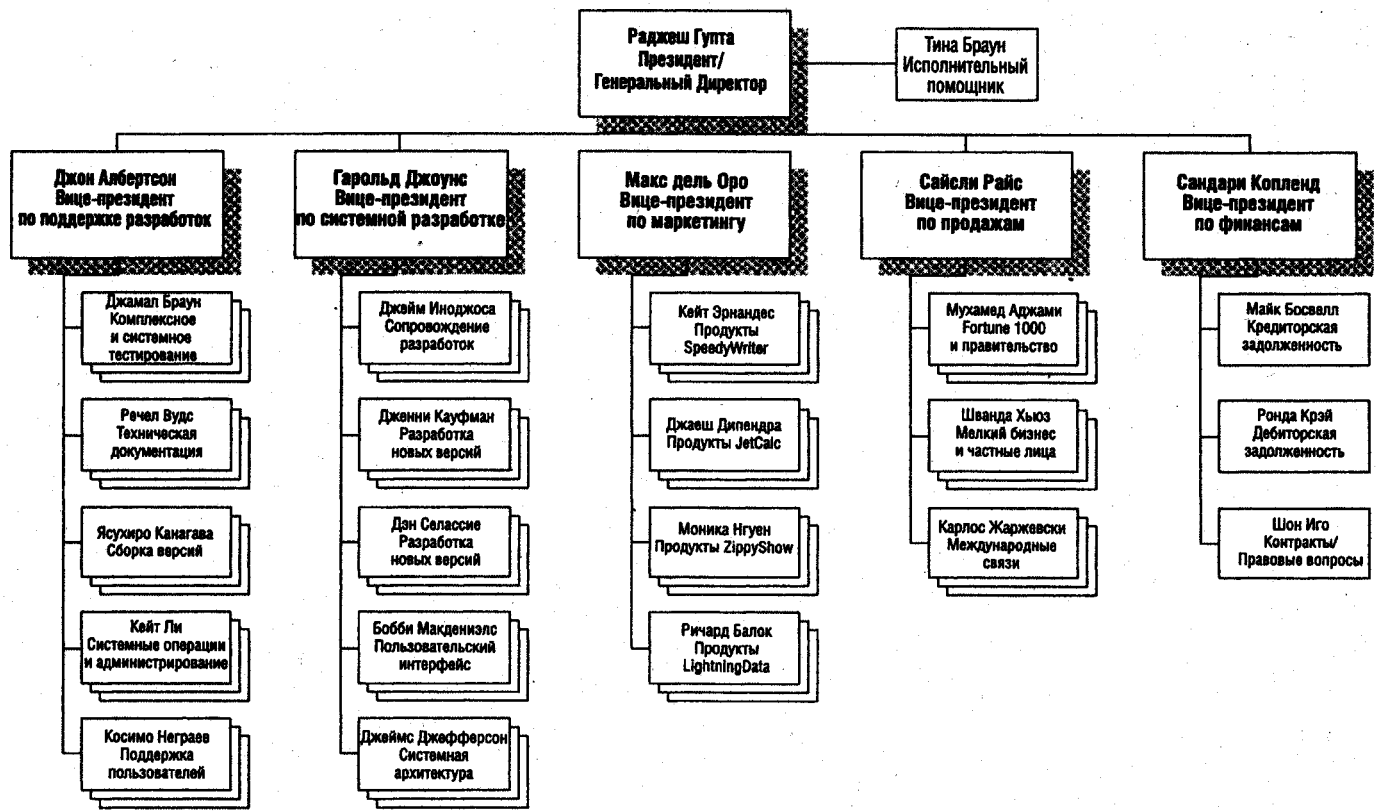


Рис. 1.2. Структура компании Software Cafeteria

но, пройти тестирование и быть готовыми к поставке. Такой подход потребует более серьезных затрат на тестирование, чем в предыдущих версиях, выполненных по каскадной модели. Однако, утверждала она, инкрементный подход позволит компании улучшить управление как качеством, так и планом выпуска версий Суматры. Ранние версии, разработанные каскадным методом, сказала она, вели к бурному росту ограничений бюджета и графика, что в конечном счете выливалось в выбор, по ее словам, «между немедленной поставкой разрушающего репутацию хлама и поставкой удовлетворительного для клиентов продукта с опозданием на три месяца». Она также пообещала не допустить, чтобы проект потонул в «позорной яме кодирования и исправления», что уже привело к скверной версии 2.5 с плохой репутацией и со всеми вытекающими последствиями в виде физической и моральной усталости и цинизма персонала. В заключение она сказала, что целевой датой выпуска первой инкрементной версии является 21 октября (т.е. примерно через семь недель) и что версия будет включать в себя лишь средства управления иерархическим хранилищем.

№ этапа	Этап	Выполнено?
1.	Понять процессы жизненного цикла, применяемые в вашей компании для разработки, сопровождения или поставки системы, включая любые планируемые или осуществляемые инициативы по совершенствованию процессов.	☑

После собрания Джамал обсудил с Дженни и Кейт некоторые детали проекта. Джамал воспринял с энтузиазмом возможность раннего начала тестирования и большие объемы работ по тестированию в проекте. Его, правда, тревожил вопрос, готово ли руководство предоставить ресурсы, необходимые для работы в рамках инкрементной модели жизненного цикла.

"Если я правильно вас понял, — сказал он, — я должен быть готов начать тестирование 21 октября, и далее тестирование будет продолжаться до конца марта. В ходе работы над версией 3.0 было запланировано 6 недель на тестирование, которые, естественно, растянулись на 12 недель, пока все было сделано".

"Да, — сказала Кейт, — и, хотя тестировщики проделали огромную работу, мы поставили продукт с очевидными проблемами в качестве. Твоя группа нашла около 1000 ошибок, из которых лишь 200 были исправлены. Кос и я до сих пор сталкиваемся с тем, что пользователи отказываются от той версии, — продолжала Кейт, ссылаясь на менеджера по поддержке пользователей Косимо Неграева. — Конечно, рекорд версии 2.5 был побит, но не намного".

"Хм, никогда снова не допущу этого", — поддержала Дженни, а Джамал утвердительно кивнул.

№ этапа	Этап	Выполнено?
2.	Изучить всю документацию, относящуюся к тестированию и качеству, данные, метрики и измерения, в том числе планы тестирования, базы обнаруженных дефектов, данные об ошибках, найденных при эксплуатации, системы тестов и т.п. Понять, кто разработал все эти артефакты, почему они это сделали, почему они это сделали таким образом и что привело к таким решениям и таким результатам в компании.	☑

"Джамал, ты знаком со мной уже какое-то время, и тебе известно, что я никогда не стремилась стать менеджером проекта, — добавила Дженни. — Я сказала руководству, что соглашусь на эту должность лишь при условии обязательного выполнения ряда требований. Одно из них — избавиться от старых моделей типа "кодируй и исправляй" и каскадной модели. Если у тебя будут проблемы с получением нужного объема ресурсов, какими бы они ни были, чтобы начать тестирование 21 октября и продолжать работу до конца марта, просто зайди ко мне в кабинет и скажи, что тебе нужно. Я буду решать проблемы".

Затем Кейт, Дженни и Джамал перешли к обсуждению модульного и компонентного тестирования, а также альфа- и бета-тестирования. Дженни рассказала о своих планах поручить каждому программисту подготовить автоматизированные тесты и проводить с их помощью тестирование собственных модулей по мере их разработки в своей тестовой среде в качестве частично формализованного (semiformal) модульного тестирования. Тесты для модульного тестирования, как и код, станут предметом перекрестного рецензирования (peer review). Регистрация кода в проектном репозитории будет производиться лишь при соблюдении следующих трех критериев:

1. Модульные тесты пройдены без ошибок.
2. Рецензирование кода завершено и утверждено.
3. Рецензирование тестов завершено и утверждено.

Подготовленные тесты становятся частью автоматизированного набора тестов для компонентного тестирования, они выполняются в качестве части автоматизированного комплекта при ночном прогоне[1].

Джамалу было по душе слышать все это, поскольку предложенный подход поможет ему решить техническую проблему: комплексное тестирование. Он предложил Дженни решение, при котором группа тестирования будет участвовать в рецензировании тестов для компонентного тестирования, поскольку тестировщики смогут потом использовать их в качестве основы для комплексного тестирования. Дженни высоко оценила идею Джамала. Они вдвоем обсудили проблемы ресурсов, относящиеся к этому виду тестирования, который обычно дает хорошие результаты, но может затягивать процесс разработки и требовать активного участия тестировщиков в тестировании на фазах разработки. Кейт рассказала Джамалу, что аналогично разработке предыдущих версий группа маркетинга планирует и будет проводить как альфа-, так и бета-тестирование. Альфа-тестирование начнется, как только будет готова первая интегральная сборка с полной реализацией первой итерации наращивания функциональности, прошедшей компонентное тестирование. Альфа-тестирование предполагает поставку продукта внутри компании, в том числе в про-

---

[1] Это не так уж сложно сделать. За дополнительной информацией можно обратиться к статье Рекса Блэка (Rex Black) и Грега Кубачковски (Greg Kubaczkowski) "*Mission Made Possible*", впервые опубликованной в журнале "*Software Testing and Quality Engineering*", Volume 4, Issue 4 (Jul/Aug 2002) и доступной в настоящее время на www.stickyminds.com и www.rexblackconsulting.com.

ектной команде Суматры, а также некоторым тщательно отобранным партнерам по бизнесу.

Бета-тестирование начнется после завершения первого цикла системного тестирования первой итерации наращивания функциональности, т.е. при удовлетворительных результатах тестирования. Кейт сказала Джамалу, что она должна работать в тесном контакте с ним, чтобы убедиться, что наиболее существенные для бета-тестирования свойства и функции тщательно протестированы в ходе первого цикла системного тестирования. Джамал ответил, что это отличная мысль, но Дженни также должна проверить, что соответствующие свойства продукта уже реализованы группой разработки и подвергались тщательному компонентному тестированию. Дженни согласилась, добавив, что она понимает, что Джамал не может повысить качество некачественного продукта за счет тестирования.

№ этапа	Этап	Выполнено?
3.	Обсудить с другими участниками проекта, в каких действиях тестирования они участвуют и что они собираются делать с учетом того, что вы тоже участвуете в работе группы тестирования.	☑

К этому моменту Джамал полностью представлял контекст своей работы, по крайней мере, так, как видят его Кейт и Дженни. Поскольку они являются ключевыми заинтересованными лицами, крайне важно, чтобы они понимали, каким образом группа Джамала будет встроена в проект. Однако Джамалу известно, что понимание текущей ситуации является непрерывным процессом. Лучшие тестировщики не прекращают следить за внешними проявлениями того, что означает непонимание или несогласие коллег с тем, что делает группа тестирования. Уместность тех или иных действий существенна для доверия, а доверие в свою очередь, как вам скажет любой газетчик, имеет огромное значение для тех, кто поставляет жизненно важную информацию. Хотя Джамал работал в компании уже больше года, он планировал встречи с руководством и коллегами для уточнения их ожиданий и собирался повторять такие встречи в ходе всего проекта.

№ этапа	Этап	Выполнено?
4.	Определить заинтересованных лиц своего уровня в процессе тестирования. Провести с ними переговоры для понимания состояния взаимоотношений с теми, кто сейчас проводит тестирование, их ожиданий от процесса тестирования, предыдущих разочарований от процесса тестирования и конфликтов с теми, кто участвует в тестировании и т.п.	☑
5.	Уточнить с непосредственными руководителями и старшими менеджерами их ожидания от процесса тестирования и от вашего участия в тестировании и в оценке качества.	☑

Место тестирования в моделях жизненного цикла разработки

Ранее упоминались две методологии разработки: каскадная модель (и ее вариация — V-модель) и инкрементная модель. В рассматриваемом примере Дженни предлагает инкрементную модель как наиболее подходящий выбор для проекта SpeedyWriter.

Обсуждение выбора наилучшей модели для данного проекта находится за рамками этой книги. Каждая из моделей жизненного цикла спроектирована для наиболее успешной работы с проектными рисками определенного вида[1]. Кроме того, модели жизненного цикла, как правило, связывают с определенными видами разработки, сопровождения и поставки систем. Например, на своем web-сайте компания Rational продвигает процесс разработки, известный как Rational Unified Process, в качест-

Жизненный цикл каскадной модели разработки системы (Waterfall System Development Lifecycle)

Модель разработки, в которой группа разработки проходит серию последовательных фаз в процессе создания системы. Сначала группа определяет требования, затем проектирует систему и наконец реализует ее в коде. Как только код системы готов, система проходит через последовательность фаз тестирования, начиная с модульного или компонентного тестирования, затем комплексное тестирование и наконец системное тестирование. (В случае контрактной разработки или разработки для внутреннего применения часто после системного тестирования проводятся приемо-сдаточные испытания.) Вариации этого подхода допускают наложение фаз, не требуя завершения каждой из фаз перед началом следующей.

V-модель (V-Model)

Вариация каскадной модели, в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования — вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из фаз разработки влияют на развитие системы тестирования на каждой из фаз тестирования. Модель базируется на том, что приемо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование — на требованиях и архитектуре, комплексное тестирование — на требованиях, архитектуре и интерфейсах, а компонентное тестирование — на требованиях, архитектуре, интерфейсах и алгоритмах. На рис. 1.3 дано графическое представление этой модели разработки, где система может представлять собой программу, аппаратное решение, аппаратно-программную систему или семейство или систему систем.

---

[1] Обзор моделей жизненных циклов можно найти в главе 2 книги Роджера Прессмана (Roger Pressman) "*Software Engineering, Fourth Edition*" и в главе 2 книги Стефана Кана (Stephen Kan) "*Metrics and Models in Software Quality Engineering, Second Edition*".

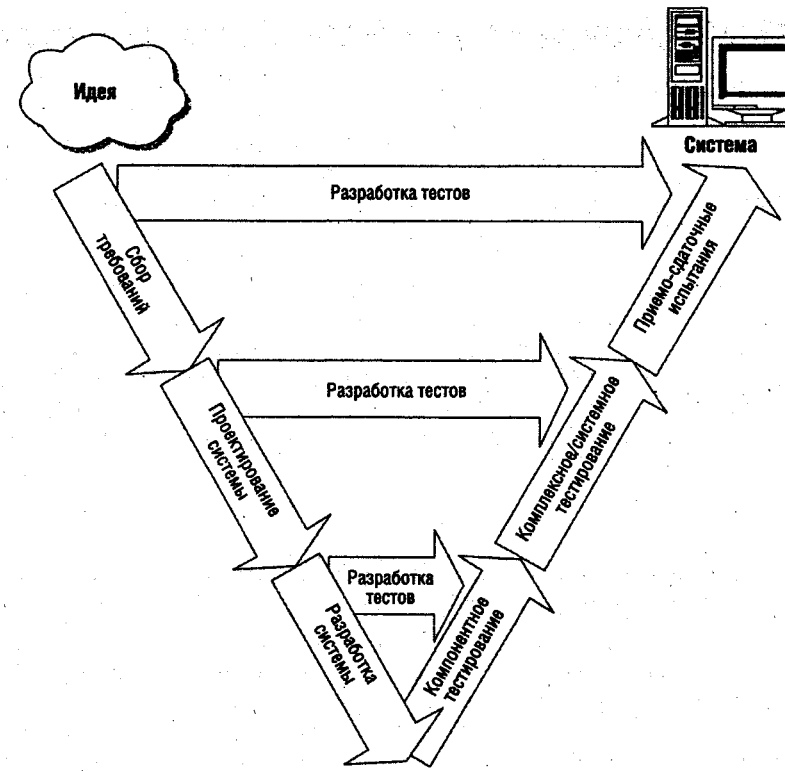


Рис. 1.3. V-модель жизненного цикла каскадной модели разработки системы

ве наиболее подходящего для проектов разработки с использованием объектно-ориентированных технологий.

Как и предусматривает слово "модель", описываемые жизненные циклы являются абстракцией и упрощенным представлением определенного круга людей о том, как делаются успешные проекты. Аналогичным образом я описываю в этой книге процессы, которые моделируют мои представления о том, как необходимо вести успешные подпроекты тестирования.

Сколько всего существует моделей жизненного цикла? Поверхностный поиск в Интернете выдает ссылки на десятки различных методологий для объектно-ориентированного программирования. Некоторые из этих методологий говорят об особых жизненных циклах, другие более легковесны, с менее строгими ограничениями на порядок фаз и сами фазы, с большим акцентом на людей, нежели на процессы[1].

[1] На www.cetus-links.org представлен целый набор ссылок, связанных с объектно-ориентированным программированием, включая ссылки на методологии. Также существуют различные web-сайты, описывающие легкие технологии, рассказывающие о так называемых быстрых методах, в частности об экстремальном программировании (Extreme Programming). См., например, www.martinfowler.com/articles/newMetodology.html.

Насколько точно эти модели описывают то, что реально имеет место? Приведем высказывание, на авторство которого претендуют и Э.Деминг, и Джордж Бокс (George Box): "Все модели неверны, некоторые модели полезны". До определенной степени распространение разных моделей указывает на различие взглядов и полемику, существующие вокруг способов создания систем. Чем более популярны жизненный цикл и методология, тем больший огонь критики они на себя вызывают. Например, некоторые специалисты писали, насколько ужасна V-модель. Тем не менее она по-прежнему широко применяется. Это не просто результат инерционности или лености мышления. Вариантом, интуитивно понятным, широко известным и более предпочтительным, нежели хаос, часто является V-модель. Однако любая методология и любой жизненный цикл могут подходить для конкретно взятой ситуации или нет. В одном случае происходит одно, в другом – другое. Поскольку каждый проект по-своему уникален, менеджеры и другие участники должны подстраивать выбранный жизненный цикл под свои, специфические потребности.

Эти модели являются не только приближением того, что *должно* быть, но и того, что *реально* происходит, когда их пытаются применять в проектах. Иногда подгонка модели хорошо продумана, иногда не столь точна и аккуратна. Я был свидетелем проектов, в которых поначалу применялся тот или иной жизненный цикл, но затем они скатывались в яму кодирования и правки, причем наиболее часто на фазе тестирования, когда проект выходил за пределы исходного графика и бюджета.

Кодируй и исправляй (Code and Fix)

Подход к разработке, исключающий большинство действий по планированию, сбору требований и проектированию, начинающийся непосредственно с кодирования системы с последующим исправлением обнаруженных ошибок. Обнаружение ошибок ведется посредством формального тестирования независимой группой тестировщиков, либо с помощью неформального (ad hoc) тестирования, либо только при отладке. Данная модель может быть удобна для небольших, несложных проектов разработки или поддержки с небольшим уровнем рисков. Модель не подходит для больших, сложных проектов и проектов с высоким уровнем рисков.

Что означает для тестировщика распространение различных и несовершенных моделей? Есть сильный соблазн проигнорировать эту "вавилонскую башню" методологий как бессмысленное отвлечение внимания. Это особенно верно, поскольку некоторые из этих моделей (например, V-модель) разработаны специально для тестирования, другие только учитывают его, а третьи вовсе не принимают его во внимание. К примеру, экстремальное программирование много внимания уделяет надежному модульному и компонентному тестированию силами программистов, но практически не затрагивает комплексное и системное тестирование. По-

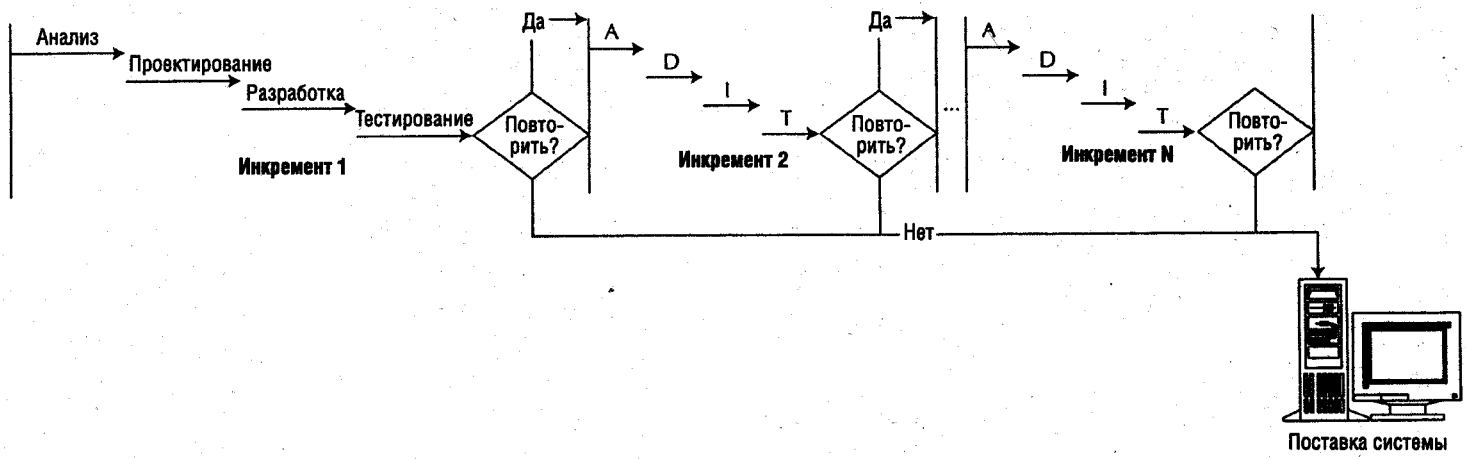


Рис. 1.4. Жизненный цикл инкрементной разработки системы

скольку различные фазы тестирования ориентированы на поиск различных категорий дефектов, тестировщики вынуждены заполнять пробелы, если они работают над проектом, который следует модели экстремального программирования.

Существуют три важных для тестировщиков обстоятельства, связанных с этой темой. Во-первых, в рамках контекста проекта необходимо сохранять гибкость при встраивании тестирования в выбранный жизненный цикл или методологию. Нужно понять, каким образом адаптировать и предоставить значимые услуги и информацию, что бы ни происходило. Также необходимо строить гибкие планы на случай, если выбранная методология начнет деградировать до безумия "кодирования и исправления", чтобы продолжать поставлять значимые услуги и информацию даже в условиях полного хаоса. На самом деле, в этой книге я описываю идеи, которые могут оказаться полезными независимо от того, работаете ли вы в строго следующем процессам проекте или внутри циклона с беспорядочно меняющимся кодом.

Во-вторых, встраивание процесса тестирования в объемлющий жизненный цикл влечет за собой необходимость понимания последовательности и взаимозависимости действий. Под выстраиванием последовательности действий понимается, насколько отдельные части процесса тестирования вписываются в жизненный цикл системы в терминах временной протяженности. В частности, V-модель устроена таким образом, что проектирование и разработка системы тестов должны начинаться как можно раньше. Тестировщики, работающие по V-модели, разрабатывают тестовые сценарии, тестовые данные, тестовые скрипты, инструменты тестирования, тестовую среду и другие компоненты системы тестов во время фаз сбора требований, проектирования архитектуры и реализации кода. Раннее начало действий по проектированию и разработке системы тестов нацелено на своевременное проведение приемо-сдаточных испытаний и системного тестирования, включая частные задачи проектирования и разработки для фаз комплексного и компонентного тестирования. Не все модели жизненных циклов столь явно описывают последовательность действий, однако в каждом проекте необходимо выбрать правильное время для начала важнейших процессов тестирования. На выбор времени оказывают влияние взаимозависимости. Например, если мы разрабатываем тесты на основе требований, мы не можем начать это делать, пока у нас нет требований.

В-третьих, в рамках более широкого сообщества системных разработчиков тестировщики обязаны заботиться о росте своего участия в непрекращающихся дискуссиях о жизненных циклах и методологиях. Процессы весьма значимы, а тестирование — важная часть процесса системной разработки. Вопросы выстраивания последовательности и взаимозависимостей действий оказывают серьезное виляние на успех подпроектов тестирования. Мы должны помочь людям, которые говорят о процессах, понять важнейшие процессы тестирования и то, как они соотносятся с другими активностями разработки. Еще одна задача этой книги — стать частью этой дискуссии.

Организация тестировщиков и тестирования

Помимо непрекращающихся дебатов по поводу методологий и функционального контекста тестирования в рамках различных моделей жизненного цикла, не утихают споры вокруг проблем, связанных с организацией тестировщиков и тестирования. Тестирование может быть организовано в рамках группы разработки, внутри проекта, в виде поставок услуг проекту независимой группой тестировщиков, некоторым образом внутри компании, приобретающей систему, или в виде комбинации описанных подходов. Группа тестирования, если она отделена от группы разработки, может подчиняться менеджеру разработки, менеджеру проекта, какому-либо из линейных менеджеров, исполнительному менеджеру либо одному из высших руководителей.

Я предпочитаю руководить отдельной группой тестирования или работать в такой группе. Мне нравятся независимые группы тестирования, которые не готовят отчеты для менеджеров, в качестве стимула выбирающих счетчик найденных дефектов и тщательную проверку качества. Однако мне также нравятся группы тестирования, которые встроены в проектную команду с целью включения в работу над проектом с его ранних фаз. Я люблю наблюдать за такими независимыми, но интегрированными группами тестирования, сосредоточившими усилия на наиболее ответственных фазах тестирования — комплексном и системном тестировании — таким образом, что они дополняют модульное и компонентное тестирование, которые в свою очередь отдаются на откуп группе разработчиков. Я также приветствую формальное и неформальное тестирование, выполняемое пользователями или замещающими их лицами, в частности альфа- и бета-тестирование для продуктов массового спроса и приемо-сдаточные испытания в мире технологий и систем управления информацией. Если услуги тестирования требуются сразу нескольким проектам, лучше, если единая группа тестирования будет предоставлять такие услуги всем проектам[1]. Почему я рекомендую этот подход?

Для оптимального поиска и устранения дефектов необходимо подвергнуть систему структурному и функциональному тестированию, а также тестированию в реальных условиях. Эти три стиля тестирования позволяют обнаружить различные типы ошибок и предоставить различный материал для отладки. Например, ошибки, связанные с неинициализированным указателем, могут иметь различные проявления при проведении функционального тестирования, часто ставя в тупик и тестировщика, и программиста. Однако существуют простые методы структурного тестирования (например, препроцессор lint в UNIX/C), позволяющие обнаружить переменные, используемые до их инициализации.

В настоящее время структурное тестирование базируется в основном на навыках программирования и другой технической подготовке, а функциональное тестирование часто требует дополнительно специализированных навыков тестирования и знания предметной области (см. главы 8 и 9). Примерами видов тестирования, требующих специализированных

---

[1] См. главу 9 моей книги "*Managing the Testing Process, Second Edition*".

Независимая группа тестирования (Independent Test Team)

Группа людей, отличная от разработчиков системы, предназначенная для тестирования; в сферу ее ответственности входят эффективная проверка и своевременное информирование о качестве тестируемой системы. Обычно я рассматриваю эту группу как поставщика услуг для проектной команды.

Поведенческое тестирование, тестирование методами «черного ящика» (Behavioral Tests, Black-Box Tests)

Тестирование на основе внешнего поведения системы, часто апеллирующее к требованиям и спецификациям архитектуры высокого уровня.

навыков, являются тестирование безопасности и производительности, в которых обычный программист не является специалистом. Что же касается примера предметной области, то один из моих заказчиков работает над системой геологического моделирования для поиска и разработки месторождений нефти. Чтобы тестировать функциональность и последовательность действий, необходимо быть геологом, геофизиком или специалистом из смежной области, но не программистом.

Я заметил, что проекты, в которых мне приходится участвовать, идут более гладко, если программисты проводят модульное и компонентное тестирование своего кода. С развитием таких подходов, как экстремальное программирование, такая позиция становится менее спорной. Выполнение модульных и компонентных тестов требует и навыков структурного тестирования, и умения программировать. Логично, что люди, ответственные за выполнение таких тестов, являются членами команды программистов. Однако многие программисты (и я в том числе) испытывают трудности при тестировании собственного кода. Удачной практикой является внедрение процесса разработки, предусматривающего модульное тестирование, в ходе которого программисты ищут ошибки в своем коде, и компонентное тестирование, когда программисты проверяют код, написанный коллегами. (Иногда такой способ называется "обменом кода".) Вариации этого подхода используют идеи парного программирования и экспертной оценки автоматизированных тестовых заглушек компонентов или тестовых драйверов[1].

Однако, поскольку для функционального тестирования требуются специальные навыки, объединять людей, на которых возложены эти обязанности, в независимую группу тестирования имеет смысл, как минимум, по трем причинам. Во-первых, когда есть двое и более примерно одинаково подготовленных людей, это повышает синергетику группы. Например, тестировщики могут рецензировать тестовые сценарии, ре-

[1] Более подробно о структурном тестировании см. Брайан Марик (Brian Marick) "*Craft of Software Testing*", Роберт Биндер (Robert Binder) "*Testing Object-Oriented Software*" и Борис Бейзер (Boris Beizer) "*Software Testing Techniques*".

Структурное тестирование, тестирование методами "белого ящика" (Structural Tests, White-Box Tests)

Тестирование, основанное на внутренних особенностях работы системы, обычно апеллирующее к деталям архитектуры, логике и данным.

зультаты тестирования, отчеты об ошибках, подготовленные коллегами. Во-вторых, объединение в группу тестирования способствует укреплению боевого духа, что важно для качественного тестирования, поскольку во многих случаях приходится сообщать плохие новости. В-третьих, наличие группы тестирования дает возможность для специализации, совершенствования навыков и карьерного роста. Взаимодействие всех этих факторов способствует повышению производительности тестировщиков и укреплению доверия к результатам, получаемым группой тестирования.

Группа тестирования, работающая одновременно в нескольких проектах, предоставляет еще более широкие возможности для специализации и карьерного роста. Я работал в одной группе тестирования, поставлявшей услуги тестирования разнообразным проектам по разработке систем для персональных компьютеров. В нашей группе были отдельные подразделения, специализировавшиеся на тестировании сетей, клиентов Windows, серверов Windows, на тестировании UNIX-систем, тестировании приложений и т.п. В этих подразделениях работали сильные специалисты-тестировщики, как того требовали наши проекты.

Я отдаю предпочтение группам тестирования, возглавляемым компетентным, опытным тестировщиком, обладающим навыками управления. В этом случае тест-менеджер выступает как в роли технического лидера, так и в роли руководителя группы. Надо сказать, что лучшие менеджеры групп разработки в моей практике становились и хорошими программистами, и хорошими менеджерами. Достигнув определенного уровня в иерархии руководства, менеджеру действительно нет необходимости вникать в детали работы подчиненных. Однако, если менеджер осуществляет непосредственное руководство сотрудниками, которые собственно разрабатывают или тестируют систему, комбинация технической квалификации и умения руководить в одном лице срабатывает великолепно.

Руководство группой тестирования является определяющим, поскольку функциональное тестирование — это сложный подпроект в рамках всего проекта в том случае, если он осуществляется независимой группой тестирования. Между проектом и подпроектом тестирования существует множество взаимосвязей и взаимозависимостей. Кроме того, постоянные изменения в проекте должны находить как очевидные, так и едва заметные отражения в организации тестирования. Тест-менеджер выявляет эти зависимости и управляет изменениями в них. Он также должен обладать хорошими навыками управления персоналом, в частности для подбора людей и расширения группы тестирования. Наконец, тест-менеджер должен быть защитником интересов группы тестирования и уметь договариваться по насущным вопросам, связанным с тестированием.

Независимая группа тестирования должна получить необходимую часть ресурсов проекта. Независимая группа тестирования должна уметь беспристрастно информировать о полученных результатах, как хороших, так и плохих. Таким образом, структура управления не должна создавать у тестировщиков неправильных стимулов в их работе. Представим, к примеру, вполне обычную ситуацию: менеджер разработки получает премию в случае поставки системы с определенным набором функций в срок и в рамках бюджета. Что произойдет, если тест-менеджер сообщит менеджеру разработки, что при данных свойствах будущей системы, параметрах бюджета и сроках в системе останется много ошибок? Это вполне обычная ситуация, порождающая напряженность в отношениях тест-менеджера и менеджера разработки. Что будет, если руководитель менеджера разработки обратится к тест-менеджеру с просьбой оценить качество системы и дать прогноз, когда ее качество будет пригодно для поставки? Тест-менеджер поставлен перед нежелательным конфликтом интересов. Он может либо сообщить правду и указать на ошибки руководителя, либо приукрасить ситуацию и существенно снизить ценность группы тестирования для компании.

Я наслышан от коллег и от участников моего семинара по управлению тестированием о том, что компании могут разрешить эти политические проблемы посредством сочетания организационной структуры компании и интересов каждого из участников проекта с поставкой продукта, в котором соблюдается баланс свойств, сроков, бюджета и качества. В тех компаниях, в которых принята система вознаграждения менеджеров проектов и менеджеров разработки исключительно на основе сроков и бюджета, позиция тест-менеджера в организационной структуре на одном уровне с менеджером проекта и/или менеджером разработки может восстановить баланс между качеством, сроками, бюджетом и функциональностью.

Теперь вам известны мои предпочтения, однако здесь существует множество вариантов. Один из удачных вариантов показан в нашем примере. Тест-менеджер Джамал Браун целиком отвечает за поведенческое тестирование различных продуктов OfficeArrow на фазах комплексного и системного тестирования. Джамал отчитывается перед менеджером группы поддержки разработок Джоном Албертсоном, который занимает в структуре компании позицию на одном уровне с Гарольдом Джоунзом, менеджером системных разработок. Они оба подчиняются Раджешу Гупте, генеральному директору компании. Джамал занимает то же положение в компании, что и Дженни, его коллега, менеджер разработки.

Фаза тестирования, уровень тестирования (Test Phase, Test Level)

Выделяемая совокупность действий тестирования, адресованная фиксированной группе рисков качества, например: компонентное тестирование, комплексное тестирование, системное тестирование, приемо-сдаточные испытания. (Я предпочитаю использовать слово "фаза", тогда как другие говорят "уровень".)

При другом способе организации работ в случае тестирования нескольких проектов одновременно в группе, предоставляющей услуги тестирования, выделяется по одному старшему тестировщику, выступающему в роли эксперта по предметной области, на каждый проект от начала до конца, либо по тестировщику-консультанту, отвечающему за все задачи тестирования для данной системы. Другие сотрудники группы тестирования сосредотачиваются на предоставлении тестовых услуг в нескольких проектах одновременно. Тестировщик-консультант, который является специалистом как в структурном, так в поведенческом тестировании, выступает в роли технического эксперта по всем вопросам тестирования с первого дня проекта разработки системы. В совместной работе с программистами тестировщик-консультант содействует качественному проведению модульного, компонентного и комплексного тестирования, подчиняется по техническим вопросам менеджеру разработки и отчитывается перед ним об этих активностях, оставаясь организационно сотрудником группы тестирования. Тест-менеджер решает совместно с тестировщиком-консультантом и другими тестировщиками вопросы планирования тестирования, разработки тестов, построения тестовой среды для отдельной фазы системного тестирования. С началом фазы системного тестирования тестировщик-консультант возвращается в группу тестирования. В этот период все тестировщики являются ценными ресурсами для ведущего тестировщика и тест-менеджера, организующих системное тестирование. Тестировщик-консультант собрал огромный экспертный материал о системе, хорошо представляет ее сильные и слабые стороны и теперь консультирует группу тестирования. На рис. 1.5 показана структура группы, включая двух тестировщиков-консультантов (один работает в проекте A, второй – в проекте B), которые временно приписаны к соответствующим проектным группам. (Я упростил рисунок, удалив оттуда других возможных участников проектов разработки, не входящих в группы разработки и тестирования.)

На рис. 1.2 и 1.5 показаны только две возможности из большого списка хорошо зарекомендовавших себя эффективных вариантов организационного контекста, в рамках которых успешно проводится тестирование. Рекомендую исследовать организационную структуру различных проектов до того, как вы сделаете свой выбор. Мне довелось общаться со многими состоявшимися тестировщиками, работавшими с самыми разнообразными подходами на базе двух моделей, представленных мной[1].

Прежде чем перейти к другой теме, хотелось бы поговорить немного об одном способе организации группы тестирования, который я считаю ошибочным. Некоторые выступают за отсутствие какой бы то ни было группы тестирования, считая, что тестировщики могут работать вместе с программистами в группах разработки. Аргументы в защиту этого типа организации тестирования сводятся к положительному эффекту интег-

---

[1] Подробное обсуждение организационных структур, в рамках которых проводится тестирование, а также их преимуществ и недостатков можно найти в главе 13 книги Эдда Кита (Ed Kit) "*Software Testing in the Real World*". Брайан Марик также затрагивает организационную структуру и другие вопросы контекста тестирования в статье "Classic Testing Mistakes". Ее можно найти на сайте www.testing.com\writings.html.

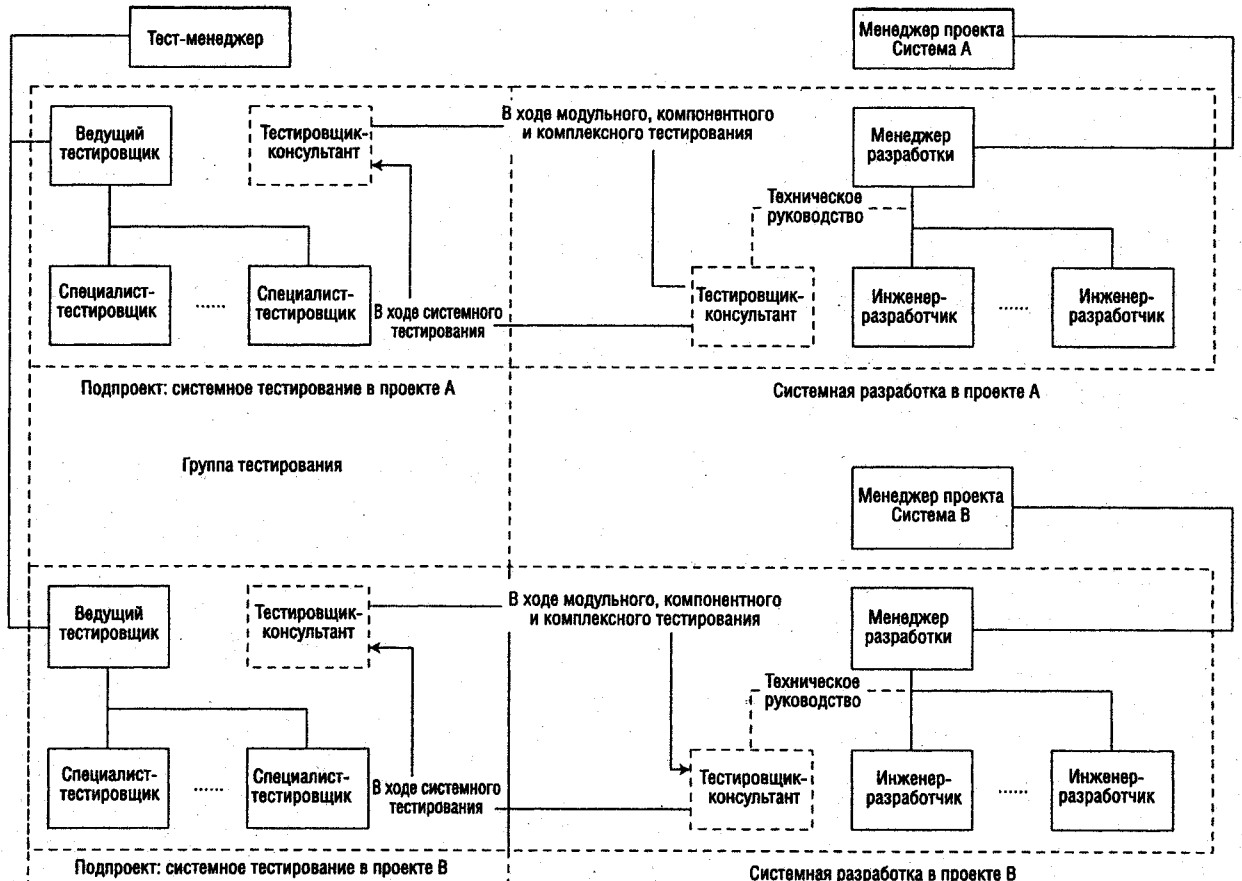


Рис. 1.5. Способ организации группы по предоставлению услуг тестирования нескольким проектам одновременно

рирования тестировщиков в группу разработки, что уравнивает в правах тестирование и разработку, позволяет отыскать больше ошибок и помогает осуществлению отладки.

Как я уже отмечал ранее, так могут быть организованы некоторые виды тестирования. Мне нравится, если структурное тестирование организовано как часть модульного и компонентного тестирования внутри группы разработки непосредственно с начала кодирования. Однако я против неявного предположения, что тестировщики, которые подчиняются менеджерам разработки, обязательно превосходят тестировщиков, подчиняющихся независимым тест-менеджерам, в показателях эффективности и качества тестирования, отчетов об ошибках, в проценте исправленных ошибок и в длительности цикла существования ошибки. В последующих главах я дам соответствующие объяснения.

Не могу не отметить, что вакансии тестировщиков в группах разработки и сопровождения часто предполагают роли младших программистов. Младшему персоналу обычно меньше доверяют, нежели более опытным сотрудникам. Ранее я писал, что основная ценность тестирования в своевременном информировании. Таким образом, тестировщики выполняют свою миссию, если люди прислушиваются к ним, а обычно более внимательно прислушиваются к тем людям, кому больше доверяют.

Идея назначения тестировщиками младших программистов, разбросанных по компании, имеет, по крайней мере, два отрицательных последствия для процесса тестирования. Первое касается квалификации: сотрудники, назначенные на роли тестировщиков, вероятно, обладают хорошими техническими навыками, но вряд ли им много известно о предметной области и о тестировании. К тому же, поскольку карьерные планы временных тестировщиков связаны с ролями программистов, они, по всей вероятности, будут стремиться улучшить техническую подготовку и знания предметной области, но лишь единицы озаботятся овладением навыками тестирования. В области тестирования они будут слабы, хотя устранение именно этих слабостей наиболее важно для их текущих обязанностей. Например, если перед младшими программистами встанет выбор между обучением структурному и поведенческому тестированию или обучением свежим методологиям программирования, вероятнее всего, они сделают выбор в пользу последнего, заботясь о своих долгосрочных карьерных устремлениях. Такие рациональные решения младших программистов снижают вероятность качественного и эффективного проведения тестирования.

Второе отрицательное последствие связано с вопросами управления. В тех случаях, когда старшие или высшие руководители принимали решение ликвидировать независимую группу тестирования и ввести тестировщиков в группы разработки, такие решения, как правило, были связаны с неудачными проектными планами. Такие способы реорганизации группы тестирования как ответ на проблемы со сроками проекта являются признаком невысокой зрелости компании. В этом качестве предлагаемые изменения представляют собой бесполезную реакцию на просчеты в управлении, а не рациональную попытку оптимизировать процесс тести-

рования. Они не позволят достичь желаемого спасительного эффекта для неудачного проекта[1].

Некоторые люди, мнение которых я уважаю, говорили мне, что они видели, как подходы, связанные с включением тестировщиков в группы разработки, успешно работают. Насколько я понимаю, успех таких решений связан, прежде всего, с относительно небольшими и несложными проектами разработки или сопровождения. Не сомневаюсь, что информация о положительном опыте, полученная не из первых рук, верна, но я не могу припомнить ни одного успешного проекта тестирования, выполненного силами только младших программистов, работающих внутри большой группы разработчиков. Большинство из участников моего семинара разделяют мое мнение. Не проведя формального исследования, тестировщики могут только высказывать предположения, а не оперировать объективными данными в поддержку своих выводов. Тем не менее, согласно моим ненаучным наблюдениям, негативное влияние на успех работ тестировщиков внутри групп разработки оказывают мощные факторы.

Выразив свое мнение по спорной теме поиска наилучшего места для тестирования и тестировщиков в компании, я намереваюсь отложить ее в сторону до конца книги. Организационная структура оказывает влияние на ключевые процессы, особенно на процессы, в наибольшей степени увязанные с контекстом, например на подготовку отчетов о результатах тестирования. Даже тестировщики, работающие в одиночку в качестве единственного ресурса тестирования в проекте, вероятно, обнаружат, что большинство, если не все важнейшие процессы, обсуждаемые в этой книге, применимы к их контексту тестирования.

Измерения управления

Началом любой карьеры, связанной с управлением или техническим руководством, обычно является совместная работа с менеджерами того же уровня или более высокими руководителями. Исполнители в повседневной работе обычно контактируют со своими непосредственными начальниками, которые дают им прямые указания. В этом случае для непосредственного исполнителя управление имеет одно измерение: взаимодействие с непосредственным руководителем. Для старшего тестировщика или тест-менеджера, имеющего в качестве основной обязанности подготовку отчетов об ошибках и результатах тестирования для рабочей группы проекта, ситуация будет меняться. Такое изменение помогает увидеть различные измерения управления и их влияние на подготовку отчетов о результатах тестирования.

Ранее были упомянуты четыре наиболее ценных результата работы большинства групп тестирования:

1. Обнаружение ошибок, которые будут устранены, или даже предотвращение их внесения.
2. Обнаружение ошибок, которые не будут устранены, но о которых становится известно.

---

[1] В введении к книге Paulk и др. "*The Capability Maturity Model*" приводятся комментарии к ситуации в компании, которая избавляется от тестирования при первых признаках серьезных проблем со сроками или бюджетом.

3. Проведение тестов, которые снижают (потенциально затратные) риски.
4. Обеспечение проекта своевременной, точной и заслуживающей доверия информацией.

Хотя результаты, которые ожидает получить компания от группы тестирования, могут меняться, ключевым отличием между ними является то, насколько сложно структурировать работы по тестированию, чтобы получить информацию и передать ее проектной группе таким образом, чтобы эта информация была бы использована с наибольшей эффективностью. Вот почему, по моему мнению, многие группы тестирования редко выходят за пределы первой половины пункта 1, занимаясь обнаружением дефектов в текущей системе. В правильном организационном, технологическом и проектном контексте, если внедрить 12 ключевых процессов, представленных в этой книге, то удастся порождать информацию, позволяющую добиться максимальной отдачи от вложений в тестирование.

Порождение информации требует преимущественно управления "вниз" (managing inward). Под управлением "вниз" я понимаю выполнение всех действий тестирования внутри группы тестирования. Как ключевой процесс тестирования оно вовлекает то, о чем я говорил во введении как о преимущественно внутренних процессах.

Но порождение информации – это не все, что требуется. Возврат вложений происходит посредством эффективного обмена полученной информацией, причем эта информация должна быть привязана к проектному, организационному и технологическому контексту.

Эффективный обмен информацией, привязанной к соответствующему контексту, требует преимущественно управления "вниз" и "вне". Управление "вверх" предполагает обмен информацией с непосредственными руководителями, другими старшими менеджерами и высшими руководителями. Управление "вне" включает в себя обмен информацией с техническими руководителями и менеджерами разработки, конфигурационного управления, управления сборкой, продаж, финансовыми менеджерами, менеджерами операций и технической поддержки, бизнес-аналитиками и т.д. того же уровня, что и тест-менеджер. На рис. 1.6 показана упрощенная схема, реальные проекты значительно сложнее. Контекст вашей работы может состоять из другого множества вертикальных и горизонтальных связей.

Что касается должностей ведущего тестировщика и тест-менеджера, то эффективность его управления "вверх" и "вне" оказывает большое влияние на общую оценку его работы. Иными словами, руководители и коллеги оценивают компетентность тест-менеджера не только по тому, как он получает информацию, но и (в не меньшей степени) по тому, как он выбирает информацию для дальнейшей работы и как он обменивается этой информацией с ними.

Управление "вверх" и "вне" ставит разнообразные проблемы перед ведущим тестировщиком и тест-менеджером. Даже опытные руководители разработки не всегда понимают роль и значение тестирования. В настоящее время область тестирования программ включает в себя такой объем знаний, что не каждый опытный тестировщик может охватить все. Это делает для тестировщика задачи взаимодействия еще более сложными.

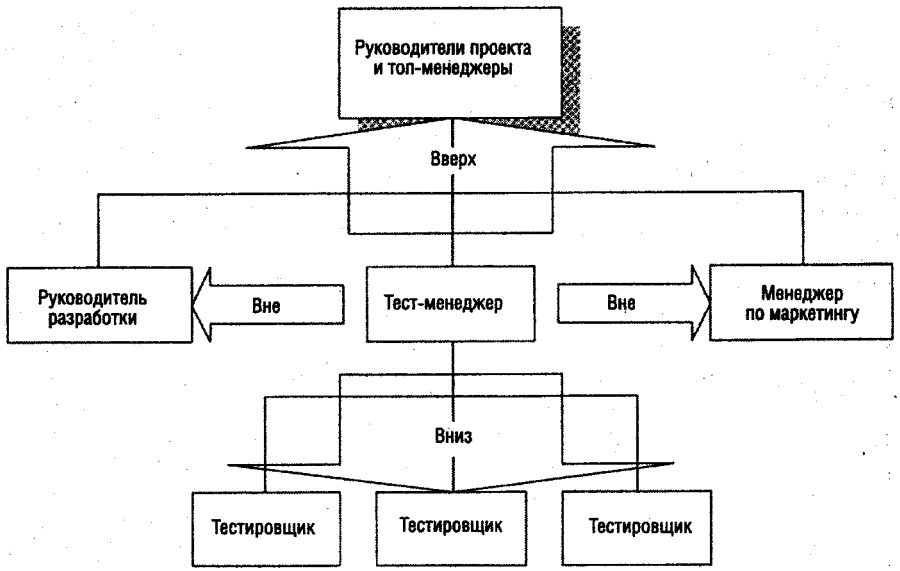


Рис. 1.6. Обмен информацией при управлении тестированием

Если тестирование проводится верно, возможно, наиболее важные результаты тестирования часто означают плохие новости для коллег и руководителей тест-менеджера. Компетентные тестировщики редко бывают в ситуации, когда они сообщают руководителям проекта: "Все тесты были выполнены в течение ночи, не обнаружено никаких значимых ошибок, можно поставлять продукт".

Проблемы управления "вне" и "вверх", в отличие от технических проблем, редко имеют простые решения. Однако здесь могут пригодиться полезные методики. По мере обсуждения критических процессов тестирования, в которых выявляются проблемы управления "вне" и "вверх", будут рассмотрены некоторые идеи решения таких проблем[1].

За рамками контекста процесса тестирования

В этой главе сначала было введено понятие процесса тестирования на самом низком уровне детализации, а затем мы определились с вопросами контекста, которые оказывают влияние на процесс тестирования. Сейчас я собираюсь перейти к другой теме, однако понимание и определение

[1] Части этого раздела впервые были опубликованы в статье "Effective Test Status Reporting" в журнале "*Software Testing and Quality Engineering*", Volume 2, Issue 2 (Mar/Apr 2000), доступной сейчас на сайтах www.stickyminds.com и www.rexblackconsulting.com.

контекста — это не единовременный акт. Меняются ожидания, к проектной группе присоединяются новые заинтересованные лица, у высшего руководства неожиданно возникает специфический интерес к проекту, в середине проекта могут произойти реорганизации. Благоразумные тестировщики бдительно следят за актуальностью контекста.

В этой главе мы уже прошли большой путь в определении контекста. Установление контекста тестирования нуждается в исчерпывающем обсуждении, поскольку процесс тестирования существует в самых разнообразных формах. Понимание контекста, в котором работают тестировщики, и внесение коррективов в случае, если что-то не функционирует в нем, — очень важный первый этап для тестировщиков, поскольку это позволяет группе тестирования перейти от практически бесконечного множества задач тестирования к более приемлемому набору в границах выбранной функциональной и организационной схемы. Задачи тестирования из последнего набора, вероятно, будут выполнены для создания новой стоимости. Следующие несколько глав будут посвящены разработке реалистичного, выполнимого плана с серьезной организационной поддержкой в рамках выстроенного контекста. Сначала я составляю план на основе представления о том, что *нужно* тестировать, а затем на основе того, что я реально *могу* протестировать. Каждый из этих дополняющих друг друга процессов планирования, а также все последующие процессы для успешного выполнения должны без помех встраиваться в контекст тестирования.

ГЛАВА 2

Основа успеха: анализ рисков качества

В функциональном и организационном контексте, выстроенном для процесса тестирования, *можно* выполнить огромное число тестов. Для любой по-настоящему сложной системы число тестов, которые можно было бы разработать и выполнить, потребовало бы человеко-годы, если не человеко-века. Так или иначе, необходимо сосредоточить внимание на тех тестовых условиях, которые *нельзя не* проверить, и не принимать к рассмотрению огромное количество несущественных условий, которые мы также могли бы проверять. Что же отличает области системы, которые мы *могли бы* тестировать, от тех областей, которые мы *должны* тестировать? Ответ – качество!

В предыдущей главе говорилось, что процесс тестирования приносит дивиденды благодаря предоставлению услуг и информации о рисках, угрожающих качеству системы. Чтобы предоставляемые услуги и информация имели ценность, необходимо, чтобы тестирование было нацелено на проверку функций, которые значимы для заказчиков и пользователей, функций, которые оказывают влияние на мнение пользователя о работе с системой. Тестирование несущественных частей системы вызывает ложную уверенность в правильной работе системы в случае успешного прохождения тестов, расходует впустую время разработчиков и тестировщиков на решение элементарных проблем в случае обнаружения ошибок. В то же время серьезные дефекты в непокрытых тестами областях системы остаются незамеченными, всплывая после выпуска версии перед пользователями системы. Чтобы провести тестирование, направленное на проверку наиболее критических рисков качества системы, заложим фундамент анализа рисков качества.

По аналогии рассмотрим коллег-программистов. Многие методологии разработки предписывают, чтобы сбор требований проводился перед проектированием и реализацией системы. Сбор требований позволяет бизнес-аналитикам, проектировщикам системы и программистам создавать системы, удовлетворяющие ожиданиям пользователей и заказчиков. Пропуск фазы сбора требований может привести к построению системы, которая ос-

Вид ошибки (Failure Mode)

Определенный способ возникновения или тип ошибки в системе, т.е. категория ошибок или сбоев с определенными отличительными общими признаками. Одному виду ошибки могут соответствовать ноль, одна и более отдельно взятых ошибок или неисправностей. При рассмотрении вида ошибки в любом варианте анализа рисков качества необходимо тщательно выверять степень детализации категорий видов ошибки, т.е. расширенные категории, состоящие из большого числа ошибок, или узкие категории, состоящие лишь из нескольких ошибок.

Риск качества (Quality Risk)

Потенциальный вид ошибки, способ поведения системы, при котором она, вероятно, не соответствует обоснованным ожиданиям качества системы, имеющимся у пользователя или заказчика. Заметим, что риск качества — это потенциальный, а не обязательный результат с вероятностью больше нуля и меньше единицы.

Рассматривают категории рисков качества. Поскольку виды ошибок также разбиваются на категории, категории рисков качества — это категории категорий потенциальных дефектов.

нована на замечательных идеях и титанических усилиях рабочей группы проекта, но которая не будет удовлетворять требованиям заказчиков и пользователей. Точно так же анализ рисков качества позволяет понять, что означает наличие достаточного качества у системы, предназначенной для тестирования, и каким образом можно распознать, что это не так.

Процесс анализа рисков качества

Процесс анализа рисков качества может быть прямым, как показано в описании Процесса 3. Возможно, что в ходе проекта потребуется повторить процедуру анализа рисков. Подходящее время для этого — основные рубежи (milestones), в частности окончательное завершение разработки требований, архитектуры или кода.

Группа тестирования Джамала проводит анализ рисков качества

После разговора с менеджером продукта Кейт Эрнандес и менеджером разработки Дженни Кауфман тест-менеджер Джамал Браун, отвечающий за комплексное и системное тестирование, некоторое время провел за сбором информации о Суматре и подсистеме управления документооборотом. Подсистема управления документооборотом должна предоставить

Ключевые лица, заинтересованные в качестве и тестировании, ключевые заинтересованные лица (Key Quality and Testing Stakeholders, Key Stakeholders)

Сотрудники компании, непосредственно контактирующие с заказчиками, пользователями или участвующие в процессе тестирования, а также сотрудники, которые хорошо знакомы с техническими деталями системы.

пользователям возможность создавать, обновлять, организовывать поиск и совместно использовать большие множества документов. Основным целевым рыночным сегментом пользователей этой функциональности являются юридические компании. Частью их работы является создание, сохранение, снабжение ссылками больших объемов крупных документов, относящихся к делопроизводству. Подсистема документооборота должна взаимодействовать с системами управления иерархических хранилищ в режимах онлайн (на диске), почти онлайн (на оптическом дисководе с автоматической сменой дисков) и оффлайн (на ленте).

Процесс 3. Процесс анализа рисков качества

№ этапа	Этап	Выполнено?
1.	Определить ключевых лиц, заинтересованных в тестировании и качестве. Добиться их согласия на участие в работе по анализу рисков качества.	☐
2.	Сделать для ключевых заинтересованных лиц обзор существующих методик и методов анализа рисков качества. Если уместно, предложить методику для практического применения в проекте. Достичь консенсуса в выборе предлагаемых методов.	☐
3.	Собрать идеи ключевых заинтересованных лиц о рисках качества, о видах ошибок, связанных с этими рисками, о влиянии на качество этих проблем и о расстановке рисков по приоритетам. Определить рекомендуемые действия по снижению каждого риска.	☐
4.	Сообщать о любых ошибках, обнаруженных в других проектных документах в ходе анализа, в том числе о неудачных и пропущенных требованиях, проблемах, связанных с архитектурой и т.д.	☐
5.	Документировать риски качества в соответствии с выбранной методикой. Разослать документ ведущим заинтересованным лицам. При необходимости повторить шаги 3 — 5 для завершения анализа рисков качества, расстановки приоритетов и выработки рекомендуемых действий.	☐
6.	Поместить документацию с анализом рисков качества в проектную библиотеку или систему управления конфигурацией. Организовать контроль над изменениями документации.	☐

Затем Джамал переговорил с Джоном Албертсоном, вице-президентом департамента по поддержке разработок, которому непосредственно подчиняется группа тестирования Джамала. Джамал рассказал Джону, что познакомился с методикой анализа риска качества под названием "Анализ видов ошибок и их влияния" (Failure Mode and Effect Analysis, FMEA). Он также пообщался с сотрудниками, которые уже работали по этой методике. С учетом всего комплекса проблем, с которыми столкнулся Джамал в ходе работы над последними двумя версиями при поиске консенсуса при определении областей системы, покрываемых тестированием, он предположил, что инструментарий FMEA мог быть помочь в разрешении такого рода проблем. Джон поддержал предложение Джамала апробировать эту методику.

Далее Джамал отправил по электронной почте сообщение Кейт, Дженни, Бобби Макдениэльсону, менеджеру по вопросам удобства использования, Максу дель Оро, вице-президенту по маркетингу, и Косимо Неграеву, менеджеру по поддержке пользователей. Джамал сообщил им, что начинает анализ рисков качества для определения областей системы, нуждающихся в тестировании, и что он хотел бы использовать методику анализа видов ошибок и их влияния. Он предложил провести день за пределами офиса, чтобы определить риски качества и расставить их по приоритетам сразу же, как будет завершен первый черновой вариант спецификаций требований. В течение нескольких дней Джамал получил согласие всех участников, в результате ассистент Кейт по административным вопросам зарезервировала помещение под семинар в местном отеле за неделю до его проведения.

№ этапа	Этап	Выполнено?
1.	Определить ключевых лиц, заинтересованных в тестировании и качестве. Добиться их согласия на участие в работе по анализу рисков качества.	☑
2.	Сделать для ключевых заинтересованных лиц обзор существующих методик и методов анализа рисков качества. Если уместно, предложить методику для практического применения в проекте. Достичь консенсуса в выборе предлагаемых методов.	☑

К началу семинара по анализу видов ошибок и их влияния Кейт и Макс выпустили первую черновую версию спецификаций требований и было проведено ее обсуждение. Все участники предстоящего семинара присутствовали на совещании, где обсуждались требования, поэтому не было необходимости резервировать время на семинаре для обсуждения требований к продукту.

На семинаре Джамал познакомил присутствующих с методикой FMEA и с идеей мозгового штурма[1]. Он отметил, что в данном случае применение методики должно быть ограничено проблемами, которые могут возникнуть в Суматре, и призвал не рассматривать проблемы, которые, вероятно, еще остаются в текущей (3.0) версии продукта SpeedyWriter. Чтобы начать работу, Джамал обозначил несколько общих категорий

[1] Более подробно о FMEA см. Stamatis, "*Failure Mode and Effect Analysis*". Краткое изложение методики приводится в справочнике McDermott et al., "*The basics of FMEA*".

рисков качества, под которые могли подпадать отдельные виды ошибок. В таблице 2.1 приводится перечень, представленный Джамалом участника семинара.

Таблица 2.1. Общие категории рисков качества для продукта SpeedyWriter

Категория риска качества	Проблемы, соответствующие категории
Функциональность	Проблемы, в результате которых не работают конкретные функции
Нагрузка, производительность, объем	Проблемы обработки ожидаемых пиковых нагрузок при параллельной работе пользователей
Надежность, стабильность работы	Проблемы, при которых система слишком часто зависает или долго не восстанавливается
Перегрузки, обработка ошибок и восстановление	Проблемы, возникающие ввиду превышения допустимых пиковых нагрузок или из-за обработки недопустимых условий (например, побочный эффект от сознательно внесенных ошибок)
Обработка дат и времени	Ошибки в математических действиях с датами и временем, в их форматировании, в планируемых событиях и других операциях, зависящих от времени
Качество данных	Ошибки в обработке, хранении или извлечении данных
Производительность	Проблемы с временем завершения задач при ожидаемой загрузке
Локализация	Проблемы, связанные с локализацией продукта, в том числе в обработке страницы символов, языковой поддержке, в использовании грамматики, словаря и тезауруса, а также в сообщениях об ошибках и файлах помощи
Безопасность	Проблемы в защите системы и охраняемых данных от мошеннического и злонамеренного использования
Установка/перенос	Ошибки, которые препятствуют поставке системы
Документирование	Ошибки в руководствах по установке и работе с системой для пользователей и системных администраторов
Интерфейсы	Ошибки в интерфейсах между компонентами

Используя этот перечень в качестве затравки, Джамал вместе с участниками семинара приступил к мозговому штурму, выделяя те отдельные проблемы, которые подпадают под каждую из категорий рисков, и проверяя, нет ли пропущенных категорий. Макс обнаружил, что пропущена категория "удобство использования", а Косимо добавил конкурентные недостатки. После того как были предприняты усилия по устранению избыточной информации и комбинированию пересекающихся проблем, группа выделила около 80 отдельных проблем.

Имея перечень видов ошибок, теперь можно расставить риски, связанные с ними, по приоритетам и определить действия по смягчению критических рисков. Джамал пояснил, что для оценки рисков он использует упрощенную методику анализа видов ошибок и их влияния. Он попросил участников семинара для каждого вида ошибок рассмотреть три фактора: серьезность, приоритетность и вероятность — и проранжиро-

вать их по шкале от 1 (наибольший риск) до 5 (наименьший риск). **Серьезность** — значимость влияния ошибки для данного вида ошибок, принимает одно из возможных значений от 1 (наиболее разрушительная) до 5 (наименее разрушительная).

1. *Потеря данных.* Ошибки, относящиеся к этому виду ошибок, могут вызвать потерю пользовательских (конечного пользователя, оператора системы и т.п.) или системных данных.
2. *Потеря функциональности.* Ошибки, относящиеся к этому виду ошибок, могут заблокировать использование основной функциональности (могут включать в себя и нефункциональные проблемы, например связанные с производительностью, которые вызывают неприемлемые задержки в использовании функций).
3. *Потеря функциональности с наличием обходного пути.* Ошибки, относящиеся к этому виду, могут заблокировать использование основной функциональности, но для пользователя существует разумный обходной путь.
4. *Частичная потеря функциональности.* Ошибки, относящиеся к этому виду, могут заблокировать использование некоторой несущественной части функциональности.
5. *Косметическая ошибка.* Ошибки, относящиеся к этому виду, допускают нормальное функционирование системы, но со значительными недостатками (в особенности, в пользовательском интерфейсе или в способности системы реагировать на запросы пользователя).

Приоритетность — значимость исправления ошибки для этого вида ошибок либо из-за влияния на способность системы соответствовать требованиям пользователей, либо из-за проблем в логике проекта, несоответствия стандартам или инструкциям, либо в связи с другими соображениями, связанными с бизнесом, от 1 (наиболее важно исправить) до 5 (наименее важно исправить).

1. *Неотложно.* Ошибки, относящиеся к этому виду, могут потребовать немедленного исправления.
2. *Существенно.* Ошибки, относящиеся к этому виду, скорее всего, будут обязательно исправлены к моменту выпуска версии.
3. *Значимо.* Ошибки, относящиеся к этому виду, могут существенно снизить ценность системы для одного и более заказчиков или пользователей.
4. *Желательно.* Ошибки, относящиеся к этому виду, могут быть исправлены к выпуску текущей версии, если это будет возможно в рамках ограничений по функциональности, бюджету и срокам, в противном случае они будут исправлены в следующей версии.
5. *На усмотрение.* Ошибки, относящиеся к этому виду, будут исправлены в одной из будущих версий, если это позволит приоритетность.

Вероятность — возможность возникновения ошибки этого вида и всех ее проявлений, от 1 (наиболее вероятно) до 5 (наименее вероятно). На величину этого рейтинга влияют три фактора. Во-первых, для всех ошибок, включенных в этот вид, устанавливается, велика ли вероятность

того, что они могут появиться в системе, с точки зрения факторов технического риска: сложности и предыдущей истории дефекта. Во-вторых, велика ли вероятность того, что ошибки будут пропущены разработчиками (не включая тестировщиков)? И, в-третьих, велика ли вероятность того, что они будут проявляться в повседневном использовании, в текущих действиях пользователя или при инсталляции? Если программисты не внесут ошибок, вызывающих проблему, или в ходе рецензирования кода такие ошибки будут обнаружены, или пользователи не будут применять систему таким образом, что столкнутся с этими ошибками, тогда риск минимален. Если указанные ошибки существуют, при рецензировании кода они были пропущены, а пользователи столкнутся с ними, то риск велик.

1. *Очень вероятно.* Почти наверняка окажет влияние на работу всех пользователей, функций или инсталляционных программ.
2. *Вероятно.* Может оказать влияние на работу многих пользователей, функций или инсталляционных программ.
3. *Возможно.* Может повлиять на работу некоторых пользователей, функций или инсталляционных программ.
4. *Маловероятно.* Может оказать влияние на работу ограниченного числа пользователей, функций и инсталляционных программ.
5. *Очень маловероятно.* Практически не повлияет на работу большинства пользователей, функций или инсталляционных программ.

Произведение трех значений — серьезности, приоритетности и вероятности — дает приоритет риска (risk priority number, RPN), который может принимать значение от 1 (крайне опасно) до 125 (практически несущественно).

После установления приоритетов рисков участники семинара определили рекомендуемые действия по снижению рисков в ходе тестирования. В основном действия распределились по четырем уровням тестирования[1].

1. *Подробное тестирование* (extensive). Тестировщики должны сделать попытку в рамках проектных ограничений покрыть функции и свойства, относящиеся к этим рискам качества, как вширь, так и вглубь. При рассмотрении ошибок, относящихся к этим рискам, готовя отчет об ошибке, тестировщики должны потратить определенное время на воспроизведение и локализацию проблемы.
2. *Сбалансированное тестирование* (balanced). Тестировщики должны принимать во внимание ограничения бюджета и сроков при создании широкого, не обязательно глубокого, покрытия тестами функций и свойств, попадающих под эти риски качества. Обнаружив ошибки, связанные с этими рисками, тестировщики должны ограничивать время на воспроизведение и локализацию проблем, основываясь на оценке серьезности и приоритетности ошибки.

---

[1] Стив Сплайн (Steve Splain), один из рецензентов этой книги, указал на то, что можно также рассмотреть относительные затраты на тестирование в сравнении с другими мерами, которые можно было бы использовать для снижения данного вида риска. Мы также могли бы перепроектировать систему, застраховать ее, использовать рецензирование кода и другие подходы.

3. *Тестирование по возможности* (opportunity). Если в ходе проектирования, разработки или выполнения очередного теста тестировщики могут соразмерять деятельность по настройке, выполнению тестов, операциям по демонтажу системы или подготовке данных для проведения тестов, чтобы выполнить поверхностное тестирование некоторых или всех условий, относящихся к данному риску, то в этом случае они должны выполнить такой тест. Тестировщики должны ограничивать время на воспроизведение и локализацию ошибки на основе оценки ее серьезности и приоритетности.
4. *Информирование о наблюдаемых дефектах.* Тестировщики не должны проектировать или разрабатывать тесты для этих категорий рисков качества, но обязаны информировать об ошибках, относящихся к этим рискам, при их обнаружении. Тестировщики должны ограничивать время на воспроизведение и локализацию ошибки на основе оценки ее серьезности и приоритетности.

Рабочая группа проекта "Суматра" присвоила категории тестирования на основе значений приоритета рисков. Рискам со значениями от 1 до 8 присвоена категория "подробное тестирование". Рискам со значениями от 9 до 24 присвоена категория "сбалансированное тестирование". Рискам со значениями от 25 до 48 присвоена категория "тестирование по возможности". Рекомендовано не проводить тестирования для рисков с приоритетом выше 50, но информировать о таких ошибках. Исключения сделаны для некоторых рисков, относящихся к граничным условиям. Например, два вида рисков для качества данных получили приоритет, равный 8, но рабочая группа рекомендовала сбалансированное тестирование, поскольку, по ее мнению, подробное тестирование потребует слишком много времени для такого уровня риска[1].

№ этапа	Этап	Выполнено?
3.	Собрать идеи ключевых заинтересованных лиц о рисках качества, о видах ошибок, связанных с этими рисками, о влиянии на качество этих проблем и о расстановке рисков по приоритетам. Определить рекомендуемые действия по снижению каждого риска.	☑

В ходе анализа видов ошибок и их влияния рабочая группа проекта "Суматра" обнаружила две области, пропущенные в спецификациях требований. Во-первых, в требованиях не описано, каким образом система должна обрабатывать права доступа; вместо этого она полагается по умолчанию на контроль при входе в сеть. Участники семинара выразили общее мнение, что система должна также позволять пользователям изменять их учетные записи внутри самой системы документооборота. Во-

[1] В некоторых случаях я добавляю *поверхностное тестирование* как категорию между *сбалансированным тестированием* и *тестированием по возможности.* Поверхностное тестирование предполагает отбор небольшого числа рисков качества в определенной области системы. Можно было бы также добавить категорию "*не тестировать*» после категории "*информирование о наблюдаемых дефектах*». "Не тестировать» означает, что об ошибке, относящейся к данным рискам качества, не будет сообщено даже в случае ее обнаружения.

вторых, в спецификациях требований ничего не говорится о том, что должно происходить, если библиотекарь не загрузит объект — ленту, компакт-диск или оптический диск — в устройство. Эти находки служат в качестве примера раннего обнаружения дефектов в документе, описывающем требования.

№ этапа	Этап	Выполнено?
4.	Сообщать о любых ошибках, обнаруженных в других проектных документах в ходе анализа, в том числе о неудачных и пропущенных требованиях, проблемах, связанных с архитектурой и т.д.	☑

По итогам семинара Джамал подготовил документ с анализом видов ошибок и их влияния (см. рис. 2.1). Он разослал документ всем участникам семинара, а также Гарольду Джоунзу, вице-президенту по системному программированию. Адресаты высказали несколько замечаний, повышающих или понижающих значения серьезности, приоритетности и вероятности, но в целом они сочли документ приемлемым. Джамал внес заключительные изменения и добавил документ в проектный репозиторий.

№ этапа	Этап	Выполнено?
5.	Документировать риски качества в соответствии с выбранной методикой. Разослать документ ведущим заинтересованным лицам. При необходимости повторить шаги 3 — 5 для завершения анализа рисков качества, расстановки приоритетов и выработки рекомендуемых действий.	☑
6.	Поместить документацию с анализом рисков качества в проектную библиотеку или систему управления конфигурацией. Организовать контроль над изменениями документов.	☑

Завершив анализ рисков качества, Джамал достиг консенсуса с группой управления проектом "Суматра" относительно того, какие области *должны быть* протестированы. *Могут ли* теперь Джамал и его бравая команда тестировщиков нацелиться на проверку всех выделенных областей? Чтобы ответить на этот вопрос, Джамал приступил к оценке затрат и разработке плана. Этому посвящены несколько следующих глав.

Построение успешного процесса анализа рисков качества

Проведение мероприятий по анализу рисков качества, аналогичных тому, в котором участвовали Джамал и его коллеги, требует больших затрат времени и сил, однако это очень полезная работа. (Менее формальный анализ рисков может быть дешевле и проще, чем анализ видов ошибок и их влияния.) Правильно организованный процесс анализа рисков качества включает в себя следующие действия и дает следующие преимущества.

	A	B	C	D	E	F	G	H	I
1	**Failure Mode and Effect Analysis (Quality Risk Analysis) Form**								
3	**System Name: SpeedyWriter (with Document Management System add-on)**							**Model/Product: SpeedyWriter Release 3.1 (w/ DMS)**	
4	**Product Manager: Kate Hernandez**							**Target Release Date: 1Q2003**	
5	**Project Manager: Jenny Kaufman**							**Prepared By: Jamal Brown**	
6	**Other Stakeholders: Bobbi McDaniels (Dev), Max del Oro (Mkt), Cosimo Negraev (Sppt)**							**FMEA Date: September 11, 2002**	
7	**Test Manager: Jamal Brown**							**FMEA Rev Date: N/A**	
9	**Risk ID Number**	**Quality Risk Category**	**Failure Mode/Quality Risk/Effect**	**Severity**	**Priority**	**Likelihood**	**Risk Priority Number**	**Recommended Action**	**Who/ Which Phase (Unit, Component, Integration, System)?**
11	**1.000**	**Functionality**	Failures that cause specific features not to work						
12	**1.001**		Regression of existing SpeedyWriter features.	1	1	3	3	Rerun entire R3.0 test set.	Test/IS
13	**1.002**		Can't cancel incomplete actions using cancel or back.	2	3	4	24	Opportunity testing.	N/A
14	**1.003**		Can't get and select from list of managed files.	2	1	3	6	Extensive testing.	Dev/UC-Test/IS
15	**1.004**		Can't get and select from a list of non-managed files.	2	1	4	8	Balanced testing.	Dev/UC-Test/IS
16	**1.005**		Check-in of new document to DMS fails.	2	1	2	4	Extensive testing.	Dev/UC-Test/IS
17	**1.006**		Check-in of existing document to DMS fails.	1	1	2	2	Extensive testing.	Dev/UC-Test/IS
18	**1.007**		Check-out of DMS documents fails.	1	1	2	2	Extensive testing.	Dev/UC-Test/IS
19	**1.008**		Load of online DMS document fails.	1	1	3	3	Extensive testing.	Dev/UC-Test/IS
20	**1.009**		Load of near online DMS document fails..	1	2	3	6	Extensive testing.	Dev/UC-Test/IS
21	**1.010**		Load of offline DMS document fails.	1	2	3	6	Extensive testing.	Dev/UC-Test/IS
22	**1.011**		Can't display or modify doc properties in DMS before saving.	2	3	5	30	Opportunity testing.	N/A
23	**Quality Risk Category Priority**						2		
25	**2.000**	**Load, Capacity, and Volume**	Failures in scaling of system to expected peak concurrent usage levels						
26	**2.001**		System fails at or before 25 concurrent users.	1	1	3	3	Extensive testing.	Test/S
27	**2.002**		System fails at or before 255 concurrent users.	1	3	2	6	Extensive testing.	Test/S
28	**2.003**		System disallows 255 or fewer user accounts.	1	1	3	3	Extensive testing.	Test/S
29	**2.004**		System disallows 32,767 or fewer user accounts.	1	3	3	9	Balanced testing.	Test/S
30	**2.005**		System fails on documents larger than 100KB.	1	1	1	1	Extensive testing.	Test/S

Рис. 2.1. Начало отчета о проведении первичного FMEA

Привлечение подходящих участников

Любой сотрудник компании может иметь мнение по поводу рисков качества системы и ожиданий качества со стороны заказчиков и пользователей. При проведении анализа рисков качества подбор участников основывается не на их мнениях, а на том, кто обладает интуицией и может понять, что определяет качество и какие риски качества системы существуют. Должны рассматриваться:

- Цели, с которыми заказчики приобретают систему, а пользователи ее применяют
- Полезные свойства системы
- Частная задача, которую решает система, и общие цели компании, предполагающей использовать систему
- Стратегический маркетинг и направление технического развития продуктового ряда (иногда называемые *направлением развития продукта*) или семейства систем, а также компаний, разрабатывающих, сопровождающих, приобретающих и использующих систему
- Процесс и технологии, применяемые в проектировании, разработке и сопровождении системы
- Возможности и ограничения процесса тестирования, группы тестирования и системы тестов

В таблице 2.2 представлены роли потенциальных участников процесса анализа. Заметим, что компании, разрабатывающие рыночные или заказные продукты, обычно отличаются от компаний, которые производят продукты для внутреннего употребления, поэтому в вашей компании необязательно будут существовать все роли.

Таблица 2.2 — это только фрагмент списка участников, которые могут быть привлечены к анализу рисков. Нужно ли (да и хочется ли) приглашать всех? Вероятно, нет. Анализ рисков качества таков, что для требований нужен один список участников, для архитектуры — другой, для разработки — третий. Участники со стороны группы тестирования могут оставаться теми же, либо тест-менеджер делегирует тестировщиков, которые являются экспертами в областях деятельности низкого уровня, а сам участвует персонально только в анализе требований.

Некоторые из заинтересованных лиц могут быть недоступны для участия в анализе рисков качества. Наиболее важные лица — это заказчики и пользователи. В идеале, если возможно, их необходимо привлекать к анализу рисков качества на основе требований, и компании, разрабатывающие системы для внутреннего применения, могут иногда так делать. Однако в проектах по созданию рыночных и заказных систем заказчики и пользователи обычно не принимают участия в процессе анализа рисков качества, причем будущие заказчики и пользователи часто неизвестны заранее. В этом случае те ключевые заинтересованные лица, которые наиболее плотно работают с потенциальными заказчиками и пользователями — менеджеры продаж, маркетинга, техподдержки и т.п., — в ходе анализа должны представлять интересы пользователей и заказчиков, выступая в качестве доверенных лиц отсутствующих участников.

Таблица 2.2. Вероятные участники процесса анализа рисков качества

Роль в процессе анализа	Роль в проекте (компании)	Оценивает
Позиционирование системы на рынке	Менеджер по маркетингу	Ценность системы для заказчика или пользователя
Сбор системных требований	Бизнес-аналитики, системные аналитики	Требования пользователей и основное предназначение системы
Выработка стратегического направления для продуктового ряда	Менеджер по маркетингу, менеджер продукта, главный архитектор, главный научный сотрудник	План развития продукта или системы
Выбор стратегического направления технологического развития и руководство им	Главный архитектор, менеджер разработки, вице-президент по разработке, главный научный сотрудник, главный информатор	Техническая сторона долгосрочного развития
Проектирование системы, предназначенной для тестирования	Бизнес-аналитик, системный аналитик, архитектор продукта, менеджер разработки, ведущий программист	Ограничения системной архитектуры
Сборка системы, предназначенной для тестирования	Менеджер разработки, ведущий программист, программист, технический персонал, системный аналитик	Ограничения реализации системы
Проектирование системы тестов	Тест-менеджер, ведущий тестировщик	Ограничения тестирования
Документирование свойств системы	Технический писатель, менеджер технических публикаций	Что вы сообщаете пользователям о реальном поведении системы
Тестирование системы	Ведущий тестировщик, тестировщик, тестировщик-лаборант	Дефекты, обнаруженные в системах, разработанных ранее, сложные тестовые сценарии
Взаимодействие с заказчиками и пользователями	Менеджер по продажам, торговый агент, специалист по продажам, сотрудник по связи с клиентами, бизнес-аналитик	О каких свойствах системы беспокоятся заказчики
Работа с производителями подсистем	Инженер по качеству со стороны поставщика	Что в подсистемах тестировали их производители, известные дефекты в подсистемах
Поставка системы	Инженер сборок, менеджер сборок, администратор информационной системы, ИТ-менеджер	Технические проблемы, связанные с разработкой системы

Таблица 2.2 (*продолжение*)

Роль в процессе анализа	Роль в проекте (компании)	Оценивает
Поддержка среды поставки	Администратор сети, системный администратор, менеджер конфигурации, администратор информационной системы	Работа сетей, аппаратного обеспечения, сосуществование различных компонентов ПО; плановые процедуры; процедуры обработки исключительных ситуаций (например, восстановление после сбоя)
Поддержка пользователей и заказчиков	Менеджер поддержки пользователей, менеджер технической поддержки, мейджер поддержки заказчиков, менеджер по работе с заказчиком, сотрудник системы поддержки пользователей в сети, администратор информационной системы	Параметры реальных пользователей, известные дефекты
Устранение дефектов из развернутой системы	Сотрудник службы поддержки заказчика, программист службы технической поддержки, технический персонал	Известные дефекты, проблемы внесения изменений в развернутую систему
Финансирование или помощь в финансировании разработки или сопровождения системы	Заказчик (внутренний или внешний)	Проблемы, решение которых ожидается с помощью системы, или новые полезные возможности, которые привнесет система
Явное или косвенное использование системы	Различные классы пользователей	Как пользователи применяют систему для решения задач или получения прибыли; каковы типичные наборы данных и сценарии использования системы; какие проблемы были у пользователей ранее
Ответственность за процессы оценки качества	Специалист по качеству, менеджер группы процессов разработки ПО, менеджер по качеству	Гарантия согласованности процесса и качества в ходе разработки всей системы
Персональная ответственность за качество продукта	Директор по качеству, генеральный директор, руководитель администрации, президент	Насколько должен быть качественным продукт, чтобы его можно было выпустить под маркой компании

Мой опыт анализа рисков качества показывает, что наилучшие результаты достигаются, когда привлекается максимально возможное число заинтересованных лиц с разными интересами и наличием квалификации в разных областях. Чем больше обнаруживается точек зрения в компании, тем сложнее перечень рисков и более точной будет расстановка приоритетов рисков. Хотя для анализа рисков качества действительно полезны спецификации требований и архитектуры, привлечение участников,

которые являются специалистами в разных областях, — более важный фактор успеха процесса анализа рисков. Позвольте проиллюстрировать данный факт двумя историями.

В первом случае ведущие тестировщики и я как тест-менеджер применяли анализ видов ошибок и их влияния, после того как проектная группа подготовила качественную проектную документацию, в том числе спецификации требований и архитектуры. Однако проектная группа не участвовала в проведении анализа. Вместо этого мы отправляли по электронной почте черновики документов сотрудникам рабочей группы проекта. Многие из ключевых заинтересованных лиц отказались потратить время, чтобы предоставить нам полезную информацию. Кончилось все тем, что мы остановили выполнение тестов, на которые мы потратили много времени и сил, разрабатывая данные, сценарии, тестовую среду и т.п., посередине процесса, поскольку наши оценки степеней рисков не совпали с приоритетами проекта. Другими словами, преимуществ от определения того, что тестировать, а что — нет, не было получено, поскольку заинтересованные лица не участвовали в оценке. Проектная документация оказалась не в состоянии адекватно заменить вклад разных специалистов в анализ рисков качества.

Во втором случае метод анализа видов ошибок и их влияния использовался при отсутствии какой бы то ни было проектной документации. Группа специалистов различных направлений, участвовавшая в процессе анализа рисков качества, хранила все знания, необходимые для проведения совещаний, у себя в головах. Анализ оказался успешным. Предотвращение дефектов за счет изменений в процессе программирования и обнаружение ошибок за счет правильного выбора приоритетов тестирования дали возможность своевременно выпустить версию качественного продукта.

Конечно, отсутствие документации требований и архитектуры отрицательно повлияло на ход проекта, включая тестирование. Однако существенного влияния на анализ рисков качества это не оказало.

Выбор подходящей методики

Существует много методик, которые могут использовать тестировщики при анализе рисков качества. Я хорошо знаком с четырьмя методиками, которые регулярно и успешно применяются.

- *Неформальный анализ.* Определить потенциальные проблемы системы, а затем совместно с ключевыми заинтересованными лицами расставить их по приоритетам. В качестве источников списка потенциальных проблем могут использоваться книги, стандартные списки проблем качества ПО, списки проблем предметной области, общие знания о видах дефектов, присущих типу системы, которая находится в разработке[1].
- *Метод стандарта ANSI/ISO 9126.* Для шести характеристик качества — функциональности, надежности, удобства использования, произво-

[1] Четыре перечня типичных дефектов системы можно найти в главе о систематизации дефектов в книге Бориса Бейзера (Boris Beizer) "*Software Testing Techniques*", в приложении, содержащем перечень типичных ошибок, в книгах Сема Канера и др. (Cem Kaner et.al) "*Testing Computer Software*", Петера Ноймана (Peter Neumann) "Computer-Related Risks» и в списке рисков качества программных и аппаратных средств в книге Рэкса Блэка (Rex Black) "*Managing the Testing Process, Second Edition*".

дительности, удобства сопровождения и мобильности (для краткого обозначения шести характеристик качества в оригинале используется сокращение FRUEMP по первым буквам английских слов, обозначающих эти характеристики) — определяются признаки качества, метрики для измерения качества системы на основе каждого признака качества и способ перевода этих метрик в оценки качества[1].

- *Методика стоимости обнаружения проблемы* (cost of exposure, COE). Определяются потенциальные проблемы и их влияние, а затем оцениваются стоимость наличия проблем для бизнеса и вероятность их возникновения.
- *Анализ видов ошибок и их влияния* (FMEA). Определяются потенциальные проблемы, оценивается их влияние (на заказчиков и пользователей), а затем выполняется классификация (с учетом мнений заинтересованных лиц) по серьезности (влияние на систему), приоритетности (влияние на бизнес) и вероятности (возможность возникновения дефекта).

Методики перечислены в порядке усложнения структуры и увеличения точности. В тех случаях, когда это уместно, я предпочитаю точность и количественные оценки и, следовательно, использую метод анализа видов ошибок и их влияния. Однако я обнаружил, что этот метод не всегда работает в неформальном или быстро меняющемся контексте проекта. И я выбираю ту методику, которая соответствует общему уровню структуры и степени точности, присущим проекту, находящемуся в работе. Методы формального анализа — анализ видов ошибок и их влияния — и метод стоимости обнаружения проблемы требуют привлечения проектной группы к участию в продолжительном совещании. Неформальный подход обычно означает встречи один на один с ключевыми заинтересованными сторонами и последующее рецензирование короткого документа для достижения согласия. Таким образом, неформальный подход приводит к упрощенному процессу, который, вероятно, проще реализовать.

Более структурированные и формализованные подходы позволяют добиться более тщательно откалиброванного процесса тестирования с более точно определенными целями. В примере с продуктом "Суматра" присвоение уровней тестирования, основанное на значениях приоритета рисков, — пример точного определения целей. Однако применение таких методов требует большей квалификации и знаний, чем применение неформальных методов. Например, мне потребовалось поучаствовать в шести или семи

---

[1] См. руководство по стандарту ANSI/ISO 9126. Обсуждение применения стандарта приводится в книге Вана Веенендааля и др. (van Veenendaal et.al.) "Measuring Software Product Quality during Testing". Этот вопрос также рассматривается в книге Поула (Pol) и др., посвященной T-MAP. Кроме того, Стефан Кан (Stephen Kan) в "*Metrics and Models in Software Quality Engineering, Second Edition*" дает ссылки на два других списка характеристик качества с запоминающимися сокращениями на английском языке: функциональность, удобство использования, надежность, производительность и удобство сопровождения (FURPS), а также устойчивость, удобство использования, производительность, надежность, удобство установки, удобство сопровождения и работоспособность (CUPRIMO). Эти два сокращения можно добавить к ранее рассмотренному способу FRUEMP. Наконец, в главе 11 книги "*Software Requirements*" Карл Вигерс (Karl Wiegers) приводит интересное суждение по поводу атрибутов качества программ, включая компромиссы между атрибутами.

проектах, чтобы почувствовать себя опытным и уверенно использовать анализ видов ошибок и их влияния. Если нет времени на обучение, не надо пытаться сразу объять необъятное. Чтобы просто погрузиться в анализ рисков качества, возможно, лучше начать с неформального подхода.

Можно перейти от неформальных методов к формальным, если выявится необходимость в более точной оценке. Я часто использую неформальную методику в качестве базы для методики FMEA как раз для получения исходного множества категорий рисков качества.

Какую методику ни использовать, все равно желаемым результатом является список рисков качества системы, расставленных по приоритетам. Значения приоритетов, присвоенные рискам, определяют в основном необходимый объем тестирования. Обычно серьезные риски требуют тщательного тестирования, риски средней степени нуждаются в ограниченном тестировании, несущественные риски не требуют тестирования вовсе.

Уровням риска могут быть присвоены нечисловые (например, высокий, средний, низкий, отсутствует) или числовые значения. Мне довелось работать с разными числовыми шкалами. Предпочитаю шкалу от 1 до 5 в порядке убывания для каждой из трех переменных в методе анализа видов ошибок и их влияния. Однако один из участников моего семинара рассказал, что его сотрудник использует шкалу от 0 до 9 в порядке возрастания в методике определения качества функциональности (Quality Functional Deployment, QFD), причем 9 означает высокий, 3 — средний, 1 — низкий, 0 — нет. Стаматис (D. H. Stamatis) в книге "*Failure Mode and Effect Analysis*" использует шкалу от 1 до 10 в порядке возрастания.

Методики стоимости обнаружения проблемы и анализа видов ошибок и их влияния различаются способом представления информации, но мной было обнаружено, что можно преобразовать результаты, полученные по одной методике, в результаты другой. На рис. 2.2 показано, как трансформируются результаты анализа видов ошибок и их влияния (см. рис. 2.1) при преобразовании в результаты методики стоимости обнаружения проблемы. Преобразование потребовало около часа времени, включая конвертацию шкалы от 1 до 5 в шкалу для определения качества функциональности.

Заключительное замечание касается того, что нужно помнить, что нет необходимости все время использовать одну и ту же методику, тем более на разных фазах проекта. В некоторых случаях может быть полезным проведение анализа рисков проблемной области для комплексного и интеграционного тестирования и анализа рисков проектирования и реализации для модульного и компонентного тестирования. Вероятно, неформального подхода вполне достаточно для модульного и компонентного тестирования, в то время как для комплексного и системного тестирования потребуется более формализованный подход (методики стоимости обнаружения проблемы или анализа видов ошибок и их влияния). Пожертвовав совсем немного полнотой состава и координацией специалистов в разных областях и разных проектах, за счет гибкости более простого процесса, требующего меньших ресурсов, можно продвинуться вперед, когда время является критической величиной.

Cost of Exposure Analysis (Quality Risk Analysis) Form

System Name: SpeedyWriter (with Document Management System add-on)					Model/Product: SpeedyWriter Release 3.1 (w/ DMS)		
Product Manager: Kate Hernandez					Target Release Date: 1Q2003		
Project Manager: Jenny Kaufman					Prepared By: Jamal Brown		
Other Stakeholders: B. McDaniels (Dev), M. del Oro (Mkt), C. Negraev (Support)					COE Analyzed On: September 11, 2002		
Test Manager: Jamal Brown					COE Revised On: N/A		
Risk ID Number	Quality Risk Category	Failure Mode/Quality Risk/Effect	Potential Cost	Likelihood	Risk Priority Number	Recommended Action	Who/ Which Phase (Unit, Component, Integration, System)?
1.000	Functionality	Failures that cause specific features not to work					
1.001		Regression of existing SpeedyWriter features.	9	3	27	Rerun entire R3.0 test set.	Test/IS
1.002		Can't cancel incomplete actions using cancel or back.	3	1	3	Opportunity testing.	N/A
1.003		Can't get and select from list of managed files.	9	3	27	Extensive testing.	Dev/UC-Test/IS
1.004		Can't get and select from a list of non-managed files.	9	1	9	Balanced testing.	Dev/UC-Test/IS
1.005		Check-in of new document to DMS fails.	9	9	81	Extensive testing.	Dev/UC-Test/IS
1.006		Check-in of existing document to DMS fails.	9	9	81	Extensive testing.	Dev/UC-Test/IS
1.007		Check-out of DMS documents fails.	9	9	81	Extensive testing.	Dev/UC-Test/IS
1.008		Load of online DMS document fails.	9	3	27	Extensive testing.	Dev/UC-Test/IS
1.009		Load of near online DMS document fails.	9	3	27	Extensive testing.	Dev/UC-Test/IS
1.010		Load of offline DMS document fails.	9	3	27	Extensive testing.	Dev/UC-Test/IS
1.011		Can't display or modify doc properties in DMS before saving.	3	0	0	Report observed bugs.	N/A
Quality Risk Category Priority					81		
2.000	Load, Capacity, and Volume	Failures in scaling of system to expected peak concurrent usage levels					
2.001		System fails at or before 25 concurrent users.	9	3	27	Extensive testing.	Test/S
2.002		System fails at or before 255 concurrent users.	9	3	27	Extensive testing.	Test/S
2.003		System disallows 255 or fewer user accounts.	9	3	27	Extensive testing.	Test/S
2.004		System disallows 32,767 or fewer user accounts.	9	3	27	Balanced testing.	Test/S
2.005		System fails on documents larger than 100KB.	9	9	81	Extensive testing.	Test/S

Рис. 2.2. Начало отчета о проведении первичного анализа стоимости обнаружения проблемы

Отделение немногих важных аспектов от множества тривиальных

Независимо от того, какая методика применяется, сила анализа рисков качества зависит от того, насколько удается отделить серьезные риски, с которыми нужно тщательно работать, средние риски, которым необходимо уделить определенное внимание, и незначительные риски, которыми по большей части можно пренебречь. Можно сравнить методики анализа рисков с решетом, которое отсеивает камни и пропускает песок.

Мне довелось наблюдать естественное развитие группы, анализирующей риски, когда грубое решето становилось тонким фильтром. Участники процесса анализа рисков не спешат принимать грубые решения и проводят скрупулезное исследование, необходимое для того, чтобы отделить потенциальные проблемы от простого раздражения. Если этого не происходит, каждый риск классифицируется как высокий. Это может быть верно для систем с особыми требованиями к безопасности, но для большинства систем существует большое число ошибок, которые не помешают группе разработки выпустить версию, либо эти ошибки не связаны ни с одним реалистичным сценарием использования системы.

При качественном тестировании идет поиск дефектов, которые могут оказать влияние на мнение пользователей и заказчиков о системе. Как правило, есть проблемы, без устранения которых менеджеры продаж, маркетинга или технической поддержки задерживают выпуск версии. Хотя обычно приветствуется обнаружение других видов проблем в начале тестирования, обычно все заканчивается необработанными отчетами об ошибках и невыполненными наборами тестов в конце проекта.

Вышесказанное позволяет четко очертить рамки совещаний, посвященных анализу рисков качества. Неверно, что ключевым вопросом на них является *выяснение того*, хотим ли мы, чтобы версия системы не имела ошибок определенного вида. Такой вопрос не имеет отношения к реалиям проекта. В принципе все выступают *за* улучшение качества. Однако ключевым вопросом является выяснение того, от чего мы *готовы* отказаться, чтобы гарантировать, что в версии системы нет ошибок определенного вида. Другими словами, является ли потенциальная проблема качества такой, что за ее обнаружение и исправление группа управления проектом готова расплачиваться дополнительными финансами на тестирование, ожидаемой прибылью или готова просто удалить эти свойства системы в случае задержки выпуска или поставки версии? Таким образом, очерчивание рамок анализа рисков качества позволяет более точно расставить риски по приоритетам.

Поиск возможных дефектов до начала тестирования

Часто результаты анализа рисков качества начинают проявляться еще до написания первого тестового сценария. Рассмотрим процесс анализа рисков качества как форму структурированного перекрестного рецензирования. Как при любом процессе рецензирования, полученные результаты могут распространяться и оказывать влияние на весь ход проекта. Обычно это происходит тремя основными способами.

Во-первых, в ходе анализа могут быть обнаружены противоречивые, не являющиеся тестопригодными или нереализуемые требования, слабости архитектуры и даже, если проверке подвергается код, потенциально проблемные структуры данных и фрагменты управляющей логики программ. В этом смысле анализ рисков качества сводится к рецензированию этих документов через уникальную призму рисков качества системы[1].

Во-вторых, помимо поиска проблем, процесс анализа рисков качества может быть также направлен на поиск решений. Например, в качестве действия по снижению определенных рисков могут быть рекомендованы более тщательное рецензирование кода, использование формальных утверждений в системе, пошаговая трассировка выполнения программы в критических областях и другие методы безопасного программирования и оценки качества. В нашем примере анализ видов ошибок и их влияния можно использовать с целью оценки качества проекта "Суматра".

В-третьих, процесс расстановки приоритетов выступает в качестве раннего предупреждения программистов о том, что будет тестироваться. Как отмечал Борис Бейзер, иногда всего лишь "угрозы тестирования" достаточно для предотвращения ошибок[2]. Если группе разработки известно, на поиск каких дефектов будет ориентировано тестирование, ее участники, вероятно, потратят больше времени на проверку работы соответствующих компонентов системы.

Таким образом, анализ рисков качества уменьшает число ошибок, которые обнаруживаются в ходе тестирования. Как мы увидим в главе 4, исправление ошибок, обнаруженных на стадиях сбора требований и разработки архитектуры, обходится существенно дешевле, чем исправление ошибок, обнаруженных на фазе тестирования. До выбора методов, направленных на предотвращение дефектов, компания должна расстаться с использованием непродуктивных способов работы компании. Примерами способов, мешающих предотвращению ошибок, являются стимулирование или оценка работы тестировщика на основе числа дефектов, найденных в ходе тестирования. В этом случае сообразительные тестировщики быстро понимают, что начинают работать против собственных интересов, если помогают предотвращению дефектов.

Когда оценка рисков качества используется для предупредительных действий, например для уточнения требований, усовершенствования архитектуры, безопасного программирования и других мер по поддержке качества, необходимо, чтобы каждый участник этого процесса понимал необходимость дальнейшего смягчения важных рисков качества за счет соответствующего объема тестирования. Как тестировщик-профессионал я верю в предотвращение дефектов. Однако я также полон профессионального пессимизма. Как можно узнать, что ошибка предотвращена, пока не проведешь тестирование с целью поиска этой ошибки и не убе-

---

[1] В идеале проектная группа также применяет и другие методы рецензирования требований, архитектуры и кода, а не полагается лишь на анализ рисков качества. Например, способы рецензирования требований рассматриваются в книге Карла Вигерса (Karl Wiegers) "Software Requirements», а также в статье Роджера Драбика (Rodger Drabick) "*On-Track Requirements*" в "*Software Testing and Quality Engineering*", Vol.1, Issue 3 (May/June 1999).

[2] В книге Бориса Бейзера (Boris Beizer) "*Software Testing Techniques*" есть целый ряд наблюдений о том, как тестирование выполняет свою миссию и в каких случаях не выполняет.

дишься, что ее действительно нет? Как говорит русская пословица: «Доверяй, но проверяй».

Своевременное начало

В примере с проектом "Суматра" Джамал и его коллеги приступили к анализу рисков качества по завершении рецензирования первой черновой версии документа с описанием требований. Правильно ли выбрано время начала? Группа проекта "Суматра" не имела спецификаций архитектуры, в частности архитектуры сети и программного обеспечения, которые могли бы способствовать рассмотрению разнообразия видов ошибок, связанных с архитектурой, например состояний гонок в базах данных и одиночных точек сбоев в нерезервируемых серверах. В дополнение к информированию групп, занимающихся анализом рисков качества, о других вероятных проблемах информация об архитектуре дает возможность участникам увидеть технические риски. Это позволяет более точно оценить вероятность вида ошибки, существующего в системе. Таким образом, нужно ли откладывать начало анализа рисков качества до завершения подготовки спецификаций архитектуры?

До определенной степени ответ на этот вопрос зависит от того, какие методы тестирования планируется применять. С одной стороны, если есть намерение уделить большое внимание структурному тестированию, понимание особенностей архитектуры и реализации системы может быть важно для анализа рисков. Если, с другой стороны, тестирование будет основано на поведении системы, не очень важно, каким образом устроена система изнутри. Хотя знание деталей ее устройства позволяет улучшить оценку рисков с точки зрения вероятности, на пользователя оказывают влияние множество других факторов, которые будут приняты во внимание при анализе рисков качества (в частности, факторы, рассмотренные группой "Суматра" при использовании анализа видов ошибок и их влияния).

Также можно провести анализ рисков качества требований, а после завершения спецификаций архитектуры организовать короткое дополнительное совещание по рискам. Более того, можно встретиться и в третий раз, когда станет доступной дополнительная информация о деталях и компонентах архитектуры. Я бы не рекомендовал еженедельное обсуждение рисков качества, однако небольшое совещание по уточнению оценок рисков качества по достижении основных рубежей проекта — отличная идея. По мере того как факторы, влияющие на оценку рисков, перевешивают в сторону технологий, список участников может изменяться за счет приглашения системных проектировщиков и старших программистов и исключения сотрудников отделов продаж и маркетинга.

Готовность к изменениям

Источники информации, влияющие на оценку рисков, появляются не только в моменты достижения рубежей проекта. Приоритеты бизнеса из-

меняются во времени даже в хорошо управляемых и стабильных проектах. Вероятность проблем определенного вида становится очевиднее по мере того, как вы получаете опыт тестирования реальной системы. Можно рассчитывать на дополнительные знания о рисках качества системы в процессе проектирования и разработки системы, а также в процессе разработки тестов. Иногда оказывается, что мы недооценили ресурсы и потребуется еще время на достижение уровня тестирования, согласованного в ходе анализа оценки рисков. Это означает, что необходимы уточнения и подстройки процесса в ходе всего подпроекта тестирования. Процесс анализа рисков качества нуждается в непрерывном изучении рисков и в улучшении управления этим процессом.

Традиционный подход к управлению изменениями состоит в помещении документа с описанием анализа рисков качества в проектный репозиторий и постановке под контроль коллегиального органа, как правило, группы управления изменениями (Change Control Board, CCB) (см. главу 16). По мере того как группа проекта получает новую информацию о рисках и по достижении рубежей проекта, она повторяет анализ. Тест-менеджер или кто-то другой, отвечающий за процесс анализа рисков, берет документ с описанием анализа рисков из репозитория на изменение, обновляет его, а затем выносит обновленный документ на утверждение совещанием группы управления изменениями. В то же время вполне возможно, что в ходе анализа рисков выясняется, что некоторые риски стали менее вероятными, например, если установлено, что сетевая или серверная архитектура, используемая для реализации некоторого фрагмента системы, является устойчивой к некоторому сбою. Обычно обновленные результаты анализа рисков качества требуют некоторого инкрементного наращивания границ или объема тестирования. Если планирование было проведено качественно (см. главы 6 и 7), участники проекта не удивятся такому развитию событий.

Иногда можно обнаружить большие новые области рисков, которые влекут за собой существенные изменения в границах тестирования. Предположим, что спецификация архитектуры для продукта SpeedyWriter показывает, что система должна работать не только в интранете компании, но и в Интернете как система ASP (поставщик услуг по аренде приложений, модель ASP (Application Service Provider) — концепция аренды приложений с размещением всей серверной инфраструктуры у провайдера этих услуг). Хотя это не входит в перечень требований к продукту для готовящейся версии, системный проектировщик принял решение, что пора технически готовить систему к этому. В результате необходимо добавить новые области тестирования, в том числе медленные коммутируемые (dial-up) соединения, а также обновить задачи тестирования в некоторых ранее включенных областях, это может быть, например, проверка совместимости и безопасности браузеров.

Вероятность риска может изменяться не только при обнаружении новых рисков, но и в результате его возрастания в тех областях системы, которые, казалось бы, уже исследованы. В качестве примера приведу случай из моей практики. В одном из проектов было довольно поздно обнаружено, что ранее согласованное с одним из поставщиков подсистем тестирование производительности не проводилось. Я обнаружил это упущение,

когда выполнял трассировку тестового покрытия относительно рисков качества (см. главы 10 и 11). Эта находка позволила проектной группе увеличить объем тестирования, проверяющего риски, связанные с производительностью системы, в рамках фаз комплексного и системного тестирования. Конечно, было бы лучше, если бы этого упущения не было вовсе, однако анализ рисков качества предоставляет нам инструмент, позволяющий учесть и такое изменение.

В конечном счете, любое изменение требует пересмотра соглашений о функциональности, качестве, сроках и бюджете, присущих разработке любой системы. Эффективные группы управления проектом знают, как при получении новых сведений использовать возможности для пересмотра решений. Имея больше информации, можно принимать более успешные решения. Я работаю над процессами, которые позволят поддерживать непрерывное совершенствование понимания системы. Группы управления изменениями часто становятся частью такого процесса и организуют форум для открытых дискуссий и поиска компромиссов по поводу ранее принятых решений.

В некоторых компаниях нет такого способа организации сотрудников. В этом случае можно использовать регулярные обновления документа с описанием анализа рисков качества параллельно с проведением обзоров изменений совместно с заинтересованными лицами. Если с их стороны вы чувствуете поддержку, планируйте совещания по достижении основных рубежей проекта. Это также будет способствовать более качественной работе группы тестирования, координирующей принятие решений при отсутствии структуры, управляющей обсуждением, хотя тестировщики могут по-прежнему получать существенную отдачу от процесса анализа рисков качества и последующих обновлений итогов этого анализа.

Считаю, что необходимо быть предельно внимательным к организационным, функциональным, технологическим и проектным изменениям и принимать в расчет любые дополнительные знания о проекте, которые оказывают воздействие на риски качества. Также надо быть готовым к упреждающим мерам в случае возникновения изменений. Я отдаю себе отчет в том, что делать работу по управлению рисками качества более эффективной означает реагировать на изменения, а не рассматривать их как ужасные обстоятельства.

Решение проблем

До начала внедрения анализа рисков качества в компании необходимо принять ряд решений и подготовить ряд мероприятий по преодолению препятствий, с которыми наверняка придется столкнуться.

Реальная оценка рисков

Риск — это понятие, которое многие считают знакомым, но редко умеют с ним рационально работать. Некоторые люди, не задумываясь, водят каждый день машину, но боятся погибнуть в авиакатастрофе, хотя

из-за автомобильных аварий гибнет в сотни раз больше людей. Попытки законодательно исключить существование рисков продолжают делать правительства многих стран мира, принимая огромные бесполезные усилия во имя минимальных сокращений вероятности рисков. Обычно в школах и колледжах учащиеся не уделяют много внимания вероятности, статистике и рискам, если они не обучаются математике, бизнесу, инвестициям или страхованию. Лишь очень немногие журналисты в состоянии разъяснить читателям суть риска в статьях, которые затрагивают рассматриваемую тему. Я стал понимать степени риска в моей жизни и эффективно управлять ими только тогда, когда изучил риски и статистику[1].

Неспособность рационально работать с рисками может вызвать проблемы в проектах разработки, сопровождения и поставки системы. Как-то одна тест-менеджер сказала мне, что она испытывает трудности при объяснении своим менеджерам рисков качества системы. В ходе двух проектов разработки она предупреждала их о серьезных упущениях в покрытии тестами и говорила о возможном отрицательном эффекте, который может быть произведен на заказчика по причине наличия ошибок в этих областях. После выпуска версий, когда этих ошибок не обнаруживалось, менеджеры говорили ей: "Посмотрите, в конце концов риск не был столь велик". Она сказала мне, что просто сбита с толку и начинает думать, что они правы[2].

Не сомневаюсь, что иногда менеджеры проектов правы, делая такого рода заявления. Но иногда им просто везет. Иногда очень везет. Но везение не может повторяться всегда.

Обманчивая сторона риска состоит в том, что, когда мы пытаемся заглянуть в будущее, существует некоторая статистическая вероятность того, что нежелательное событие произойдет. Точное значение этой вероятности, расположенное где-то между нулем и единицей, обычно неизвестно. Когда мы оглядываемся в прошлое, нежелательное событие либо произошло, либо не произошло. Вероятности событий в прошлом либо равны нулю, либо равны единице. Выводы об уровнях рисков на основе небольшого числа удачных исходов означают попытку выдать желаемое за действительное, когда идет речь о способе проявления риска. История индустрии программирования слишком богата погибшими компаниями (например, когда-то в секторе баз данных для персональных компьютеров доминировала компания Ashton-Tate), чтобы пренебрегать опасными рисками.

Полезно понимать, что ожидание принятия рациональных решений по рискам от всех участников проекта — это ожидание чересчур многого. При принятии решений по рискам добавляется большое число иррацио-

---

[1] Две отличные, практичные и нетехнические книги о рисках: Питер Бернстайн (Peter Bernstein) "*Against the Gods*" и Джеймс Уолш (James Walsh) "*True Odds*". В этих книгах нет ничего о системном тестировании, но они помогли мне понять риски и, как следствие, смысл моей работы. Полезными также могут оказаться большинство учебников по статистике для колледжей.

[2] Интересная статья об этом политическом ракурсе "The Risk in Risk Management", написанная Питером де Ягером (Peter de Jaeger), появилась в журнале "*Software Testing and Quality Engineering*", Volume 3, Issue 4 (Jul/Aug 2001).

нальных факторов. Если какой-то риск кажется пугающим и последствия проявляются немедленно или в ближайшей перспективе или известно о чужом опыте, когда была заплачена высокая цена за этот риск, люди пытаются всеми силами избежать его, даже если на самом деле вероятность и влияние риска не требуют этого. И наоборот, если людям кажется, что они держат риск под контролем, если это риск, который они выбрали сознательно (и оценили рационально), если возможные последствия находятся где-то в отдаленном будущем или если аналогичные риски уже были и не оказали отрицательного эффекта, эти риски принимаются с большим энтузиазмом, нежели диктует логика[1].

Тестировщики-профессионалы имеют в своем арсенале множество пугающих рассказов о том, как решения о выборе того или иного риска приводили к полной катастрофе. Следовательно, по причинам и рационального, и иррационального характера можно сделать вывод о том, что тестировщики имеют склонность сильно преувеличивать опасность рисков. Я встречал тестировщиков, которые предупреждали об определенных "любимых" рисках качества на протяжении многих проектов. Причем во всех проектах их менеджеры вновь и вновь игнорировали эти предупреждения. Верным решением было не тратить время на обсуждение этих рисков. На протяжении десятков или около того версий проблемы, относящиеся к этим рискам качества, ни разу не проявлялись. Тестировщики наверняка просто переоценили вероятность этих рисков качества.

Однако компания, занимающаяся, например, электронной коммерцией и решившая не проводить тестирования нагрузки и безопасности своего web-сайта, с большой вероятностью будет нести убытки, поскольку не позаботилась о снижении влияния рисков в этих областях. Компетентные менеджеры разработки, сопровождения и поставки системы умеют эффективно управлять рисками, которые имеют существенное значение для недопущения финансовых потерь компании, а иногда и ее гибели. Тест-менеджер отвечает за получение реалистичных оценок рисков качества, включая потенциальные потери, перед группой управления проектом.

Многообразие источников рисков качества

В рассматриваемом примере Джамал и группа специалистов разного профиля считают все риски качества относящимися к природе разрабатываемой системы. Они начинают с анализа перечня различных категорий проблем, присущих системам, а затем конкретизируют виды ошибок в рамках этих категорий. Другими словами, участники анализа обследуют риски качества, которые возникают из контекста системы, поскольку они реализуют функциональность с помощью некоторых технологий на отдельно взятой архитектуре аппаратного и программного обеспечения,

---

[1] Для дальнейшего изучения темы смотрите отменную презентацию Эрика Симмона (Erik Simmon) "The Human Side of Risk", впервые опубликованную в "*The Proceedings of the International Fifteenth Internet and Software Quality Week 2002*" и размещенную в настоящее время на сайте www.rexblackconsulting.com.

а не, скажем, в соединительных проводах и ячейках, в наборе шестеренок, пластмассе и древесине[1].

Однако наряду с рисками качества, связанными с самой системой, существуют риски качества, связанные с проектом, процессом и даже организационным устройством группы проекта, которая разрабатывает систему. Например, предположим, что требования продолжают изменяться почти до даты выпуска проекта. Лично я стараюсь учесть эти виды рисков при планировании процесса тестирования (см. главы 6 и 7). В то же время для снижения этих рисков можно использовать метод анализа видов ошибок и их влияния или какой-либо другой метод анализа рисков. В книге "*Failure Mode and Effect Analysis*" Стаматис рассматривает применение этой методики к рискам качества, относящимся к процессу.

Кто управляет анализом рисков качества?

Анализ рисков качества, описанный в этой главе, является мощным инструментом. Как уже упоминалось ранее, сила этого инструмента в большой степени зависит от привлечения к анализу специалистов разного профиля. Действительно, многие методы оценки качества включают в себя сложные обсуждения, выходящие за организационные пределы проекта и концентрирующие внимание ключевых заинтересованных лиц на конкретных проблемах. Однако слишком многопрофильная природа таких процессов и широкий охват специалистов и руководителей могут вынудить людей задуматься о том, кто должен инициировать процессы и содействовать их успешному осуществлению. Ответ на этот вопрос зависит от ситуации. Возможно, вы работаете в компании с формализованным процессом разработки и с группой контроля качества, задачей которой является контроль всего процесса и качества системы в целом. В этом случае анализ рисков качества, вероятно, находится в сфере ответственности какого-либо сотрудника группы качества.

В своей книге "*Software Project Survival Guide*" Стив Макконнел (Steve McConnell) исследует управление рисками в самом широком смысле. Оно направлено на выявление и снижение значительных рисков, угрожающих разработке или сопровождению системы. Макконнел рекомендует, чтобы в каждом проекте был назначен сотрудник, отвечающий за выявление, расстановку приоритетов и корректное смягчение рисков[2]. В таких случаях ответственный за риски, очевидно, будет управлять анализом рисков качества.

По моим наблюдениям, обычно группа тестирования берет управление процессом анализа рисков качества в свои руки. Поскольку процесс наиболее эффективен только при привлечении специалистов разного профиля, это может иметь серьезные технические, персональные и по-

---

[1] В музее искусства и наук в Париже есть интересная коллекция механических вычислительных устройств, включая абак, экземпляры аналитической машины Беббиджа и логарифмической линейки. Увидев эти устройства, я подумал о том, как бы я стал проводить тестирование на них. Затем, я начал размышлять, как повлияет на тестирование новый скачок в компьютерных технологиях (например, изобретение квантового компьютера).

[2] См. "*Software Project Survival Guide*", с. 93–101.

литические последствия для тестировщика, который инициирует анализ рисков качества и способствует его проведению.

На техническом уровне этот человек должен обладать соответствующими навыками в использовании выбранной методики. Неформальные методы наиболее просты и, вероятно, являются наилучшим вариантом для новичка, особенно в случае, когда процесс разработки не слишком формализован. Если есть ощущение, что нужно апробировать более сложные методы: анализ видов ошибок и их влияния, T-MAP или ISO 9126, я бы рекомендовал внимательно прочесть материалы, на которые приведены ссылки в разделе "Выбор подходящей методики" (см. выше). Возможно, вы также захотите подготовить краткое изложение методики для других участников.

Ответственный за проведение анализа рисков качества должен иметь в своем арсенале более простые методы управления коллективом, включая мозговой штурм, разрешение конфликтов, следование повестке дня и графику и т.п. Наиболее опытные менеджеры со временем приобретают эти навыки, поэтому, если у вас не было случая научиться всему этому, для начала ознакомьтесь хотя бы с идеями. Продолжительные сложные дискуссии, подобные тем, которые возникают в ходе анализа рисков качества, требуют постоянного совершенствования навыков управления.

Кроме того, тестировщик, ведущий совещание по вопросам рисков качества, должен обладать доверием и политической поддержкой. Если по каким-либо причинам вы чувствуете, что не имеете достаточного влияния для ведения совещания, в котором участвуют приглашенные высокопоставленные менеджеры, попытайтесь заручиться поддержкой своего менеджера или привлечь опытного консультанта либо для помощи, либо для проведения совещания. Если вы ввязались в процесс и устроили из него бессмысленную болтовню, в результате чего бесполезно расходуется рабочее время участников, совещание выходит из-под контроля и не приносит полезных результатов, то вы можете нанести непоправимый ущерб как доверию, так и вашему политическому влиянию.

Исключение магии из количественных оценок рисков

Британский ученый лорд Кельвин однажды писал, что без количественных оценок нет настоящего знания. Хотя будет логической ошибкой при рассмотрении обратного утверждения считать, что, приспособив некоторый способ измерения к атрибуту, можно получить глубокое представление о его внутреннем устройстве. Числа и метрики сами по себе не создают, помимо точности модели, никаких волшебных способностей у их пользователей.

В таких методах, как анализ видов ошибок и их влияния и метод стоимости обнаружения проблем, я получаю значения приоритетов рисков и затем использую их для принятия решений о тестировании. Эти значения помогают упорядочить множество всех выявленных рисков. Другими словами, я получаю представление об относительном значении каждого риска. На рис. 2.2 можно видеть, что идентификаторы рисков 1002, 1003, 1004 и 1005 выстроены в порядке убывания значений приоритетов рисков: 3, 9, 27 и 81 соответственно.

Однако, если скрыть значения приоритетов рисков, становится трудно сделать какие-либо выводы. Компоненты значений приоритетов рисков, в особенности оценки приоритетов бизнеса и вероятности, основываются большей частью на опыте, интуиции и профессиональных знаниях. Даже значения вероятности, предмета актуарного анализа, редко основаны на статистически проверенных данных, собранных на большом числе проектов. Даже если бы у нас и были такие данные, из-за принципиальной новизны каждого проекта экстраполяция может применяться лишь ограниченно. Страховые компании экстраполируют риски, связанные с курением сигарет, на другие формы потребления табака, например курение сигар. Однако мы не можем экстраполировать виды ошибок, наблюдаемые при клиент-серверной реализации системы, на уровни риска при реализации той же системы как web-приложения.

Практический результат состоит в том, что количественные оценки рисков, несмотря на полезные модели упорядочения рисков, остаются неточными приближениями. Предусмотрительные тестировщики должны убедиться в том, что другие заинтересованные лица, особенно впервые применяющие методику, понимают ограничения, связанные с использованием количественных оценок. Если вы услышите, что участники процесса связывают точные ожидания с применением методов управления рисками, например: "Если мы снизим все риски, имеющие приоритет, равный 30 и ниже, за счет тестирования на всех фазах, мы уменьшим потери от низкого качества в данной версии на 37.5%", — не сомневайтесь, что нужно потратить время и усилия на то, чтобы убедить таких людей, что не стоит ожидать чудес от количественных оценок.

Наконец убедитесь, что *вы* сами не жертвуете пониманием проблем в пользу магии чисел. Я уже показал на примерах, каков должен быть объем тестирования, если он управляется значениями приоритетов рисков. В реальности мне часто приходится тестировать некоторые риски более тщательно, поскольку, исходя из моего опыта, это разумно, в то время как их значения приоритетов показывают, что эти риски менее опасны, чем другие. Рассмотренные количественные методы являются полезными, если их применять осмотрительно и с достаточной гибкостью.

Как работать с вариациями значений оценок

До определенной степени оценка рисков зависит от субъективного взгляда, от суждений участников анализа. В методе анализа видов ошибок и их влияния серьезность потенциальной проблемы обычно очевидна, что в меньшей степени относится к приоритету и вероятности. Ценность числового значения, для которого существуют проблемы субъективности и индивидуальной оценки, снижается, если заинтересованные лица соглашаются с оценкой. Если относительно определенного риска с низким приоритетом достигается консенсус, маловероятно, что этот прогноз будет точным, если считать, что участники проинформированы о консенсусе.

С другой стороны, предположим, что заинтересованные лица не согласны с субъективной шкалой. Широкий разброс в оценках, полученных от разных участников, позволяет сделать предположение об отсутствии точности в оценках. Иначе говоря, среднее значение при разбросе деся-

ти оценок от 1 до 5 мало что дает, если стандартное отклонение этих оценок велико.

Например, представим, что десять человек участвуют в оценке приоритета риска по шкале от 1 до 5. Двое участников оценили риск как 1, двое других — как 2, двое — 3, двое — 4 и еще двое — 5. Среднее значение равно 3, но стандартное отклонение равно 1.5. Из полученных данных очевидно, что консенсуса по данному риску не достигнуто. Среднее значение может лишь только привести к заблуждению относительно реальной ситуации. Даже если получено множество значений, близких к статистическому идеалу нормального распределения, например {1, 2, 2, 3, 3, 3, 3, 4, 4, 5}, по-прежнему есть шесть человек, не согласных с оценкой 3, а стандартное отклонение примерно равно 1.2.

Я часто замечал, что, когда возникают расхождения в оценке приоритета, заинтересованные лица представляют интересы разных групп заказчиков и пользователей. Например, продвинутые пользователи могут считать важными свойства приложения, связанные с использованием макросов, в то время как остальные пользователи никогда не обратятся к этим свойствам. Оценка числа и значения для компании определенного заказчика или групп пользователей, для которых значимо то или иное свойство приложения или которые являются предметом определенного риска, может способствовать нахождению определенного компромисса по вопросу приоритета свойств. Еще одна причина различия в оценках может заключаться в том, что некоторые заинтересованные лица не учли при рассмотрении какие-либо важные соображения. В этом случае поиску консенсуса может помочь открытая дискуссия о том, почему люди выбрали ту или иную оценку.

Для вероятности также полезным соображением является сегментирование групп заказчиков и пользователей по принципу: кто использует свойства или какому риску подвергается. Оценка вероятности должна быть единой для всех пользователей. Если какое-то предположение оказывается верным лишь для очень малой группы пользователей, но этот сегмент пользователей очень важен для долгосрочной стратегии компании, нужно увеличить значение приоритета, а не фабриковать значение вероятности.

Еще одно соображение по поводу вероятности касается технического риска. Какова вероятность внесения дефекта? Здесь ведущую роль должны сыграть заинтересованные лица, являющиеся специалистами в технологических вопросах, с учетом наличия у них соответствующих знаний и опыта. Когда люди строят новые системы на основе новых технологий, решающих задачи новыми способами, обычно число участников, которые могут похвастаться соответствующим опытом, невелико. В результате оценки технических рисков могут основываться на личном уровне оптимизма и неприятия риска. В этих случаях я пытаюсь понять, на чем основана оценка вероятности: на опыте или на догадках. Если она основана на догадках, то при оценке рисков получается определенный разброс оценок вероятности и разброс значений приоритета риска. Можно посоветовать группе управления проектом отдельно провести оценку для некоторых рисков, а затем дать ей возможность сделать запрос.

Как убедить заинтересованных лиц в принятии реальности

Когда вы впервые проводите анализ рисков качества, возможно, люди откликнутся с энтузиазмом, однако энтузиазм может иссякнуть, если станет очевидным объем затрат. Формальный подход к оценке рисков качества требует существенных затрат времени, а следовательно, и денег. Даже неформальный подход потребует затрат какого-то времени от каждого участника.

Вероятно, важно и то, что процесс анализа требует готовности каждого из заинтересованных лиц зафиксировать решения, которые не всегда совпадают с их мнением. Они должны подходить к задаче с пониманием реалий разработки, сопровождения и поставки системы с учетом ограничений по бюджету, срокам, качеству и перечню свойств системы. Ожидания систем с нулевым риском или со 100%-ным снижением рисков за счет тестирования являются нереалистичными, но любая компания способна улучшить работу по управлению рисками качества систем, которые разрабатываются, сопровождаются или поставляются.

Неудовлетворенность текущим состоянием дел — это один из мощнейших инструментов, имеющихся в распоряжении тестировщика. Считается ли, что в компании отлично обстоят дела с продажами, маркетингом и поддержкой пользователей? Если да, то организация процесса анализа рисков качества может оказаться весьма непростой. Однако чаще всего люди знают, что компания может работать лучше. Но сотрудники, не связанные с технологиями, должны также понимать, что проблемы существуют не потому, что люди в производственных подразделениях компании ленивы или умышленно плохо выполняют свои обязанности, а потому, что процессы и навыки сотрудников этих подразделений нуждаются в усовершенствовании[1]. Анализ рисков качества — это одно из направлений усовершенствования процесса, которое может оказаться полезным.

Внедрение усовершенствований

Итак, мы выбираем методику анализа рисков качества как основу для следующего шага. Вот несколько идей о том, как начать эту работу.

1. Исследуйте категории рисков качества. Начните с изучения категорий, которые подходят для вашей системы. Выше были представлены четыре книги, которые могут быть хорошими начальными пособиями. Также полезные идеи по поводу технических рисков можно найти в различных книгах по проектированию и реализации данного вида систем, например клиент-серверных систем или систем на основе web-технологий.
2. Определите ключевых заинтересованных лиц. Кто отвечает за качество и тестирование? Используя представление о структуре ком-

[1] Другие заблуждения руководства по поводу понимания ограничений возможностей процессов из-за лености персонала, в том числе классическое упражнение с черными и красными бусинами, описаны в "*Out of the Crisis*", W. E. Deming.

пании и таблицу 2.2, вы должны определить сотрудников, чье участие необходимо.

3. Уговорите заинтересованных лиц заняться управлением рисками качества. Вооружившись некоторыми идеями по поводу того, каким образом система может вести себя некорректно, а также перечнем людей, которые, как вы полагаете, заботятся о качестве системы, можно найти ключевых заинтересованных лиц и рассказать им о ваших соображениях. Помогите им понять, как управление рисками качества может способствовать тому, что будет проведено тестирование только тех частей системы, которые существенны для выпуска версии. Проинформируйте их о процессе. Укажите на то, что процесс направлен на достижение разумных компромиссов относительно различных областей риска – сроков, бюджета, перечня реализуемых свойств и качества – и различных категорий рисков качества.
4. Выстраивайте процесс анализа рисков качества. Заручившись поддержкой заинтересованных лиц и имея на руках идеи о существующих рисках, начинайте анализ рисков качества. Если в компании пока нет выстроенных процессов, попытайтесь начать с какого-либо из неформальных методов. Если вы уже используете неформальный метод, попробуйте перейти к одному из формальных методов. Чтобы обеспечить поддержку процесса, не усложняйте его.
5. Используйте оценку рисков для распределения ресурсов. В рассматриваемом примере Джамал и его коллеги распределили уровни тестирования по значениям приоритетов рисков, полученным в результате применения FMEA. Можно также применять качественные оценки, если вы видите, что это более соответствует ситуации. Покрытие определенного риска качества не является выбором между полным вариантом и отсутствием покрытия. Различные уровни тестирования – это точки на непрерывной шкале. Основная идея состоит в применении уровня риска в качестве основы для оценки времени и ресурсов, выделяемых на тестирование в любой данной области системы.

По мере внедрения всех этих новшеств при проведении тестирования вы начинаете разговаривать на языке рисков. Говорите о тестировании в терминах рисков и управления рисками. Вы обнаружите, что при обсуждении тестирования в этой манере вас хорошо понимают и принимают ваши доводы. Управление рисками – популярная тема. Менеджеры понимают управление рисками интуитивно, даже если они незнакомы с формальным введением в это понятие. Я заметил, что даже менеджеры высшего звена, которые имеют ограниченное представление о тестировании, понимают, что я имею в виду, если я говорю об управлении рисками качества системы. Это не просто звучит разумно, управление рисками качества является разумным само по себе.

ГЛАВА 3

Внутри хрустального шара: оценка предстоящей работы

После завершения анализа рисков качества, описанного в предыдущей главе, мы знаем, что *нужно* тестировать. Мы понимаем, какие риски качества системы угрожают успеху проекта, и их относительную значимость. Это понимание невозможно переоценить, поскольку без него тестирование не будет управляться качеством и может легко превратиться в безрассудную трату времени, денег и сил программиста, ввести в заблуждение руководство и доставить ему лишние заботы. Чтобы правильно ориентировать подпроект тестирования, необходимо адекватно покрывать эти риски с помощью грамотно спроектированных тестовых сценариев. Я вернусь к этой теме в главах 10 и 11, в которых рассматриваются проектирование и разработка тестовых сценариев, тестовых данных, инструментов тестирования и другие компоненты системы тестов.

Но я бы пропустил некоторые важные этапы, если бы начал уже сейчас проектировать и разрабатывать систему тестов. По завершении анализа рисков качества я знаю, какие риски угрожают качеству системы. Однако в успешных проектах рабочая группа находит баланс между качеством, свойствами системы, бюджетом и сроками. Если принять, что итоги анализа рисков качества высечены из камня, и объявить, что все рекомендованные действия группы тестирования должны быть выполнены и каждый риск качества с высоким приоритетом должен быть покрыт системой тестов, можно неприемлемо увеличить риски для свойств, бюджета и сроков. Например, если требуется слишком много времени, чтобы завершить все тестирование в проекте, проект выйдет за границы бюджета и сроков.

В этой и следующих двух главах рассматривается процесс оценки ресурсов. Процесс оценки ресурсов описывает механизм, с помощью которого тестирование встраивается в рамки сроков и бюджета проекта. В ходе процесса осуществляется переход от решения вопроса "что *нужно* тестировать" к решению вопроса "что я *могу* протестировать" в рамках проекта. В ходе этого процесса часто приходится удалять определен-

Проект (Project)

Временное предприятие, целью которого является создание или поставка некоторого уникального продукта, системы или услуги. Это предприятие состоит из некоторой последовательности задач, выполняемых группой людей при определенных ресурсных и временных ограничениях для передачи продукта, системы или услуги некоторому множеству заказчиков или пользователей. Диапазон, в котором в течение проекта могут изменяться группа, бюджет, график, требования и целевые заказчики и пользователи, может существенно отличаться[1].

Подпроект (Subproject)

Проект в рамках основного проекта, предоставляющий продукты и услуги для основного проекта. Согласно этому определению, тестирование, обсуждаемое в этой книге, обычно является подпроектом. Выпуск бухгалтерией чеков для оплаты работы участников проекта не является подпроектом, поскольку это выходит за пределы контекста проекта. Аудит соответствия документированному процессу, проводимый усилиями сотрудников отдела качества, не является ни подпроектом, ни проектом, поскольку он выполняется для нужд компании.

Подпроект тестирования (Test Subproject)

Совокупность задач, необходимых для предоставления услуг тестирования и информирования о качестве и ходе тестирования заинтересованных лиц проекта, а также содействие проектной команде в управлении рисками качества.

Оценка (Estimation)

Прогноз графика работ и затрат в проекте. В случае оценки тестирования это прогноз графика работ и затрат подпроекта тестирования.

ные активности, задачи и ресурсы из рамок подпроекта тестирования из-за их стоимости. Выбирая объект для удаления, можно опираться на иерархию рисков, выстроенную в ходе анализа. Она служит в качестве сети для отбора наиболее важных рисков качества и отбрасывания наименее важных.

Оценка ресурсов — это объемный процесс, требующий напряжения сил. Он хорошо заметен в компании, особенно если выполнен слабо, и оказывает значительное влияние на способности группы тестирования выполнять свою миссию в проекте. Это также процесс, который многие, в том числе и я, считают сложным. Довольно легко соорудить сносно выглядящий график Гантта, используя коммерческие версии программного обеспечения для управления проектами. Однако если отвлечься, можно

[1] Это упрощенный автором вариант определения из предисловия к "*The Guide to the Project Management Body of Knowledge*", 2000 Edition, www.pmi.org.

понять, что нам нужны реалистичные, работоспособные и честные оценки. Давайте изучим, как можно создавать такие оценки[1].

Процесс оценки затрат

Процесс оценки затрат, представленный в Процессе 4, разбит на 5 основных этапов, каждый из который имеет несколько подэтапов. В рамках этапов и подэтапов допускается любое число итераций. Поскольку оценка затрат — это сложный и ключевой процесс, в этой главе рассматривается первый этап, этап 2 — в главе 4, а этапы 3 — 5 описываются в главе 5.

С чисто механистической точки зрения, составление декомпозиции работ на первом этапе не будет сложной, если использовать программы для управления проектами. Инструмент управления проектом генерирует красивые графики Гантта и сетевые диаграммы, но это не означает, что их пользователь сможет понять, что будет происходить дальше. Конечно, нужно уметь пользоваться программным обеспечением для управления проектом, но в действительности необходимо уметь заглянуть в будущее.

При декомпозиции работ важно знать, когда остановиться при выделении действий, задач и подзадач. В хорошо согласованной декомпозиции работ для подпроекта тестирования каждая задача на самом нижнем уровне обладает следующими характеристиками:

1. Измеряемое состояние
2. Четко сформулированные критерии начала и завершения
3. Один или более исходящих и, возможно, входящий передаваемый материал
4. Оценка требований к ресурсам, объему выполняемых работ, их продолжительности и другим затратам
5. Короткая продолжительность
6. Независимость (после того как выполнен критерий начала) от других задач в рамках и вне подпроекта тестирования
7. Четко выделенный ответственный исполнитель, в идеале — один человек
8. Отображение выявленных в ходе анализа рисков качества на одну или более рекомендуемых работ по тестированию (причем все фазы, активности и задачи, взятые вместе, покрывают все рекомендуемые работы по тестированию)
9. Место задачи в перечне работ основано на приоритетах рисков настолько, насколько это позволяют взаимозависимости задач и ограничения по ресурсам

---

[1] Фрагменты этой и последующих двух глав впервые были опубликованы в статье "Test Estimation" в "*Software Testing and Quality Engineering*", Volume 4, Issue 6 (Nov/Dec 2002), а также в "Factors That Influence Test Estimation" на www.stickyminds.com и www.rexblackconsulting.com. Благодарю за редакторскую помощь Эстер Дайсон (Esther Dyson) и Роберта Кутре (Robert Coutre).

Процесс 4. Оценка

№ этапа	Этап	Выполнено?
1.	Работая в одиночку или совместно с приданной группой тестирования, выполнить декомпозицию и составить план работ.	☐
1.A	Разбивка проекта на фазы.	☐
1.B	Разбивка каждой из фаз на составляющие ее активности (действия).	☐
1.C	Разбивка каждой активности на задачи и подзадачи, пока каждая задача и подзадача на низшем уровне декомпозиции не будут удовлетворять критериям, перечисленным ниже.	☐
1.D	Определение зависимостей, распределение ресурсов и зависимых задач в рамках подпроекта тестирования с учетом приоритетов рисков. Документирование зависимостей, ресурсов и задач, внешних по отношению к подпроекту тестирования (т.е. требующих совместных действий).	☐
1.E	Проверка, с точки зрения здравого смысла, длительности и объема работ на уровне каждой задачи, подзадачи, активности и фазы. По возможности сопоставить профессиональные знания и интуицию с данными предыдущих проектов, измерениями в отрасли и т.д. Выявить и, если возможно, разрешить расхождения между планами-графиками подпроекта тестирования и самого проекта. Там, где это невозможно, документировать проблемы.	☐
1.F	Проверка декомпозиции работ и плана-графика совместно с сотрудниками, которые назначены ответственными за выполнение задач, с особым вниманием к выявлению любых пропущенных допущений и зависимостей.	☐
1.G	Рецензирование декомпозиции работ и плана-графика силами всех сотрудников группы тестирования при участии экспертов в проблемной области как внутри, так и вне компании.	☐
2.	Использовать декомпозицию работ и план-график для подготовки бюджета.	☐
2.A	Извлечение из декомпозиции работ полного списка ресурсов. Определение для каждого ресурса первого и последнего дня работы в проекте. В том случае, если есть ресурсы, занятые сразу в нескольких проектах, уточнить процентную загрузку каждого из таких ресурсов в каждом из проектов в течение различных периодов времени.	☐
2.B	Выявление необходимых дополнительных ресурсов для поддержки ресурсов, занятых сразу в нескольких проектах.	☐
2.C	Распределение ресурсов по категориям: персонал, командировки, инструментарий, тестовая среда и, если есть, затраты на аутсорсинг. Суммирование по периодам времени и категориям.	☐

Процесс 4 *(продолжение)*

№ этапа	Этап	Выполнено?
2.D	Проверка, с точки зрения здравого смысла, бюджета и общих сумм. По возможности сопоставить профессиональные знания и интуицию с данными предыдущих проектов, измерениями в отрасли и т.д., выявить и, если возможно, разрешить расхождения между бюджетом подпроекта тестирования и общим бюджетом проекта. Там, где это невозможно, документировать проблемы.	☐
2.E	Уточнение амортизации объектов, которые являются предметом долговременного инвестирования, документирование возможностей их повторного использования и определение периода, в течение которого планируется возместить затраты.	☐
2.F	Если это необходимо или желательно, провести анализ возврата инвестиций. Если сальдо отрицательное, проверить допущения об амортизации, сделанные на шаге 2.E, и при необходимости повторить его. Если сальдо по-прежнему отрицательное, проанализировать те компоненты декомпозиции работ, которые требуют наибольших затрат с наименьшей отдачей, сверившись с рисками качества. Документировать активности, приводящие к потере средств.	☐
2.G	Если разрешено руководством, провести обсуждение бюджета с группой тестирования. Особое внимание уделить выявлению всех пропущенных ресурсов, особенно дополнительных. При необходимости повторить этапы 2.D – 2.F.	☐
3.	Утвердить у руководства план-график и бюджет.	☐
3.A	Презентация преимуществ подпроекта тестирования.	☐
3.B	Обрисовать временные и финансовые ресурсы, необходимые для получения описанных преимуществ.	☐
3.C	Осознание и принятие мер по разрешению любых возражений по поводу сделанных оценок посредством повторения шагов 3.A и 3.B.	
3.D	Если не удается получить согласие руководства на предлагаемые план-график и бюджет, обсудить, какие области тестирования могут быть исключены, установить, какие ограничения по срокам и бюджету позволят получить поддержку руководства.	☐
4.	Повторить этапы 1 – 3, если необходимо. Уточнить график работ и бюджет с целью добиться адекватности ресурсов рамкам тестирования (возможно уточненным) и одобрения руководства.	☐
5.	Поместить принятые документы, фиксирующие план-график и бюджет, в проектную библиотеку или систему управления конфигурацией проекта. Организовать контроль изменений документов.	☐

Фаза (Phase)

Логически выделенная часть проекта или подпроекта, состоящая из совокупности действий, которые завершаются выпуском продуктов, поставляемых заказчику.

Активность, действие (Activity)

Логически выделенная часть фазы, состоящая из набора задач, выполнение которых означает получение одного существенного передаваемого объекта/документа.

Задача (Task)

Рабочие элементы активности, которые являются логически последовательными множествами действий, включающих в себя соответствующие наборы умений и навыков.

Джамал предсказывает будущее

По завершении семинара, посвященного анализу видов ошибок и их влияния, Джамал отправился на свое рабочее место и запустил приложение по управлению проектом. Он также загрузил SpeedyWriter 3.0, чтобы начать аккумулировать идеи относительно плана тестирования. Увидев перед собой два пустых шаблона, он на секунду впал в уныние, но затем принялся за работу. "Итак, – думал он, – какие фазы будут в этом проекте? Ну-ка, посмотрим: я должен подготовить план, мы собираемся написать кучу новых тестов, мне нужно попросить Кейта Ли и его группу системного администрирования настроить конфигурации некоторых новых систем в нашей тестовой лаборатории, а затем мы будем проводить комплексное и системное тестирование". Он вписал эти фазы в декомпозицию работ.

№ этапа	Этап	Выполнено?
1.A	Разбивка проекта на фазы.	☑

Фазу планирования оказалось легко конкретизировать. Джамал вместе с Дженни и Кейт Эрнандес уже выстроил контекст тестирования после совещания, посвященного запуску проекта, а также начал анализировать риски. После того как были сделаны оценки затрат, он приступил к подготовке плана тестирования. Джамал назначил себя ответственным за все задачи на фазе планирования, иначе говоря, он определил себя как ресурс. Была обнаружена только одна зависимость: участие заинтересованных лиц, не входящих в группу тестирования; это было отмечено в черновом варианте плана тестирования. На рис. 3.1 показан план-график для этой фазы.

Каталожно-карточный подход к составлению декомпозиции работ (An index-card approach to building work-breakdown-structures)

Рецензируя эту главу, главный консультант компании AmiBug, Inc. (www.amibug.com) Роберт Сабурин (Robert Sabourin) указал, что для составления декомпозиции работ он использовал каталожно-карточную методику перед началом использования инструментальных средств управления проектом или в качестве их альтернативы.

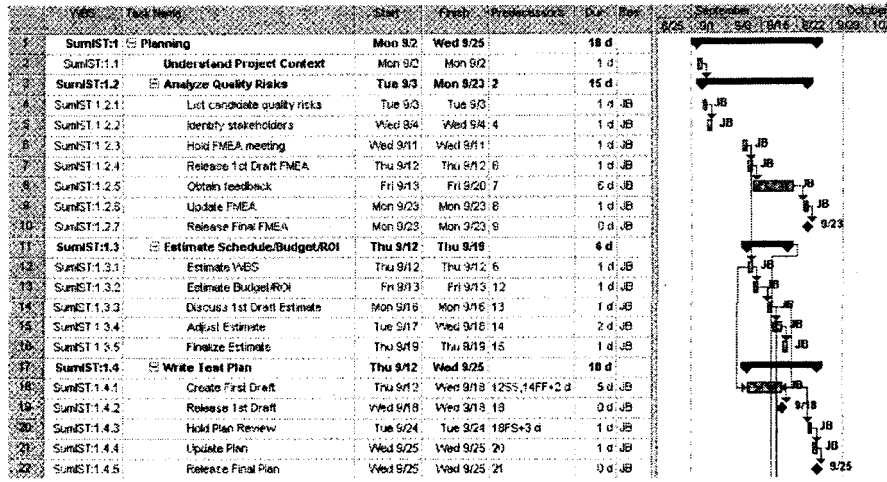
1. Возьмите стопку каталожных карточек. (Некоторые используют большие листки бумаги с клейкой полосой.)
2. Запишите по одной задаче на каждой карточке.
3. Если вы разбили задачу, замените карточку двумя или более карточками, по одной на задачу.
4. Составьте проект, разложив карточки на столе подобно большой сетевой диаграмме PERT. Сортируйте и переставляйте карточки, ищите типовые части и пропуски в схеме.
5. Как только выявлены все задачи, по одной на каждую карточку, и все карточки упорядочены, запишите на обратной стороне каждой карточки зависимости задачи.

Определение порядка выполнения, как это отмечено в п.5, может вызвать проблемы для относительно больших проектов. Вместо того чтобы пытаться завершить шаг 5 одним энергичным усилием, можно воспользоваться большой белой доской, набором маркеров и провести следующие действия.

6. Просмотрите карточки и сложите в стопку задачи, не имеющие предшественников. Разложите их с левой стороны доски.
7. Просмотрите оставшиеся карточки. Выявите те задачи, которые зависят только от задач, разложенных ранее. После того как вы это сделаете, разложите карточки на доске справа от задач (задачи), от выполнения которых они зависят, а затем нарисуйте линии, соединяющие предшественников с последующими за ними задачами.
8. Повторите шаг 7, пока не будут установлены все зависимости и все каталожные карточки не будут разложены на доске.

По завершении всех шагов можно ввести полученную информацию в систему управления проектом.

"Хм, – думал он, – когда же приступить к другим задачам?" Джамал решил начать разработку с конца, с фазы выполнения тестов. Он будет отвечать за системное тестирование. Посмотрев на план-график разработки, сделанный Дженни, он увидел, что планируемая дата поставки – 28 марта 2003 г. Судя по предыдущим проектам, его группа должна завершить тестирование неделей раньше, чтобы можно было поставить "золотую сборку" в производство. Таким образом, ориентировочная дата совещания по итогам системного тестирования – 21 марта 2003 г. К этому сроку система должна соответствовать критериям завершения тестирования, определенным в его плане тестирования.



	WBS	Task Name	Start	Finish	Predecessors	Dur	Res
1	SumIST:1	Planning	Mon 9/2	Wed 9/25		18 d	
2	SumIST:1.1	Understand Project Context	Mon 9/2	Mon 9/2		1 d	
3	SumIST:1.2	Analyze Quality Risks	Tue 9/3	Mon 9/23	2	15 d	
4	SumIST:1.2.1	List candidate quality risks	Tue 9/3	Tue 9/3		1 d	JB
5	SumIST:1.2.2	Identify stakeholders	Wed 9/4	Wed 9/4	4	1 d	JB
6	SumIST:1.2.3	Hold FMEA meeting	Wed 9/11	Wed 9/11		1 d	JB
7	SumIST:1.2.4	Release 1st Draft FMEA	Thu 9/12	Thu 9/12	6	1 d	JB
8	SumIST:1.2.5	Obtain feedback	Fri 9/13	Fri 9/20	7	6 d	JB
9	SumIST:1.2.6	Update FMEA	Mon 9/23	Mon 9/23	8	1 d	JB
10	SumIST:1.2.7	Release Final FMEA	Mon 9/23	Mon 9/23	9	0 d	JB
11	SumIST:1.3	Estimate Schedule/Budget/ROI	Thu 9/12	Thu 9/19		6 d	
12	SumIST:1.3.1	Estimate WBS	Thu 9/12	Thu 9/12	6	1 d	JB
13	SumIST:1.3.2	Estimate Budget/ROI	Fri 9/13	Fri 9/13	12	1 d	JB
14	SumIST:1.3.3	Discuss 1st Draft Estimate	Mon 9/16	Mon 9/16	13	1 d	JB
15	SumIST:1.3.4	Adjust Estimate	Tue 9/17	Wed 9/18	14	2 d	JB
16	SumIST:1.3.5	Finalize Estimate	Thu 9/19	Thu 9/19	15	1 d	JB
17	SumIST:1.4	Write Test Plan	Thu 9/12	Wed 9/25		10 d	
18	SumIST:1.4.1	Create First Draft	Thu 9/12	Wed 9/18	12SS,14FF+2 d	5 d	JB
19	SumIST:1.4.2	Release 1st Draft	Wed 9/18	Wed 9/18	18	0 d	JB
20	SumIST:1.4.3	Hold Plan Review	Tue 9/24	Tue 9/24	18FS+3 d	1 d	JB
21	SumIST:1.4.4	Update Plan	Wed 9/25	Wed 9/25	20	1 d	JB
22	SumIST:1.4.5	Release Final Plan	Wed 9/25	Wed 9/25	21	0 d	JB

Рис. 3.1. График фазы планирования

Золотая сборка, мастер-сборка, сборка общего доступа (Golden Build, Gold Master, GA Build)

Сборка или версия, поставляемая заказчикам или устанавливаемая пользователям. Тестовую сборку, предшествующую "золотой" сборке, часто называют "золотым" кандидатом (*Golden Candidate*). Если "золотой" кандидат оказывается вполне качественным, он становится "золотой сборкой". (*GA* означает *General Availability* (общий доступ), т.е. доступ всем возможным заказчикам и пользователям. *Limited Availability* (ограниченный доступ) означает бета-версию или раннюю версию для специальных заказчиков.)

Золотой" кандидат (Golden Candidate)

Плановая финальная тестовая версия, которая при условии, что не будет обнаружено ошибок, требующих исправления, становится версией систе-

Поскольку первый инкремент функциональности должен быть готов 21 октября 2002 г., в этот день должно состояться совещание, посвященное началу системного тестирования. Джамал знал, что к этому времени система должна отвечать критериям начала тестирования, определенным в плане тестирования. Также Джамалу было известно, что очередные инкременты функциональности должны поступать на тестирование с четырехнедельным интервалом. Он добавил эти рубежи и события в план-график.

Тестовый раунд (Test Pass)

Выполнение тестов, определенных для данной фазы тестирования (т.е. всего комплекта тестов).

Тестовый цикл (Test Cycle)

Выполнение некоторого подмножества тестов для одной, определенной тестовой сборки.

Джамал также отвечает за комплексное тестирование. Его необходимо завершить к 7 февраля 2003 г., когда последний инкремент с номером 5 запланирован к поставке на тестирование. Джамал взял неделю до этой даты для проверки соответствия системы критериям завершения комплексного тестирования и для проведения заключительного совещания по этому вопросу. Начало интеграции модулей запланировано у Дженни на 30 сентября 2002 г. Джамал предположил, что группа тестирования сможет получить работоспособную версию для формального комплексного тестирования 7 октября 2002 г. Он добавил эти контрольные точки и события в план-график.

В рамках этих ограничений Джамалу необходимо включить несколько раундов и в каждом раунде по нескольку циклов для каждой фазы тестирования. Это можно сделать с помощью программы составления графиков, но предыдущие попытки были неудачными для Джамала. Поскольку ему нравится работать с программами обработки таблиц для контроля выполнения тестовых сценариев, он решил воспользоваться контрольной таблицей. На основе знаний рисков качества он мог бы набросать перечень тестовых сценариев, а также оценить время их выполнения. Затем он воспользовался бы полученными цифрами для определения длительности раундов при условии, что по тем или иным причинам некоторые из тестовых сценариев были пропущены в его черновом перечне. Джамал сделал допущение, что будет разработано дополнительно 50% тестовых сценариев, помимо ранее определенных им. Он остановился на этой цифре, исходя из опыта двух своих предыдущих проектов, в которых начальная оценка тестовых сценариев была соответственно на 30 и 60% меньше итогового числа тестовых сценариев при завершении системного тестирования.

Джамал исследовал по одному риски качества, сопоставляя с ними тестовые сценарии и фиксируя информацию о покрытии тестами. Он также понимал, какие из тестовых сценариев будут выполняться без изменений, какие из них потребуют обновления, а какие должны быть написаны заново. Он стал собирать эту информацию для планирования фазы разработки тестов. За основу оценки затрат на написание и обновление тестовых сценариев он взял предыдущий опыт, просматривая планы-графики своих двух последних проектов.

Завершив первые грубые оценки, Джамал просмотрел краткий список комплектов тестов (см. рис. 3.2). Он добавил примерные трудозатраты в

SpeedyWriter Release 3.1 (w/ DMS) IST Test Case Tracking Spreadsheet

Test ID	Test Suite/Case	Phase	Est. Effort	Est. Duration	Comment
1.000	*Functionality*				
	Suite Summary	IS	62	72	
2.000	*Performance, Load, Capacity, and Volume*				
	Suite Summary	IS	10	40	
3.000	*Reliability/Stability*				
	Suite Summary	IS	4	336	
4.000	*Error Handling and Recovery*				
	Suite Summary	IS	14	14	
5.000	*Installation, Maintenance, and Operations*				
	Suite Summary	S	72	72	
6.000	*Localization*				
	Suite Summary	S	128	128	
7.000	*Security and Privacy*				
	Suite Summary	S	24	24	
8.000	*Documentation*				
	Suite Summary	S	28	28	
9.000	*Integration*				
	Suite Summary	I	48	96	

Рис. 3.2. Оценка трудозатрат и продолжительности для тестовых сценариев, выполненная Джамалом

человеко-часах на каждый тестовый сценарий, а также продолжительность в обычных часах. И то и другое нужно для последующей оценки необходимого числа сотрудников и количества тестовых сред.

Джамал провел вычисления и выяснил потребности в персонале и длительность выполнения тестового раунда в комплексном тестировании, системном тестировании и продолжительность периода, когда фазы перекрывают друг друга (см. рис. 3.3). Он исключил тестирование локализации из своих расчетов числа необходимых тестировщиков, поскольку планировал нанять внешних исполнителей для этого вида тестирования, как это было при подготовке предыдущих версий. Для периода пиковой загрузки группы тестирования Джамал предусмотрел включение двух специалистов по автоматизированному тестированию, одного специалиста по ручному тестированию и четырех младших тестировщиков. Поскольку каждые четыре недели будет повторяться комплексное тестирование, можно запланировать по одному полному четырехнедельному раунду комплексного тестирования на каждый инкремент.

Для системного тестирования Джамал запланировал по два полных раунда на каждый инкремент, позаботившись о проведении регрессионного тестирования на случай, если сборки будут подвергаться частым изменениям. (Заметим, что Джамал, помимо исходной сборки, включающей функциональность инкремента, планирует получать версии с исправлен-

	Integration Test Phase	Effort	Duration
	Total Person Hours/Hours	138	568
	Est. Person Weeks/Weeks	7	7
	Proposed ATE Staffing	1	
	Proposed MTE Staffing	1	
	Proposed Tech Staffing	0	
	Test Pass Duration (Weeks)	4	
	Test Environments Required (w/ Some Sharing)		2
	System Test Phase (excl. Localization)		
	Total Person Hours/Hours	214	586
	Est. Person Weeks/Weeks	11	7
	Proposed ATE Staffing	2	
	Proposed MTE Staffing	1	
	Proposed Tech Staffing	3	
	Test Pass Duration (Weeks)	2	
	Test Environments Required (w/ Some Sharing)		4
	Overlapping Phase Period (excl. Localization)		
	Total Person Hours/Hours	262	682
	Est. Person Weeks/Weeks	13	9
	Proposed ATE Staffing	2	
	Proposed MTE Staffing	1	
	Proposed Tech Staffing	4	
	Test Pass Duration (Weeks)	2	
	Test Environments Required (w/ Some Sharing)		5

Рис. 3.3. Расчет необходимого числа сотрудников для перекрывающихся фаз тестирования

ными дефектами в ходе тестирования каждого инкремента.) «Да, – думал он, – итерационный подход увеличивает период, когда обе фазы перекрывают друг друга. При настоящей каскадной разработке мне нужно было всего три лаборанта, и готов поспорить, что мне хватило бы двоих. Однако при условии, что два младших тестировщика, работающих по контракту, обойдутся чуть дороже чем $75 000, по всей видимости, компании выгодно их нанять, чтобы получить дополнительную гарантию успеха итерационного подхода к разработке».

№ этапа	Этап	Выполнено?
1.B	Разбивка каждой из фаз на составляющие ее активности. [заполненный checkbox]	☑
1.C	Разбивка каждой активности на задачи и подзадачи, пока каждая задача и подзадача на низшем уровне декомпозиции не будут удовлетворять нижеследующим критериям.	☑

Джамал заметил серьезную проблему с людскими ресурсами. Сначала он предполагал, что просто переведет в новый проект Эмму Мурхаус и Лин Цу Ву на роли специалистов по автоматизированному и ручному тестированию соответственно, так как они уже работали над версией 3.0

и сейчас занимаются ее сопровождением. Джамал хотел заполнить эти вакансии, используя их и еще двух опытных младших тестировщиков, которые, как он полагал, уже готовы выполнять более ответственную работу как специалисты-тестировщики. Однако только Эммы и Лин Цу недостаточно. Ему нужен еще инженер-тестировщик, поскольку, в противном случае, Лин Цу будет разрабатывать тесты до ноября, а Эмма еще дольше — до января. Это может поломать все планы по выполнению тестов, поскольку ни Эмма, ни Лин Цу не смогут руководить работой младших тестировщиков по выполнению тестов. Поэтому Джамал переработал часть плана-графика, относящуюся к разработке тестов, чтобы отразить добавление еще одного тестировщика. Он также дополнил план-график задачами по поиску и найму инженера-тестировщика и четырех младших тестировщиков, а также задачами по их обучению и вводу в проект.

Получив первую серьезную оценку затрат, Джамал быстро проверил декомпозицию работ относительно каждого из девяти критериев. Он обнаружил проблемы, относящиеся только к распределению ресурсов, оценки для которых уже были получены, но еще нужно было уточнить их с участием Эммы и Лин Цу. Воспользовавшись средствами построения графиков инструмента управления проектами и добавив вновь выявленные задачи, Джамал завершил первую часть оценки ресурсов. Еще остались проблемы с недозагрузкой и перезагрузкой ресурсов, но эти колебания присущи процессу выполнения тестов. В эти периоды у тест-менеджера есть определенная свобода в распределении нагрузки тестировщиков и в сокращении планового числа исполняемых тестов, если это необходимо.

№ этапа	Этап	Выполнено?
1.D	Определение зависимостей, распределение ресурсов и зависимых задач в рамках подпроекта тестирования с учетом приоритетов рисков. Документирование зависимостей, ресурсов и задач, внешних по отношению к подпроекту тестирования (т.е. требующих совместных действий).	☑

Кроме того, некоторые из перекрывающихся работ являются результатом неточностей при проведении оценок. Например, Эмма перезагружена в начальные периоды системного тестирования, поскольку она отвечает и за выполнение, и за обновление скриптов для тестирования производительности и нагрузки. Однако, поскольку скрипты еще не будут готовы к работе, ей придется заниматься только их обновлением. Поскольку Джамал помнил об этом при составлении плана-графика выполнения тестов, он понимал, что это не проблема при условии, что все заявленные тесты будут готовы к выполнению в порядке приоритетов рисков.

Чтобы проверить эти два условия, Джамал вновь обратился к документу с анализом видов ошибок и их влияния. Он проверил полноту покрытия каждого риска, как это требуется в поле "Рекомендуемые действия". Он также проверил, что задачи выстроены в соответствии с порядком приоритетов рисков там, где этот порядок может повлиять на возможность выполнения тестов на определенных фазах. Пока было не совсем ясно, какие из свойств появятся в каких инкрементах и какие ошибки бу-

дут исправлены. Однако Джамал подготовил ряд гипотез, как это должны делать все квалифицированные менеджеры.

Джамал зафиксировал покрытие каждого риска качества тестовым сценарием на специальном листе в контрольной таблице. На основании рекомендаций, данных в таблице метода анализа видов ошибок и их влияния, он принял решение об использовании модифицированной шкалы, применяемой в методе определения качества функциональности (см. главу 2):

- 0 – тестовый сценарий не покрывает данный риск качества.
- 1 – тестовый сценарий обеспечивает вероятностное покрытие данного риска качества.
- 3 – тестовый сценарий обеспечивает сбалансированное покрытие данного риска качества.
- 9 – тестовый сценарий обеспечивает подробное покрытие данного риска качества.

Помимо того, что данная информация дает уверенность в том, что ничего не пропущено, она имеет большое значение для подготовки отчета. Естественно, поскольку тестовые сценарии еще не написаны, Джамал снова делает оценки. Теперь он оценивает, какие области, по его мнению, *будут* покрывать тесты. В то же время он проектирует тестовые сценарии с минимальной детализацией, исходя из своих представлений о том, что они *должны* покрывать. Данные оценки и результаты проектирования тестов необходимо будет проверить в ходе разработки системы тестов.

В ходе отслеживания покрытия рисков Джамал обнаружил, что при проведении анализа видов ошибок и их влияния были пропущены два риска, связанных с интерфейсами. Помимо того, что клиент обращается к web-серверу и web-сервер обращается к серверу баз данных, web-сервер обращается к серверу приложений, а сервер приложений – к серверу управления иерархическим хранилищем. Джамал думал, что соответствующие риски имеются в таблице анализа видов ошибок и их влияния, и отправил электронное сообщение заинтересованным лицам с просьбой подтвердить свои подозрения.

Чтобы проверить свою работу, Джамал просмотрел пару планов-графиков и других документов из предыдущих проектов, отслеживающих выполненную работу. Его оценки вписывались в представления о том, что происходило в предыдущих проектах, по крайней мере, на верхнем уровне затрат на разработку автоматизированных тестовых сценариев и сценариев для ручного тестирования, а также человеко-часов на выполнение тестового раунда. Поскольку продолжительные фазы тестирования и их перекрытие являются неотъемлемой частью итерационного подхода к разработке систем, общая продолжительность тестирования в проекте и количество персонала оказалось существенно выше, примерно вдвое.

Джамал также обратился к плану-графику разработки, подготовленному Дженни, и просмотрел распределение ресурсов. У нее было два проектировщика системы, два специалиста по пользовательским интерфейсам из группы Бобби, пять программистов на Java, один эксперт по SQL и базам данных и один эксперт по web-технологиям и приложению.

Это означало соотношение тестировщиков и разработчиков 7 к 11. Джамалу известно, что при разработке коммерческих приложений это

соотношение обычно должно соответствовать среднему значению по отрасли. В то же время он понимал, что эти цифры существенно зависят от конкретной ситуации.

Джамал уже почти готов перейти к следующему этапу — уточнению оценок с привлечением группы тестирования и других экспертов-тестировщиков компании Software Cafeteria. Во-первых, хотя он уже просмотрел графики Дженни и Кейт и другие сведения на их интранет-сайтах, необходимо взглянуть еще раз, не забыл ли он что-то, не получил ли он график подпроекта тестирования, который не соответствует потребностям проекта SpeedyWriter. В целом план выглядел неплохим, однако Джамалу все еще не было ясно, как альфа- и бета-версии для группы маркетинга могут повлиять на виды тестирования, за которые он отвечает. Поскольку еще не существовало перечня ожидаемых свойств системы, он отправил сообщение Дженни и Кейт с просьбой организовать встречу, чтобы прояснить эту ситуацию.

№ этапа	Этап	Выполнено?
1.E	Проверка, с точки зрения здравого смысла, длительности и объема работ на уровне каждой задачи, подзадачи, активности и фазы. По возможности сопоставить профессиональные знания и интуицию с данными предыдущих проектов, измерениями в отрасли и т.д. Выявить и, если возможно, разрешить расхождения между планами-графиками подпроекта тестирования и самого проекта. Там, где это невозможно, документировать проблемы.	☑

Затем Джамал пригласил Лин Цу и Эмму в свой кабинет для обсуждения распределения задач в предстоящем проекте "Суматра". Он представил свой план-график и рассказал о методах, которые он использовал для получения оценок фаз выполнения тестов. Затем он описал все задачи и распределение ответственности за их выполнение и спросил коллег, есть ли у них замечания.

"Итак, — сказала Эмма, — я думаю, ты слишком оптимистично оцениваешь наши возможности по поиску, найму и обучению в течение трех недель еще одного специалиста по автоматизированному тестированию. Ведь если подходящий кандидат уже имеет работу, то он появляется и принимает предложение в течение одной недели после опубликования объявления, а затем делает стандартное двухнедельное уведомление своему нынешнему работодателю. Не думаю, что сейчас есть большое количество подходящих людей, которые неподалеку болтаются без дела".

"Ты права, — согласился Джамал. — Возможно, потребуется больше времени, от четырех до шести недель при энергичных поисках. Мы могли бы также рассмотреть возможность найма на временную работу по контракту"[1].

[1] В Соединенных Штатах, Канаде и некоторых странах Европы агентства по подбору временного персонала и некоторые независимые компании, занимающиеся тестированием, предоставляют сотрудников на короткие контракты (60–90 дней). По завершении контракта сотрудник может быть переведен в штат компании. См. Ruhl "*The Computer Consultant Guide*".

"С учетом планов по выпуску версий продукта в следующем году, — ответила Эмма, — нам нужен в штат еще один специалист по автоматизированному тестированию. Поиск контрактника, вероятно, вполне приемлем, только если он рассчитывает на перевод в штат в случае, если ему понравится работать у нас".

"Или ей", — вставил Джамал.

"Что?" — переспросила Эмма.

"Ты сказала, если ему понравится работа. Не забывай, что двое из лучших тестировщиков в нашей группе (это ты с Лин Цу) — женщины. Вероятно, новым специалистом по автоматизированному тестированию будет еще одна женщина, — ответил Джамал с улыбкой. — Но если серьезно, я понял твою мысль".

Далее и Эмма, и Лин Цу выразили беспокойство по поводу большой загрузки в ходе двух первых месяцев проекта, в период разработки системы тестов, проведения комплексного тестирования, и даже когда кончится ноябрь, проведение системного тестирования будет в полном разгаре. "Смотри, — Лин Цу указала на график загрузки ресурсов, — я имею по две задачи в течение почти трех месяцев!")

"Да, я знаю, как это выглядит, — согласился Джамал, — а ты знаешь, как это работает. Ты разрабатываешь тесты, поэтому тебе нужно их тестировать. Это препятствует выполнению всех запланированных тестов, но нам и не нужны проблемы, связанные с обнаружением большого числа ошибок в начале тестирования. Я просто хочу быть уверен в том, что мы разрабатываем и выполняем тесты в соответствии с порядком, в котором выстроены риски, по крайней мере, как нам позволяет это делать состав очередного инкремента".

Некоторое время шло обсуждение. Джамал согласился передать одного или двух младших тестировщиков из проектов по сопровождению версий пакета OfficeArrow в перерывах работ по этим версиям для усиления группы проекта "Суматра". По мнению Джамала, хотя отдел тестирования невелик (в нем всего 14 сотрудников), он сможет привлекать людей на день-два для проведения тестирования, слегка урезая тестирование сопровождения без сужения покрытия.

№ этапа	Этап	Выполнено?
1.F	Проверка декомпозиции работ и плана-графика совместно с сотрудниками, которые назначены ответственными за выполнение задач, с особым вниманием к выявлению любых пропущенных допущений и зависимостей.	☑

Наконец Джамал распечатал копии подготовленных документов с планом-графиком, таблицей контроля тестов и т.д. для группы тестирования и для двух человек из группы Дженни, которые будут участвовать в подготовке инструментов для модульного и компонентного тестирования. Раздавая копии документов, он попросил каждого не только отрецензировать его оценки, но и дать лучшую, худшую и ожидаемую оценки для задач, за выполнение которых они непосредственно отвечают, и для задач, опыт решения или понимание которых у них имеется.

Собрав на следующее утро запрошенную информацию, Джамал проверил свои оценки. Подавляющая часть оценок совпала с ожидаемыми оценками других тестировщиков. Просматривая комментарии каждого, он просил аргументировать лучшие и худшие оценки. Один из экспертов из группы Дженни удивил его плохими оценками затрат на разработку тестов для комплексного тестирования.

"Знаешь, – сказал ему Джек Нильсон, – возможно, ты прав; потребуется примерно семь недель, чтобы собрать вместе и проверить все, что нужно для комплексного тестирования. И тесты для модульного и компонентного тестирования, за которые отвечаем мы, программисты, как раз точно впишутся в эти рамки. Я уже видел, что так бывает".

"Однако, – продолжал он, – я также видел, как ход процесса комплексного тестирования серьезно нарушался по множеству причин, не последняя из которых – это немыслимое сокращение периода комплексного тестирования на стороне группы разработки. Понимаешь, что я имею в виду? Обычно происходит недооценка ресурсов на кодирование, то есть чем-то приходится жертвовать. Одна из возможных жертв – это комплексное тестирование. То же происходит с модульным и компонентным тестированием. Если это так, группе тестирования приходится делать много дополнительной работы, принимая во внимание, что все ошибки, которые мы не нашли, проявляются то тут, то там в ходе системного тестирования. Не утверждаю, что это обязательно случится. Я даже не говорю о том, что слышал, как кто-то предположил, что время, выделенное на проведение комплексного тестирования, пока скрыто в плане-графике под разными названиями. Я просто хочу сказать, что тебе, вероятно, потребуется план на случай возникновения этой ерунды".

Джамал согласился с доводами Джека. Он добавил соответствующее примечание в план тестирования, чтобы включить его позднее в качестве возможного риска.

№ этапа	Этап	Выполнено?
1.G	Рецензирование декомпозиции работ и плана-графика силами всех сотрудников группы тестирования при участии экспертов в проблемной области как внутри, так и вне компании.	☑

Теперь Джамал почувствовал, что план-график выглядит неплохо. Но каковы будут затраты на его осуществление? Еще важнее то, как можно продемонстрировать ценность этих затрат? Перед тем как заняться подготовкой примерного бюджета, Джамал решил выделить немного времени на изучение вопроса, как тестирование может вернуть затраты на его проведение.

№ этапа	Этап	Выполнено?
1.	Работая в одиночку или совместно с приданной группой тестирования, выполнить декомпозицию работ и составить примерный план-график.	☑

Использование эмпирических приемов для грубой оценки

В нашем примере Джамал составил декомпозицию работ на основе результатов анализа рисков качества. В проектах разработки менеджеры разработки составляют декомпозицию работ на основе требований и архитектуры. Эти методы называют восходящими или "*снизу-вверх*". Оценки "*снизу-вверх*" можно проверить с помощью нисходящих правил. В компаниях с достаточной степенью формализации процессов для нисходящих оценок используют такие метрики, как балльная функциональная оценка (function points) и ожидаемое число строк кода (projected lines of code). Однако в большинстве компаний тест-менеджеры вынуждены опираться на простые эмпирические правила, сравнивающие относительные размеры различных фаз проекта разработки. Кейперс Джоунз (Capers Jones), Мартин Пол (Martin Pol), Тим Коомен (Tim Koomen) и Джефф Рашка (Jeff Rashka) опубликовали все полезные эмпирические правила. Тем не менее, как осторожно заметил Джоунз, эмпирические правила неизбежно являются неточными и должны применяться совместно с декомпозицией работ, если требуются достаточно точные оценки затрат в проекте[1].

Один из наиболее употребляемых показателей оценки затрат на тестирование — это отношение числа тестировщиков к числу разработчиков. Кроме только что упомянутых работ, могу порекомендовать две статьи об использовании этого метода, которые можно найти в Интернете. Это великолепная работа Джоанны Ротман (Johanna Rothman) "It Depends: Deciding on the Correct Ratio of Developers to Testers". В ней Ротман представляет подход к выбору правильного отношения числа тестировщиков к числу программистов на основе анализа рисков и приводит три примера. Она исследует области риска, относящиеся к продукту, проекту, его процессам и его рабочей группе. Риски продукта — это большой объем и повышенная сложность. К рискам проекта и процессов относятся короткие сроки, отсутствие приемлемого процесса разработки, тестирования и управления проектом и потребность в высоком качестве. Рисками, связанными с рабочей группой, являются неопытные тестировщики, разработчики, маркетологи, менеджеры проектов и другие ключевые участники проекта. В статье приводится описание метода анализа представленных рисков с целью определения, нужно ли увеличивать отношение числа тестировщиков к числу программистов для снижения рисков каче-

---

[1] Эмпирические правила оценки проектов разработки, включая фазу тестирования, собранные Джоунзом, можно найти в его книге "*Estimating Software Costs*", в главе 11. Помимо декомпозиции работ, Джоунз рекомендует использовать инструменты автоматизированной оценки, продажей одного из которых занимается его компания. Однако, по моим наблюдениям, редко какие компании, разрабатывающие и сопровождающие программное обеспечение, используют эти инструменты. Таким образом, декомпозиция работ, составленная с помощью инструмента управления проектом, является наиболее точным неавтоматизированным методом, доступным типичному тестировщику и тест-менеджеру. В статье Рашки есть неплохое введение на шести страницах, однако отсутствуют некоторые важные предостережения, упоминаемые в этом разделе и в работе Джоунза. Наконец, Пол и Коомен включили описание некоторых эмпирических правил оценки затрат в главу 7 книги "*Test Process Improvement*".

ства системы. Этот подход отлично уживается с методиками, представленными в книге. Поскольку читатель уже думает о рисках качества, он может включить в свои размышления и эти более широкие категории.

Помимо работы Ротман, можно прочитать глубокую статью Кетлин Айберли (Kathleen A. Iberle) и Сюзан Бартлет (Susan Bartlett) "Estimating Tester to Developer Ratios (or Not)". Авторы анализируют факторы, влияющие на точность оценок, сделанных на основе отношения числа тестировщиков к числу разработчиков. Айберли переслала мне по электронной почте ключевые идеи, зафиксированные в следующих правилах:

1. Комбинация противоречивых систем измерений и существующие различия в значениях отношений приводят к ошибкам в значениях отношений на порядок. Следовательно, сравнение отношения, которое имеется в вашем проекте, с любым другим бесполезно. Влияние системы измерений невозможно отделить от существующих различий в значениях отношений.
2. Если предлагаемое значение отношения заметно отличается от средних колебаний значения отношения в отрасли, сопоставление может оказаться полезным. Например, предложение о том, что весь отдел должен работать при отношении 1 к 30 во всех проектах, легко парировать ссылкой на существующие в отрасли значения отношения.
3. Значения отношения, которые были в других проектах с вашим участием, можно применять при выполнении следующих условий:
   A. Используется непротиворечивая система измерений.
   B. (Почти) отсутствует информация о будущей функциональности системы (в противном случае оценки затрат, например, на основе декомпозиции работ будут значительно более точными).
   C. Полученные отношения корректируются с учетом информации о различиях проектов.
   D. Ошибка в 25–50% считается приемлемой.

Если необходимо сделать первоначальную оценку затрат на основе отношения числа тестировщиков к числу разработчиков, обе статьи предоставляют достаточно информации, чтобы разумно выполнить поставленную задачу[1]. Однако нерационально ограничиваться одним подходом. Комбинируя эмпирические правила Джоунса, Пола, Коомена и Рашки с методами на основе анализа рисков, предлагаемыми Ротман, и моделью факторов, влияющих на отношение, разработанной Айберли и Бартлет, можно уточнить грубые оценки. Выполняя различные оценки с помощью разнообразных эмпирических правил, в том числе получая средние оценки, можно повысить точность окончательных оценок. Различие между моделями позволяет получить наилучшие и наихудшие оценки. Тем не менее следует помнить, что при необходимости получить более точные оценки, по всей вероятности, наилучшим вариантом является использование декомпозиции работ.

---

[1] Статья Ротман доступна на www.jrothman.com, статья Айберли и Бартлет есть в "*Proceedings of the STAR West Conference 2001*" и на домашней страничке Айберли www.kiberle.com. Работая над книгой, мне посчастливилось обсудить эту тему со всеми авторами.

Объем, продолжительность и зависимости работ

Поскольку мы рассматривали технологию декомпозиции работ проекта, было упомянуто, что для грубых оценок или проверки оценок, полученных с помощью декомпозиции работ, сгодятся эмпирические правила. Пусть мы составляем декомпозицию работ. Каков типичный перечень задач на каждой из фаз проекта, и сколько времени необходимо для их выполнения? В следующих разделах для обсуждения этой темы мы рассмотрим все основные фазы проекта на примере графика Гантта для проекта "Суматра".

Планирование

Как показано на рис. 3.1, Джамал разбил фазу планирования подпроекта тестирования на четыре основные активности. Первая — знакомство с проектом — была проведена в форме стартового совещания и последовавших за ним обсуждений с менеджером продукта Кейт Эрнандес и менеджером разработки Дженни Кауфман. Это заняло один день в рассматриваемом примере, поскольку Джамал уже некоторое время проработал в Software Cafeteria, а продукт "Суматра" является инкрементным расширением существующей системы, которую группа Джамала уже тестировала.

Как уже обсуждалось в главе 1, ряд факторов могут сделать знакомство с проектом более сложным. Например, если в проект вводится новый сотрудник компании, если компания недавно пережила сокращение штатов или реструктуризацию, если система новая или уникальная, если планируется использовать новые технологии и процессы, если для проекта или его подпроекта тестирования необходима значительная поддержка со стороны лиц с разнообразной квалификацией, знакомство с проектом может занять несколько дней и даже недель.

Следующая активность на фазе планирования — это анализ рисков качества. Джамал разбил этот этап на семь задач. Определение кандидатов в риски качества, как упоминалось в главе 2, состоит из исследования различных источников и выбора основных категорий рисков качества. Активность также включает в себя изучение метода анализа видов ошибок и их влияния. Это не обязательно займет целый рабочий день Джамала, но продолжительность задачи будет показана как один рабочий день. Следующая задача Джамала — определение ключевых заинтересованных лиц. Как и знакомство с проектом, задача упрощается за счет достаточно длительной работы Джамала в компании. Факторы, упомянутые выше, могут сделать определение ключевых заинтересованных лиц и получение их согласия на участие в обсуждении более долгим, чем один потраченный Джамалом день.

Как отмечалось в главе 2, само совещание по анализу видов ошибок и их влияния может проводиться в течение целого рабочего дня за пределами компании. Если метод анализа видов ошибок и их влияния применяется для анализа рисков качества небольшой или средней по размеру системы, один рабочий день — это разумная оценка затрат. Более сложная система или анализ низкоуровневых проблем архитектуры и реализации могут потребовать больше времени. В одном из проектов с моим участием

три тестировщика потратили более двух недель на проведение подробного анализа на основе детальной архитектуры для пяти-шести совместно работающих подсистем. Если используется неформальная методика, анализ может быть завершен за один день при условии, что всех участников удалось собрать вместе. Если нужно опрашивать каждого по отдельности, а затем искать пересечения в перечне рисков, вероятно, потребуется провести более одного цикла черновик-отзывы-обновление-утверждение, как это было учтено в плане-графике Джамала.

В случае с графиком Джамала цикл черновик-отзывы-обновление-утверждение состоит прежде всего из документирования результатов анализа в форме анализа видов ошибок и их влияния, полученных на выездном совещании. Это может потребовать от одного часа до одного рабочего дня в зависимости от объема собранной информации. В случае если проводились интервью с каждым из участников по отдельности, наверняка будут иметь место конфликтующие точки зрения, которые необходимо или привести к консенсусу или, по крайней мере, зафиксировать документально. Джамал выделил неделю на ожидание отзывов на черновик итогового документа с анализом видов ошибок и их влияния, но не потому, что на подготовку отзывов каждый из ключевых заинтересованных лиц потратит пять полных рабочих дней, а потому, что он вынужден ожидать, пока занятые люди найдут примерно по часу в своих рабочих графиках, чтоб выполнить эту работу. По моему мнению, выделение одной недели на подготовку отзывов вполне разумно в большинстве компании. Может быть, в вашей компании это не так? Подумайте об этом. Быстро ли отвечают сотрудники компании на электронные сообщения и входящие запросы, сразу ли прочитывают почту или есть ключевые заинтересованные лица, которые очень заняты и обычно не отвечают, пока не получат неоднократные напоминания?

Джамал отвел один день на обновление документа после получения отзывов, и в этот же день он предполагает выпустить итоговую версию документа. Снова отметим, что при использовании персонального опроса, а не группового совещания, обновление документа требует больше времени и приходится готовить не одну промежуточную версию документа. Вот почему я отдаю предпочтение совещанию с участием заинтересованных лиц для достижения согласованного мнения по рискам качества и их приоритетам. Процесс опроса обычно затягивается. Поскольку процесс тестирования, который я выстраиваю, как правило, основан на результатах анализа рисков, я избегаю делать какую-либо работу до тех пор, пока не получу надежных результатов анализа качества.

Заметим, что Джамал решил приступить к оценке затрат и планированию сразу по завершении выездного совещания по анализу видов ошибок и их влияния. Он предположил, что на совещании удастся разрешить основные противоречия и останется сделать лишь незначительные уточнения, поэтому можно относительно спокойно начинать оценку и планирование. Он выделил по одному дню на составление декомпозиции работ и подготовку бюджета. Это обоснованная оценка для опытного менеджера, выполняющего знакомые задачи совместно с небольшой группой опытных специалистов. Однако если вы беретесь за новые задачи, скажем, это первый ваш проект с автоматизацией тестирования, времени

потребуется больше. Необходимо дать сотрудникам, ответственным за выполнение задач, больше чем несколько часов на раздумья о том, сколько времени потребуется на выполнение новых задач. Когда я берусь за новую задачу, я выделяю время на ее обсуждение с опытными людьми, на прототипирование, чтение книг и статей для того, чтобы изучить процесс. Думаю, вполне обоснованно в этих условиях можно выделить на оценку затрат пару недель. Кроме того, если вы впервые взялись за составление декомпозиции работ и подготовку бюджета, вам потребуется дополнительное время на изучение методов (т.е. освоение используемых в компании программных средств управления проектом) и на консолидацию идей для подготовки хорошего плана-графика. Когда я впервые готовил оценку затрат для проекта с участием трех или четырех человек продолжительностью примерно в один месяц, я потратил на это целый день. Оценка затрат теперь, после 15 лет практики, является моей второй натурой. Но эта работа определенно не может осуществляться на основе интуиции менеджера-новичка, да и растеряться от сложностей инструментария тоже довольно легко.

Джамал запланировал четыре дня на согласование окончательной оценки и на получение одобрения у руководства проекта. Он будет показывать свои оценки своему менеджеру, примет замечания, внесет изменения на основе полученных замечаний и затем представит итоговую версию документа на утверждение менеджеру. Этот процесс рассматривается в следующих двух главах, и читатель увидит, как выполняются задачи. Четыре дня достаточно, если вы уверены, что представляемые оценки не будут оспорены. Однако в случае, если существует большой разрыв между тем, что согласно анализу рисков *должно быть* протестировано, и тем, что в рамках существующих сроков и бюджета *может* быть протестировано, по всей видимости, потребуется некоторое время для разрешения этого противоречия. В такой ситуации я как тест-менеджер должен помочь руководству проекта принять взвешенные решения по поводу рисков, связанных с урезанием покрытия системы тестами. В таких случаях продолжительность выполнения задач, связанных с переговорами и утверждением решений, оценивается в неделях.

В дополнение к подготовке примерного графика и бюджета Джамал включил еще одну активность — планирование. Эта активность и получаемый в результате план тестирования рассматриваются в главах 6 и 7. Джамал предполагает потратить неделю на написание плана, причем эта задача выполняется параллельно с оценкой затрат. Джамал намеревается объяснить логику, которая скрыта в задачах, определенных в декомпозиции работ. Как только будет готов черновик плана, он отправит его на рецензирование заинтересованным лицам на неделю. Вновь повторюсь, что основанием для этой оценки является стиль работы заинтересованных лиц. Один цикл рецензирование-обновление-выпуск, запланированный Джамалом на два дня, также зависит от графика работы заинтересованных лиц и от их мнения о документе. Если план сложный, возможно более длительное согласование. Если запланированы новые и незнакомые задачи, им уделяется больше внимания.

Всего Джамал выделил три с половиной недели на завершение всех задач по планированию. Некоторая часть этого времени выделяется на

ожидание отзывов. Джамал потратит примерно 140 человеко-часов без учета периодов ожиданий, поскольку к менеджерам предъявляются требования одновременной работы над несколькими проектами. Мой опыт показывает, что в большинстве проектов необходимо учитывать эти присущие природе проектной работы задержки и переключения внимания с одного проекта на другой. В некоторых случаях, когда можно сосредоточиться на работе в одном проекте, а планирование является внутренним делом рабочей группы, я готовлю план в течение дня, а то и нескольких часов. Для более сложных проектов на планирование могут уйти месяцы.

Подбор персонала

На рис. 3.4 показан примерный план-график выделения персонала. Заметим, что одна задача, получение требований к подбору и найму, зависит от задачи на фазе планирования. Джамал должен получить одобрение бюджета и плана-графика прежде, чем можно будет ожидать разрешения на подбор людей. В компании, где вы работаете, вероятно, подбор персонала происходит иначе. В некоторых компаниях требуется получить окончательное одобрение оценок затрат. Я также работал с заказчиками, у которых до начала работы по поиску и найму специалистов должны пройти всесторонние обсуждения контрактов. Это усложняет и удлиняет фазу подбора персонала.

Джамал запланировал три недели на поиск и наем специалиста по автоматизированному тестированию и пять недель на поиск младших тестировщиков. В рассматриваемом примере Эмма указала на то, что это слишком оптимистичные оценки, и Джамал согласился. В периоды экономического спада трехнедельный срок для найма сотрудника вполне реален. Это также реальное время для поиска подходящего контрактника. В конце 90-х во время бума Интернет-технологий (dot-com boom) найти за три недели квалифицированного тестировщика со специальными знаниями было нереально в большинстве отраслей высоких технологий. Если в компании есть отдел по работе с персоналом, его сотрудники должны оказать помощь в ознакомлении с фазой подбора персонала и в более точной оценке затрат на нее в проекте.

В небольших и вновь открывающихся компаниях обычно нет отдела по работе с персоналом. В этом случае можно обратиться за помощью в оценке задач по подбору персонала к внешним организациям. При наличии четкого описания должностных обязанностей (этой теме посвящена глава 8) местные компании по подбору и найму персонала могут сказать, сколько времени потребуется для заполнения этих вакансий. Конечно,

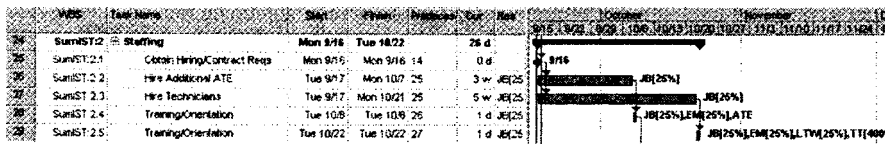


Рис. 3.4. Фаза подбора персонала подпроекта тестирования проекта "Суматра")

эти компании ожидают, что вы заплатите им в случае успешного поиска кандидата для вашей компании. Не исключено, что оценку времени, необходимого на заполнение вакансии, они делают бесплатно, но это совсем не означает, что вам удастся найти подходящего человека за указанное время. Рекрутинг-онлайн и web-сайты по подбору персонала могут предоставить некоторые идеи, как оценить время, необходимое на заполнение вакансии, но эти идеи должны применяться с учетом региона, что может оказать серьезное влияние на оценки. Ваши коллеги-менеджеры также могут поделиться своим опытом по подбору персонала, но время на поиск программиста со знанием C++ или XML может сильно отличаться от времени, необходимого для заполнения вакансии тестировщика, владеющего конкретным инструментом автоматизации тестирования. Если успех проекта в значительной степени зависит от скорейшего заполнения определенной вакансии, премия рекрутинговой компании может оказаться отличным вложением средств.

Последние две задачи этой фазы — обучение и ввод в проект. Джамал совместно с отделом по работе с персоналом компании запланировал однодневное обучение для новых сотрудников. Это обычная практика в некоторых компаниях, где я работал, хотя я также слышал, что период обучения может длиться до трех месяцев. В других компаниях в качестве метода обучения применяют классический подход: бросают в воду и смотрят, выплывет или нет. (Эта тема будет рассматриваться в главах 8 и 9.)

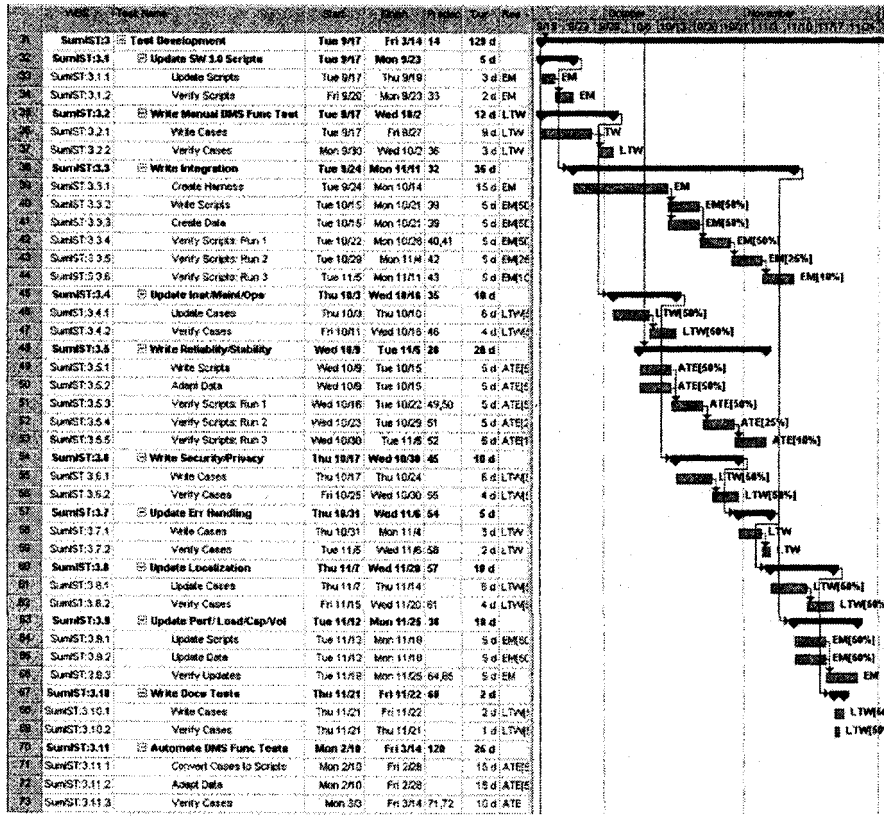
На фазу подбора персонала Джамал выделил чуть более пяти недель. Как и для фазы планирования, этот период не трансформируется в 200 часов загрузки для каждого из участников: Джамала, Эммы и Лин Цу. Деятельность, требующая больших временных затрат, — это интервьюирование кандидатов, но тщательный отбор кандидатов, как это показано в главе 8, может сократить число интервью, в которых вам придется участвовать. Обычно моя группа тестирования тратит примерно два человеко-дня на проведение одного интервью и обычно встречается не более чем с пятью кандидатами, чтобы заполнить любую вакансию.

Разработка тестов

Наиболее продолжительная фаза в подпроекте тестирования проекта "Суматра") — это разработка тестов. Очень часто применяются методы как ручной, так и автоматизированной разработки скриптов, поэтому заранее должно быть выделено достаточно времени для разработки тестов. На рис. 3.5 показана фаза разработки тестов.

Как и в случае фазы подбора персонала, Джамал установил наличие следующей зависимости: разработка тестов не может начаться, пока он не получит предварительного одобрения оценок затрат, возможно, с некоторыми последующими изменениями. Я предпочитаю не начинать разработку тестов, пока не выясню, какие средства тестирования: тестовые данные, сценарии, инструменты, скрипты, конфигурации тестовой среды,- необходимо разработать для последующего выполнения тестов.

Зависимости между активностями разработки показывают, когда Джамал предпочитает завершить выполнение задач этих этапов. В ряде случаев его предпочтения основываются на приоритетах рисков, покрываемых



	WBS	Task Name	Start	Finish	Pr	Dur	Res
31	SumIST:3	Test Development	Tue 9/17	Fri 3/14	14	128 d	
32	SumIST:3.1	Update SW 3.0 Scripts	Tue 9/17	Mon 9/23		5 d	
33	SumIST:3.1.1	Update Scripts	Tue 9/17	Thu 9/19		3 d	EM
34	SumIST:3.1.2	Verify Scripts	Fri 9/20	Mon 9/23	33	2 d	EM
35	SumIST:3.2	Write Manual DMS Func Test	Tue 9/17	Wed 10/2		12 d	LTW
36	SumIST:3.2.1	Write Cases	Tue 9/17	Fri 9/27		9 d	LTW
37	SumIST:3.2.2	Verify Cases	Mon 9/30	Wed 10/2	36	3 d	LTW
38	SumIST:3.3	Write Integration	Tue 9/24	Mon 11/11	32	36 d	
39	SumIST:3.3.1	Create Harness	Tue 9/24	Mon 10/14		15 d	EM
40	SumIST:3.3.2	Write Scripts	Tue 10/15	Mon 10/21	39	5 d	EM[50
41	SumIST:3.3.3	Create Data	Tue 10/15	Mon 10/21	39	5 d	EM[50
42	SumIST:3.3.4	Verify Scripts: Run 1	Tue 10/22	Mon 10/28	40,41	5 d	EM[50
43	SumIST:3.3.5	Verify Scripts: Run 2	Tue 10/29	Mon 11/4	42	5 d	EM[25
44	SumIST:3.3.6	Verify Scripts: Run 3	Tue 11/5	Mon 11/11	43	5 d	EM[10
45	SumIST:3.4	Update Inst/Maint/Ope	Thu 10/3	Wed 10/16	35	10 d	
46	SumIST:3.4.1	Update Cases	Thu 10/3	Thu 10/10		6 d	LTW[5
47	SumIST:3.4.2	Verify Cases	Fri 10/11	Wed 10/16	46	4 d	LTW[5
48	SumIST:3.5	Write Reliability/Stability	Wed 10/9	Tue 11/5	28	28 d	
49	SumIST:3.5.1	Write Scripts	Wed 10/9	Tue 10/15		5 d	ATE[5
50	SumIST:3.5.2	Adapt Data	Wed 10/9	Tue 10/15		5 d	ATE[5
51	SumIST:3.5.3	Verify Scripts: Run 1	Wed 10/16	Tue 10/22	49,50	5 d	ATE[5
52	SumIST:3.5.4	Verify Scripts: Run 2	Wed 10/23	Tue 10/29	51	5 d	ATE[2
53	SumIST:3.5.5	Verify Scripts: Run 3	Wed 10/30	Tue 11/5	52	5 d	ATE[1
54	SumIST:3.6	Write Security/Privacy	Thu 10/17	Wed 10/30	45	10 d	
55	SumIST:3.6.1	Write Cases	Thu 10/17	Thu 10/24		6 d	LTW[
56	SumIST:3.6.2	Verify Cases	Fri 10/25	Wed 10/30	55	4 d	LTW[
57	SumIST:3.7	Update Err Handling	Thu 10/31	Wed 11/6	54	5 d	
58	SumIST:3.7.1	Write Cases	Thu 10/31	Mon 11/4		3 d	LTW
59	SumIST:3.7.2	Verify Cases	Tue 11/5	Wed 11/6	58	2 d	LTW
60	SumIST:3.8	Update Localization	Thu 11/7	Wed 11/20	57	10 d	
61	SumIST:3.8.1	Update Cases	Thu 11/7	Thu 11/14		6 d	LTW[
62	SumIST:3.8.2	Verify Cases	Fri 11/15	Wed 11/20	61	4 d	LTW[
63	SumIST:3.9	Update Perf/Load/Cap/Vol	Tue 11/12	Mon 11/25	38	10 d	
64	SumIST:3.9.1	Update Scripts	Tue 11/12	Mon 11/18		5 d	EM[5
65	SumIST:3.9.2	Update Data	Tue 11/12	Mon 11/18		5 d	EM[5
66	SumIST:3.9.3	Verify Updates	Tue 11/19	Mon 11/25	64,65	5 d	EM
67	SumIST:3.10	Write Docu Tests	Thu 11/21	Fri 11/22	60	2 d	
68	SumIST:3.10.1	Write Cases	Thu 11/21	Fri 11/22		2 d	LTW[
69	SumIST:3.10.2	Verify Cases	Thu 11/21	Thu 11/21		1 d	LTW[
70	SumIST:3.11	Automate DMS Func Tests	Mon 2/10	Fri 3/14	128	26 d	
71	SumIST:3.11.1	Convert Cases to Scripts	Mon 2/10	Fri 2/28		15 d	ATE[5
72	SumIST:3.11.2	Adapt Data	Mon 2/10	Fri 2/28		15 d	ATE[5
73	SumIST:3.11.3	Verify Cases	Mon 3/3	Fri 3/14	71,72	10 d	ATE

Рис. 3.5. Фаза разработки тестов для подпроекта тестирования проекта "Суматра")

тестами, которые необходимо разработать. Например, функциональность управления документооборотом считается компонентом с большим риском, чем программа установки, обычная работа и сопровождение. Таким образом, Лин Цу должна завершить подготовку тестовых скриптов для тестирования вручную функциональности системы управления документооборотом до начала обновления тестовых скриптов для программы установки, обычной работы и сопровождения.

Иногда порядок, в котором я выстраиваю разработку тестов, соответствует порядку готовности к тестированию фрагментов системы. Например, комплект тестов для тестирования производительности и нагрузки чуть более важен (в соотношении приоритетов рисков качества), чем комплект тестов для комплексного тестирования, но основные функции системы управления документооборотом появятся в системе до того, как Эмма сможет начать тестирование производительности. Интеграция компонентов и комплексное тестирование начнутся, как только будет готова первая

пара взаимодействующих компонентов в инкременте 1. Таким образом, Эмма будет разрабатывать средства и скрипты для комплексного тестирования до обновления комплекта тестов для тестирования производительности и нагрузки.

Часть действий по разработке тестов связана с переработкой существующих ручных и автоматизированных тестовых скриптов. Естественно, что сложные и объемные тестовые скрипты потребуется обновлять дольше, нежели более короткие. Например, Джамал и Лин Цу оценивают в десять дней объем работы по переработке тестов для программы установки, обычной работы и сопровождения и выделяют пять дней на обновление тестов для обработки ошибок. Эти оценки основаны на предположениях Лин Цу и Джамала, что система управления документооборотом вызовет множество эксплуатационных накладных расходов, в частности на изменение оффлайновых носителей информации и сопровождение системы управления иерархическим хранилищем данных, а множество ошибок, вероятно, числом, несколько большим, чем для версии SpeedyWriter 3.0, будет в основном связано с такими аспектами, как недоступные или поврежденные файлы.

Работа по обновлению автоматизированных тестов для регрессионного тестирования функциональности (представлено в графике Гантта как "Обновить скрипты для SpeedyWriter 3.0")) оценивается в пять дней, а на пересмотр комплекта тестов для тестирования производительности и нагрузки отводится десять дней. Эти цифры отражают выводы Эммы и Джамала о том, что изменения в существующей функциональности представляют, в основном, несколько новых диалоговых окон, которые могут появиться для взаимодействия с системой управления документооборотом, а также корректировку работы некоторых старых функций в результате исправления ошибок. Более существенные последствия изменения производительности появляются в системе управления документооборотом, и число состояний системы может существенно возрасти (см. главу 10).

Для написания новых ручных и автоматизированных тестовых скриптов потребуется больше времени, чем на переработку старых. На подготовку новых автоматизированных тестовых скриптов для системы документооборота выделено 25 дней, тогда как на обновление существующих тестов отводится лишь пять дней.

Для тестовых сценариев могут потребоваться особые данные. Например, в проекте, где мы тестировали приложение по обработке ссуд для приобретения недвижимости, нам пришлось загружать специальные данные в приложение и несколько удаленных серверов, которые предоставляли нашей системе информацию об оценке кредитоспособности. В другом проекте сеть серверов интерактивной речевой связи (например, телефонная банковская система) была соединена с центром обработки вызовов, и в подсистеме центра обработки вызовов требовались совместно используемая база клиентов и подмножество данных об этих клиентах, загруженное в каждый сервер интерактивной речевой связи. Создание такого множества файлов с данными потребовало покупки специального инструмента тестирования и выделения трех человеко-месяцев на двухмесячный период.

В случае автоматизированного тестирования иногда можно использовать как коммерческие, так и бесплатные инструменты тестирования. В этом случае разработка тестов – это вопрос программирования или настройки инструмента под нужды пользователя. Эмма занимается как раз этим при разработке тестов для проверки производительности и нагрузки. Однако в некоторых случаях требуется разработать инструмент тестирования. Эмма создала такой инструмент для разработки тестов комплексного тестирования. Я часто видел, как заглушки, тестовые драйверы и тестовые данные, используемые для успешного модульного и компонентного тестирования, образуют отличные инструменты для комплексного и системного тестирования.

План-график должен также предусматривать время для проверки самих тестов независимо от того, новые они или старые, ручные или автоматизированные. До определенной степени этот период накладывается на период первого выполнения тестов. Однако я полагаю, что включение времени и затрат ресурсов на задачи по верификации тестов позволяет точнее оценить затраты в подпроекте. В противном случае время на поиск и исправление проблем в тестах тратится в первых одном-двух циклах тестирования, что в конечном счете приводит к перерасходу времени на выполнение тестов.

При подготовке плана-графика разработки тестов Джамал, Лин Цу и Эмма опирались на свой предыдущий опыт решения подобных задач при оценке продолжительности и затрат ресурсов на каждую задачу. Методы Дельфийского Оракула, "трех точек" и широкополосный метод могут применяться для выявления знаний группы[1].

Эмпирические правила разработки тестовых сценариев изложены в "*Estimating Software Costs*", написанной Джоунзом (Jones). Однако я видел достаточно много разных вариантов разработки тестов, чтобы выбирать в качестве основы оценок отдельных задач тестирования данные метрики. Если посмотреть на последние четыре проекта, в которых я участвовал, то в них использовались следующие измерения:

- Интернет-приложение – 350 человеко-дней на разработку 170 тестовых сценариев
- Сеть интерактивной речевой связи и центр обработки звонков – 1200 человеко-дней на разработку 1157 тестовых сценариев
- Обработка ссуд на приобретение недвижимости – 20 человеко-дней на разработку 37 тестовых сценариев
- Анализ данных маркетинга – 45 человеко-дней на написание 86 тестовых сценариев

---

[1] Для введения в основы оценки затрат в проектах и для знакомства с другими навыками по управлению проектами рекомендую обратиться к Wysocki и др. "*Effective Project Management*". В этом полезном материале, разбавленном жизненными примерами и сквозным рассмотрением одного конкретного случая, подробно описываются методы оценки затрат. Это групповые, итеративные методы получения наилучшей/наихудшей/ожидаемой оценки с помощью мозгового штурма. Также полезна статья Риты Хэдден (Rita Hadden) "Credible Estimations for Small Projects", опубликованная в "*Software Quality Professional*". В ней показана вариация метода, использующая экспертные оценки участников рабочей группы проекта.

В среднем требуется восемь человеко-часов на подготовку одного тестового сценария, но стандартное отклонение равно шести человеко-часам. Каждый из этих проектов имеет различное сочетание точности попадания тестового сценария, числа условий, покрываемых тестовым сценарием, ручного и автоматизированного тестирования, автоматизации с использованием коммерческих инструментов и средств собственной разработки, разработки тестовых данных и существовавших тестов для повторного использования, а также других факторов. Для сравнения, применение эмпирических правил Джоунза на 100 000 строк кода (100 KSLOC) программы дает около одного человеко-часа на тестовый сценарий. Затраты на разработку тестов в проекте Интернет-приложения примерно в четыре раза отличаются от затрат в проекте по анализу данных маркетинга.

Оценка разработки тестов в проекте "Суматра", сделанная Джамалом, дает примерно 75 человеко-дней на написание от 40 (начальная оценка) до 60 (включая 50% прибавки на непредвиденные обстоятельства) тестовых сценариев. Это соответствует данным для проектов по разработке Интернет-приложения и обработке интерактивной речевой связи. Я считаю, что вполне надежно предположить, что похожие проекты с похожими группами тестирования, технологиями и методами покажут примерно одинаковые цифры затрат на тестовый сценарий, но снова мы обязаны рассмотреть множество различных факторов. Предположение, что "проект В такой же, как проект А, а в проекте А мы потратили два человека-дня на тестовый сценарий при разработке тестов, следовательно, это можно использовать как оценку и не рассматривать остальные факторы, влияющие на время разработки тестов", возможно, через некоторое время не даст точной оценки. Еще менее точным будет применение чужих эмпирических правил. Данные моих проектов и эмпирические правила Джоунза иллюстрируют, что разброс данных может отличаться на порядки, и оценка, которая расходится с фактом на 1000%, не слишком полезна.

Группа тестирования Джамала будет разрабатывать тесты на протяжении большей части проекта, 26 недель. Однако основная часть этой работы относится к периоду в 11 недель, начиная с середины сентября и заканчивая концом ноября. В течение этого времени Эмма, Лин Цу и вновь принятый на работу специалист по автоматизированному тестированию будут работать практически полный рабочий день над обновлением, написанием и верификацией тестов.

Настройка тестовой среды

По сравнению с продолжительной фазой разработки тестов Джамал запланировал относительно короткую четырехнедельную фазу по настройке тестовой среды (см. рис. 3.6). Сначала должны быть закуплены клиентские рабочие станции и серверы. Оценка в пять и десять дней соответственно указывает на четкий и простой процесс их приобретения в Software Cafeteria. Также элементами среды являются вполне обычные доступные в свободной продаже аппаратно-программные средства. Никакой внутренней разработки аппаратных и программных средств (за исключением самой системы SpeedyWriter), специальных заказов на внеш-

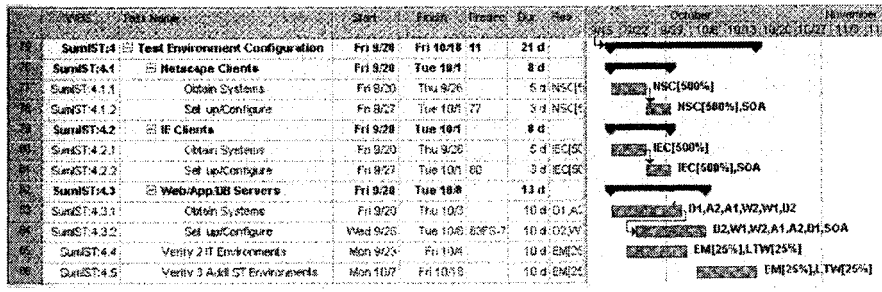


Рис. 3.6. Фаза настройки тестовой среды

нюю разработку средств или чего-то подобного не планируется. Я участвовал в нескольких проектах с применением заказного аппаратного обеспечения. В этих случаях планировались продолжительные периоды для создания заказных систем.

Работа по установке и настройке тестовой среды выполняется главным образом группой системных операций и администрирования под руководством Кейта Ли. Три дня на настройку клиентских машин и десять дней на конфигурирование каждого сервера означают, что группа Кейта может быстро реагировать на запросы Джамала. Я работал в проектах, где это было не так, особенно в тех случаях, когда группы поддержки сетевых операций были малочисленны. Иногда при установке серверов возникают вопросы безопасности, особенно если серверы должны быть включены в корпоративную сеть. Все эти факторы необходимо принимать в расчет.

Сложная тестовая среда может потребовать больших затрат времени и ресурсов. В одном из проектов я потратил два человека-месяца на планирование и руководство конфигурированием тестовой среды. Группа сетевых операций израсходовала на установку и настройку тестового оборудования 36 человеко-месяцев[1].

Бывает, что установка и настройка тестовой среды тривиальны. В одном из проектов мне пришлось купить единственную рабочую станцию, заменить процессор и добавить оперативной памяти на сервере. Я израсходовал примерно один человеко-день на эту работу, включая оформление заказа, получение, подключение к сети, установку аппаратуры и проверку конфигурации. Тем не менее продолжительность фазы была около трех недель, поскольку мы с заказчиком должны были договориться о том, на какой среде проводить тестирование приложения.

---

[1] Некоторые средства и методы управления работами по конфигурированию сложных и больших тестовых сред рассмотрены в главах 6 и 7 книги Rex Black "*Managing the Testing Process, Second Edition*".

Выполнение тестов

Джамал планирует провести 11 двухнедельных тестовых раундов в ходе системного тестирования. Из них три раунда будут проведены после добавления последнего инкремента. Джамал принимает еженедельно тестовую версию (с исправлениями ошибок, обнаруженных на ранее выполненных тестовых циклах) в течение всего системного тестирования, за исключением последнего двухнедельного раунда, который он планирует целиком посвятить тестированию "кандидата в золотую сборку". Графическое представление этого плана показано на рис. 3.7.

Первое предположение состоит в том, что каждый раунд можно завершить за две недели. Ранее в нашем примере Джамал использовал таблицу для определения продолжительности каждого раунда. Он оценил необходимые затраты персонала и объемы доступного оборудования на каждый тестовый сценарий, а затем поработал с планами по персоналу и оборудованию и пришел к выводу о двухнедельном раунде. Конечно, это только примерная оценка. Поскольку Джамал предполагает, что помимо уже имеющихся будут разработаны дополнительные тесты, то, основываясь на данных предыдущих проектов, он заложил в план 50% дополнительных тестов. Также в план-график включено время на исследовательское

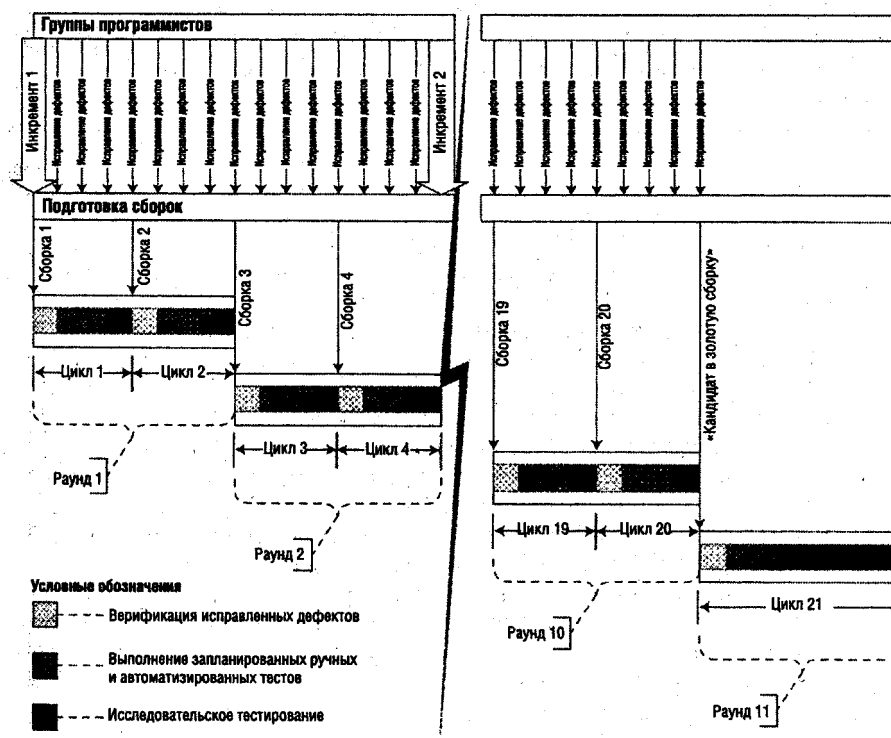


Рис. 3.7. Подробный план проведения системного тестирования

тестирование, которое может увеличиваться или уменьшаться в зависимости от изменений рисков, относящихся к выполнению тестов.

Один из других неизвестных ему заранее факторов — это сколько дефектов группа тестирования будет находить в каждом цикле (что удлинит выполнение тестовых сценариев, обнаруживших ошибки), а также сколько исправленных дефектов будет в каждой тестовой сборке (что может удлинять или укорачивать продолжительность верификации исправления дефектов в начале каждого тестового цикла). Я обычно использую следующее эмпирическое правило: при получении относительно качественных версий и при недельном цикле требуется четыре-шесть часов для верификации дефектов. Я также допускаю, что могу добавить еще немного времени (примерно 20%) к планируемому периоду выполнения тестового сценария для подготовки отчетов о дефектах (см. главу 14).

Число найденных ошибок не берется в расчет при оценке длительности раунда, а это приводит нас к некоторому допущению. Джамал считает, что он может провести 10 раундов тестирования, в ходе которых будут найдены все ошибки, подлежащие обязательному устранению. В последнем, 11-ом, тестовом раунде для "кандидата в золотую сборку" он не планирует найти такие ошибки. Но можем ли мы быть уверены, что так и будет?

Для ответа на этот вопрос существуют статистические методы, использующие модели устранения дефектов. Во втором издании книги "*Metrics and Models in Software Quality Engineering*" Стефан Кан (Stephen Kan) подробно описывает некоторые из этих моделей. Если требуется точная оценка числа тестовых раундов для обеспечения надлежащего качества поставляемого продукта, можно воспользоваться одной из моделей на основе измерений. Базовая модель устранения дефектов на основе фактических данных предыдущих проектов обсуждается в "CMM in Practice", написанной Панкоджем Джелотом (Pankoj Jalote), а также во втором издании моей книги "*Managing the Testing Process*". Если в компании имеются хорошая база данных дефектов с отслеживанием изменения их статуса и средства для оценки общего числа дефектов в системе (см. главы 4 и 14), можно опереться на базовую модель устранения дефектов Джелота. Уоттс Хемфри (Watts Humphrey) в книге "*Introduction to the Personal Software Process*" обсуждает модели устранения дефектов одним человеком, хотя они ориентированы больше на программистов[1].

В большинстве случаев сроки выпуска имеют большее значение, чем тестирование и исправление каждой найденной в конце проекта ошибки. Это одно из преимуществ жизненного цикла итерационной разработки. В рассматриваемом примере, если инкремент 5 будет нестабильным, проектная группа может вернуться к инкременту 4 и сосредоточиться на подготовке его к выпуску в объявленный срок. Джамал опирается на эту возможность, когда планирует число тестовых раундов, отсчитывая срок в обратном направлении от даты выпуска к дате получения первого инкремента для системного тестирования.

---

[1] Для прогнозирования числа найденных и исправленных в проекте дефектов не требуется использовать все имеющиеся в базе данные, хотя это может потребоваться для более точного прогноза. На web-сайте этой книги есть шаблон Estimated Bug Find-Fix.xls, использующий простую модель, которая обычно дает неплохие результаты.

Говоря об инкрементах и поставке определенных фрагментов функциональности, в разделе "Разработка тестов" я утверждал, что такие соображения до некоторой степени могут быть стимулом при подготовке определенных видов тестов. Это также часто является фактором для выполнения тестов. Если в проекте, использующем каскадную модель системной разработки, вы отвечаете только за системное тестирование, одним из критериев начала тестирования может быть получение версии с полным набором реализуемых свойств. В этом случае вся функциональность должна быть реализована в системе до начала тестирования.

Однако в итерационной, спиральной (spiral) модели, экстремальном программировании и других моделях, в которых новые компоненты и функции поступают в ходе всей фазы тестирования, оценки затрат должны учитывать порядок поступления на тестирование определенных компонентов. Это всегда верно для комплексного тестирования, которое зависит от сроков готовности различных связующих и интерфейсных компонентов. В одном из проектов я работал с проектной группой над определением последовательности из шести стержневых фрагментов для комплексного тестирования, последним из которых была система целиком, а началом было небольшое число кусков каждой подсистемы, к которым добавлялись по мере готовности другие кусочки. Добавление фрагментов происходило в порядке уменьшения риска вероятности или опасности проблем для жизнеспособности системной архитектуры. Помимо влияния на оценку затрат и сроков выполнения работ, это имело последствия для логики системы и покрытия ее тестами. Данные аспекты более подробно рассматриваются в главах 7 и 11[1].

Джамал запланировал пять раундов интеграционного тестирования, все, кроме последнего, продолжительностью в четыре недели. Каждый раунд соответствует одному инкременту. Это аналогично описанной методике стержневых фрагментов. Комплексное тестирование посвящено поиску ошибок в связях и взаимодействии компонентов, поэтому Джамал считает, что он сможет найти все такие ошибки и проверить их исправление в течение четырех недель. Поскольку это первый опыт применения жизненного цикла инкрементной разработки в компании Software Cafeteria, последнее предположение может быть неверно. Однако с учетом времени предоставления инкрементов на тестирование полученный график соответствует общему плану-графику проекта.

В графике Джамала 25 недель отводится на выполнение тестов со значительным перекрытием фаз комплексного и системного тестирования. Трудозатраты оцениваются в 366 человеко-дней; при этом учитываются четыре младших тестировщика, три тестировщика и часть рабочего времени Джамала.

---

[1] Еще в 1984 г. Борис Бейзер (Boris Beizer) в книге "*Software System Testing and Quality Assurance*" написал хорошую главу об использовании рисков для последовательной разработки и комплексного тестирования систем. Также подробное обсуждение комплексного тестирования можно найти в книге "*Systematic Software Testing*", написанной Риком Крейгом (Rick Craig) и Штефаном Яскилем (Stefan Jaskiel).

Переход на доллары и центы

Теперь читатель должен неплохо представлять, как составить декомпозицию работ и оценить ресурсы и сроки выполнения работ для подпроекта тестирования. Эти оценки постепенно все больше и больше обрастают подробностями, которые определяют время с точностью до дня и позволяют сопоставить ресурсы с конкретными задачами. Хотя план в виде графика Гантта дает общую картину в терминах времени, большинство инструментов управления проектом не могут так же хорошо справиться с финансовыми аспектами проекта. В следующей главе мы изучим, как на основе декомпозиции работ составить бюджет. Также мы изучим увязывание бюджета с тем вопросом, о котором в действительности заботятся старшие менеджеры, высшее руководство и заинтересованные лица, а именно с возвратом инвестиций.

ГЛАВА 4

Не сколько стоит, а сколько экономит: бюджет и возврат инвестиций

Декомпозиция работ и план-график помогают нам понять, сколько времени и ресурсов необходимо для выполнения подпроекта тестирования, покрывающего все риски, выявленные ранее в ходе анализа рисков качества. Однако мы пока не рассматривали еще один важный аспект организации тестирования — финансовый. На этом этапе процесса оценки затрат мы обсудим, как на основе декомпозиции работ можно составить бюджет, который соответствует финансовым ограничениям проекта. Получив бюджет и план-график, мы будем иметь полную оценку затрат, которая может использоваться в качестве первоначального предложения руководству.

В ходе обсуждения с руководством границ тестирования в проекте я часто вынужден соглашаться с их урезанием. (Это означает, что тестирование будет покрывать меньше рисков системы качества или покроет эти риски, но они будут проверяться менее детально, либо и то и другое в комбинации.) Сокращение границ тестирования будет продолжаться до тех пор, пока группа управления проектом и тест-менеджер не перейдут от обсуждения того, что они *должны* тестировать, к обсуждению того, что они *могут* протестировать в границах контекста проекта. Правильно проведенный анализ рисков (см. главу 2) обычно помогает группе управления проектом провести разумные сокращения в границах тестирования на основе представлений о рисках.

Конечно, нам хочется быть уверенными, что в ходе обсуждения осуществимых графика и бюджета мы не провалим защиту адекватного тестирования. Выбор *правильного* множества тестов для данного проекта — не только техническая задача, это также политика и бизнес. Мы должны убедительно объяснить, почему мы думаем, что определенные тесты важны для проверки критических рисков качества, и что ресурсы, которые мы намерены израсходовать на тестирование каждого риска, будут использованы правильно. К счастью, процесс подготовки бюджета также предос-

Коэффициент окупаемости инвестиций (КОИ) (Return on Investment, ROI)

Величина, полученная в результате инвестирования в некоторый проект. Коэффициент окупаемости инвестиций – это финансовая прибыль от инвестиции за вычетом начальных затрат, деленная на начальные затраты:

КОИ = (*Прибыль* – *Затраты*) / *Затраты*

КОИ обычно умножают на 100% и приводят в процентах.

тавляет тестировщикам возможность оценки отдачи от инвестиций в тестирование. Существует несколько методов, удобных для этой цели. Вычисления возврата инвестиций, хотя и несколько грубые, способствуют поддержке наиболее разумного и тщательного тестирования, которое может обеспечить проект.

Анализ возврата инвестиций в тестирование

Предыдущие шаги составления бюджета были механическими, поэтому мы не будем обсуждать их до тех пор, пока не покажем, как Джамал работает с задачами. Вместо этого рассмотрим понятие возврата инвестиций в тестирование. Как тестирование приносит доход компании?[1]

Начнем с того, что мы стоим перед фактом, что некоторые менеджеры смотрят на тестирование, как на большую черную дыру в конце проекта. Им кажется, что чем больше они тратят на тестирование, тем больше оно требует новых средств. Другие менеджеры считают тестирование полезной деятельностью, приносящей прибыль в ходе всего проекта. Они уверены, что разумные инвестиции в тестирование приносят весьма заметную отдачу.

Как тестировщик-профессионал вы обнаружите, что лучше работается со вторым типом менеджеров. Как мы можем убедить менеджеров, что тестирование – это удачное вложение средств, и с пользой обсудить возможную отдачу? Чтобы проанализировать возврат инвестиций на тестирование, вновь взглянем на четыре уже упомянутых основных способа, посредством которых тестирование приносит отдачу:

1. Обнаружение ошибок, которые устраняются, и даже предотвращение их внесения.
2. Обнаружение ошибок, которые не устраняются, но о которых становится известно.

---

[1] Нижеследующее обсуждение включает в себя подробное изучение различных финансовых вопросов, которые могут оказаться для читателя лишними. Упрощенное представление тех же понятий можно найти в моей презентации "Four Ways Testing Adds Value" на www.rexblackconsulting.com, сопровождающейся аудиозаписью моего основного выступления на эту тему, сделанного на EuroSTAR 2000 в Эдинбурге, Шотландия.

3. Проведение тестов, которые снижают (потенциально затратные) риски.
4. Обеспечение проекта своевременной, точной и заслуживающей доверия информацией.

Рассмотрим каждый из этих способов, чтобы завершить общий анализ возврата вложений в тестирование. После обсуждения принципов для каждого варианта получения прибыли от тестирования мы вновь вернемся к Джамалу Брауну, его бюджету для проекта SpeedyWriter и возврату инвестиций в его подпроект тестирования[1].

Исправленные и предотвращенные ошибки

Вычисление возврата инвестиций для найденных (при тестировании) дефектов, а затем исправленных либо предотвращенных (работой группы тестирования) до выпуска системы приводит нас к использованию хорошо известного подхода под названием "анализ *стоимости качества*". Метод стоимости качества, точнее, стоимости плохого качества, разбивает затраты, связанные с качеством на две категории, каждая из которых в свою очередь также делится на две части:

1. Затраты на соответствие (conformance), включающие в себя все затраты, связанные с обеспечением или проверкой качества системы:
   - Затраты на предотвращение несоответствий ($CC_{предотвращения}$), включающие в себя затраты на обучение, экспертные оценки документации и другие реальные методы контроля качества.
   - Затраты на обнаружение несоответствий ($CC_{обнаружения}$), которые включают в себя траты, связанные с подготовкой к тестированию, выполнением каждого теста (один раз) и обсуждением обнаруженных несоответствий.
2. Затраты на несоответствия, включающие в себя все затраты, связанные с проблемами качества системы, т.е. затраты, относящиеся к невозможности достигнуть совершенного качества с первой попытки:
   - Затраты на внутренние несоответствия ($FC_{внутренние}$), включающие в себя затраты на подготовку отчетов о дефектах, исправление дефектов, подготовку исправленной версии, проверку и подтверждение, что ошибки исправлены, регрессионное тестирование, переработку тестов и документации, реагирование

---

[1] Рик Крейг (Rick Craig), соавтор "*Systematic Software Testing*" и мой коллега, консультант по тестированию, указал, что тестовые сценарии становятся полезной частью документации системы. В дискуссии по электронной почте по поводу этой главы он писал: "Другой неожиданный доход от инвестиций в тестирование состоит в том, что сами тестовые сценарии могут рассматриваться как часть системной документации, особенно для древних систем, спецификации требований и архитектуры которых давно устарели. Часто ли к тебе приходят программисты с вопросом: "Рекс, можно взглянуть на твои тестовые сценарии, чтобы понять, как эта система должна работать?" На самом деле, это отличный признак того, что ты как тестировщик состоялся".

на изменения в системе, задержки выпуска или поставки системы и т.п.

- Затраты на внешние несоответствия ($FC_{внешние}$), которые обычно включают в себя большинство внутренних затрат на устранение несоответствий плюс дополнительные потребности в технической поддержке, поставке версий и патчей сопровождения, потенциальный ущерб репутации компании и снижение продаж, общение с рассерженными клиентами и другие еще более нематериальные потери.

Можно представить стоимость качества в виде формулы:

$$C_{качества} = (CC_{предотвращения} + CC_{обнаружения}) + (FC_{внутренние} + FC_{внешние})$$

Вы можете спросить: "Ну и что? Как эти хитрые трюки с расщеплением и соединением финансовых данных могут мне помочь?" Хороший вопрос. Это поможет вам, недофинансированному профессионалу, поскольку люди, которые изучили эту тему, обнаружили, что для многих компаний независимо от отраслей, в которых они работают, затраты на внешние несоответствия обычно существенно превосходят затраты на предотвращение, обнаружение и внутренние несоответствия. Филипп Кросби (Philip Crosby) в книге "*Quality Is Free*" объясняет, что инвестиции в качество могут принести ощутимый положительный эффект, если измерить его в терминах обнаруженных и устраненных проблем до того, как их увидели заказчики[1].

Точные затраты, связанные с внутренними и внешними несоответствиями, могут значительно различаться для разных проектов разработки и сопровождения систем. Данные по отрасли нелегко обнаружить в Интернете или в других открытых источниках. Старое эмпирическое правило, позаимствованное из "*Software Engineering Economics*" Барри Бёма (Barry Boehm), гласит, что затраты на исправление ошибки на фазе требований и архитектуры равны $1, затраты на исправление при обнаружении ее программистом равны $10, при обнаружении независимой группой тестирования и при исправлении до выпуска программы равны $100, при обнаружении заказчиком и последующем исправлении программистом, внутреннем тестировании и выпуске в рамках версии сопровождения равны $1000. Данные оценки могут в зависимости от области, в которой работает читатель, рассматриваться как чересчур консервативные или слишком либеральные (см. врезку "Калькуляция стоимости дефекта"))[2].

Это средние затраты на ошибку: некоторые ошибки стоят дороже, некоторые дешевле в зависимости от сложности их исправления, объема необходимого регрессионного тестирования, влияния исправления на

---

[1] Введение в эту тему есть в предисловии книги Кросби. (Роб Сабурин (Rob Sabourin), один из рецензентов, отметил, что необходимо убедиться в правильном понимании заголовка – некоторые считают, что добиться качества ничего не стоит.) Более подробно см. Juran, Gryna, "*Quality Control Handbook*" или Campanella, "*Principles of Quality Costs*". Наконец, о применении этого понятия в производстве программных продуктов см. Slaughter et al., "Evaluating the Cost of Software Quality" в "*Communications of the ACM*", Volume 41, Number 8 (Aug 1998).

другие части системы и т.д. На любом отдельно взятом уровне качества различие в затратах на конкретную ошибку не играет роли. Как я покажу ниже, окупаемость тестирования предполагает, что группа тестирования обнаружит ошибки, исправление которых обошлось бы более чем в $1000, если бы это было на стадии эксплуатации системы. Вообще говоря, чтобы получить положительное сальдо от инвестиций на тестирование, суммарная стоимость затрат на обнаружение дефектов и затрат на внутренние несоответствия должна быть меньше, чем суммарные затраты на внешние несоответствия, независимо от того, проводилось ли тестирование и были ли найдены ошибки. Если группа тестирования растрачивает попусту время и ресурсы в поисках ошибок, которые не относятся к реальному использованию системы и критическим рискам ее качества, то окупаемость тестирования будет невелика независимо от числа обнаруженных ошибок.

Усредненные затраты на ошибку, подлежащую обязательному исправлению, вероятно, не являются константой для различных уровней качества системы. Эта закономерность имеет ряд интересных следствий, которые не будет излишним упомянуть ниже.

Скажем, вы потратили 1 миллион долларов на проект разработки. Пусть степень удовлетворенности заказчика можно измерить в процентах от 0 до 100. При удовлетворенности заказчиков в 0% все ваши заказчики, воспользовавшись 30-дневной гарантией, возвращают отвратительную систему, т.е. вы не получаете никакого дохода от проекта. Если все вернут систему, затраты на несоответствия составят, как минимум, 1 миллион долларов, поскольку весь проект оказался простой тратой денег из-за низкого качества. При 100%-ной удовлетворенности заказчиков, т.е. если все полностью удовлетворены и никто не вернул продукт, затраты на несоответствия будут равны затратам, связанным с внутренними несоответствиями, т.е. с исправлениями ошибок до выпуска версии.

Теперь допустим, что группа разработки внесла 1000 дефектов в систему в ходе разработки. Предположим, что для дефектов, обнаруженных внутри проекта, треть исправлена на фазах требований и архитектуры, треть исправлена программистами в ходе кодирования, отладки, модульного и компонентного тестирования, а треть — в ходе комплексного и системного тестирования, проводимого тестировщиками. Таким образом, по эмпирическому правилу Бёма, система, в которой исправлены все ошибки до выпуска версии, имеет следующую стоимость несоответствия:

$$\left(\frac{1000}{3}\textit{ЛиниЖЛНЬ} \times \frac{\$1}{\textit{ЛиниЖНЬа}}\right) + \left(\frac{1000}{3}\textit{ЛиниЖЛНЬ} \times \frac{\$10}{\textit{ЛиниЖНЬа}}\right) + \left(\frac{1000}{3}\textit{ЛиниЖЛНЬ} \times \frac{\$100}{\textit{ЛиниЖНЬа}}\right)$$

$$= \$37\,000$$

---

[2] Марк Фьюстер (Mark Fewster) и Дороти Грэм (Dorothy Graham) приводят чуть более высокие цифры в "*Software Test Automation*". Джоанна Ротман (Johanna Rothman) на своем сайте www.jrothman.com сообщает об ошибках периода эксплуатации системы средней стоимостью в $10 000. Боб Барлет (Bob Barlett) из SIM Group пишет, что один из его клиентов, крупный банк, оценивает средние затраты на проблемы с программным обеспечением в производственных системах в 150 000 евро (см. "Power Testing" в "*Proceedings of the Fifth International Quality Week Europe*" conference, p.191).

Калькуляция стоимости дефектов
Эд Уэллер (Ed Weller)

Как можно определить стоимость дефекта при отсутствии учетной системы, спроектированной специально для сбора информации о стоимости качества? При отсутствии описанного процесса, которому четко следуют разработчики, тестировщики и персонал поддержки, я использую следующие приближения.

Для системного тестирования возьмите дату его начала. Обычно она варьируется в зависимости от компании, но, как правило, это событие, когда вся функциональность встроена в продукт и не планируется добавлять новой функциональности (иногда это событие называется замораживанием кода (code freeze)). Считайте с этого момента (с применением системы учета затрат труда, принятого в компании) все затраты сотрудников, занятых в проекте, до даты поставки продукта. Вычтите любые затраты в конце проекта, связанные исключительно с подготовкой продукта к поставке. Но учитывайте затраты на повторные действия по подготовке продукта к поставке, поскольку при предыдущих действиях обнаружились ошибки.

Теперь вычтите время, понадобившееся на один тестовый цикл, в ходе которого не было обнаружено ошибок. Если вы следуете процессу выполнения тестов, описанному в главе 13, то у вас есть эта цифра. Ее трудно оценить, не отслеживая результатов тестирования. Если это так, можно заметить, что, как и во многих других компаниях, это время может игнорироваться, поскольку оно вносит ошибку менее чем 10%.

Наконец, разделите полученное число часов на число обнаруженных и исправленных в ходе тестирования дефектов, где слово "исправленных" означает, что изменение было внесено в систему управления конфигурацией. Различные виды отчетов о дефектах, процессы их отслеживания и управления, обсуждаемые в главе 14, способствуют сбору этих данных.

Полученное число есть величина затрат на один дефект. Заметим, что учитывается время разработчиков на исправление дефектов, время, потраченное инженером сборок на повторную сборку версии, время, потраченное менеджером на управление, и другие проектные затраты, поскольку ни одна из этих работ не понадобилась бы, если бы не были найдены дефекты. Аналогичный подход удобно применять и для комплексного тестирования (и других фаз, проходящих до системного тестирования).

Для модульного тестирования получение этого числа из системы учета затрат труда и числа дефектов означает, что необходимо подсчитать затраты на модульное тестирование и число дефектов, обнаруженных в ходе него. Немногие делают это, но из опыта известно, что затраты на дефект значительно выше, чем разработчики, которые ненавидят вести учет обеих величин, согласны допустить. Оказывая консалтинговые услуги, я встречал отчеты и данные, указывающие на то, что затраты на модульное тестирование, примерно равные 50% всех затрат на кодирование и модульное тестирование, дают хорошую оценку.

Грустно признавать, но часто метрики недоступны для многих дефектов. Для сложных систем в моей практике диапазон затрат на дефект менялся от 8 до 32 часов. В случае простых систем разработчики будут убеждать, что это минуты, но они обычно не берут в расчет перекомпиляции, пере-

проектирование, перекодирование и перетестирование. Однако любые время и средства, израсходованные по причине наличия ошибки, являются затратами на несоответствия, если это происходит при тестировании.

Наиболее сложная проблема, связанная с получением более точных цифр, состоит в том, что необходимо иметь довольно точную систему учета затрат. Это значит, что, как только вы обнаруживаете ошибку в ходе системного тестирования, все затраты тестировщиков и разработчиков на разрешение проблемы должны быть учтены. Разработчик, который вводит активность "кодирование" в систему учета трудозатрат, вносит неточности, в результате которых затраты на дефект кажутся меньше, чем они есть на самом деле.

Я вспоминаю тест-менеджера, с которым проработал несколько лет. Он посчитал трудозатраты на последний дефект в ходе комплексного тестирования большой и сложной корабельной системы. Они оказались равными 2.5 *человека-месяца*. Не поверив, он пересчитал все заново. Основная часть затрат пришлась на повторное выполнение всего комплекта тестов после исправления дефекта. Смысл этого замечания состоит в том, что необходимо проверить, что все затраты учтены, поскольку неточные, но относительно верные цифры могут удивить.

Последнее замечание относится к затратам после поставки продукта: говоря языком стоимости качества, затратам на внешние несоответствия, включающим в себя затраты поставщика и заказчика. Как поставщик вы тратите время на обнаружение, исправление, тестирование, повторную сборку и поставку программного продукта. Во многих компаниях правила учета требуют, чтобы эти затраты калькулировались отдельно; в других компаниях есть специальная группа поддержки и другие субподрядчики по исправлению дефектов. Кроме того, не забывайте посчитать затраты на сетевую систему поддержки пользователей. Сумма этих затрат, деленная на число дефектов, дает другую цифру затрат, которую добавляют к затратам на тестирование. Затраты заказчика обычно опускают. На сколько падает производительность труда в день из-за зависаний персональных рабочих станций? Вспоминаю совещание с участием заказчиков, где они утверждали, что каждый час простоя из-за проблем в системе обходится им примерно в 1 миллион долларов. Хотя это, вероятно, явно не отразится в итоговой графе прибыли компании (если, конечно, с заказчиками не было заключено договоров об уровне обслуживания), подобная ситуация ведет к появлению недовольных заказчиков, которые часто становятся *бывшими* заказчиками. Не забывайте, что Linux пользуется популярностью не из-за бесплатного распространения, а потому что она бесплатная *и* надежная одновременно.

В терминах затрат на соответствие, предположим, что чем больше в системе дефектов, тем легче их обнаружить. Если требуется меньше затрат на обнаружение дефекта, стоимость обнаружения ниже. Напротив, чем меньше ошибок в системе, тем труднее их обнаружить. Если нужно больше затратить на поиск ошибки, затраты на обнаружение выше.

Если вы когда-либо тестировали систему, полную ошибок, то вам известно, что квалифицированный тестировщик может найти в такой системе ошибку. Для этого не нужны ни планирование, ни инструменты, ни

проектирование тестов, ни их разработка. Предположим, что в системе, которая может привести к 0% удовлетворенности пользователей, мы можем обнаружить первую ошибку не более чем за $20.

Продолжая тестирование, обнаруживая ошибки и получая новые тестовые сборки, в которых эти ошибки исправлены, вы вынуждены работать все более напряженно, чтобы обнаружить следующую ошибку. Пусть система без ошибок будет давать 100%-ную удовлетворенность заказчиков, однако затраты на обнаружение последней ошибки будут стремиться к бесконечности.

В этой модели тестирование и контроль качества, в конечном счёте, достигнут точки убывания возврата инвестиций, когда стоимость качества достигнет минимума, как только стоимость соответствия сравняется со стоимостью несоответствия[1].

Предположим, что каким-либо способом мы могли предсказать удовлетворенность заказчика на основе числа дефектов, остающихся в продукте. На рис. 4.1 представлен график того, как могут выглядеть затраты на различных уровнях удовлетворенности заказчика.

Уровень качества, минимизирующий стоимость качества, соответствует 75%-ной удовлетворенности заказчика. В терминах дефектов, поставленных заказчику вместе с продуктом, можно сказать, что это соответствует почти 100%, поскольку 90% дефектов, подлежащих обязательному исправлению, были устранены. Затраты на соответствие (и на контроль качества, и на тестирование), необходимые для достижения этого уровня удовлетворенности, приблизительно равны $125 000[2].

К счастью, нет необходимости размышлять над тщательно разработанной математической моделью и причудливыми графиками, подобными рис. 4.1, чтобы применять понятие стоимости качества. Я собираюсь продемонстрировать способы применения этого понятия для исследования текущего возврата затрат на тестирование и в некоторых случаях для оправдания увеличение затрат. Далее в этой главе в рамках проекта "Суматра" будет рассмотрен пример такой ситуации. Джамал будет использовать только метод Уэллера для оценки затрат внутренних и внешних несоответствий, оценки числа дефектов в продукте и оценки числа дефектов, которые будут найдены.

Наконец, раннее вовлечение в проект группы тестирования также способствует предотвращению ошибок. Группа тестирования, которая допускается до рецензирования спецификации требований или архитектуры до начала разработки, сопровождения или установки, может обнаружить неоднозначности, противоречия, пропущенные детали и другие

---

[1] Идея, что в некоторой точке затраты на соответствие превысят затраты на несоответствие, — это стандартная модель стоимости качества, развитая Джураном (Juran) и Гриной (Gryna) в "*Quality Control Handbook, Fourth Edition*". Однако наиболее современная мысль в управлении качеством состоит в том, что затраты на соответствие никогда не превысят затраты на несоответствие.

[2] На графике видно, что общая стоимость качества в точке минимума примерно равна $250 000, или 25% бюджета проекта. Это может показаться нереально низким. Однако в "*Principles of Quality Costs*") Кампанелла (Campanella) и др. отмечают, что компания Raytheon Electronic Systems в ходе работы по подготовке к сертификации на 4-й уровень CMM (Capability Maturity Model) сократила затраты на качество с 80% почти до 20%.

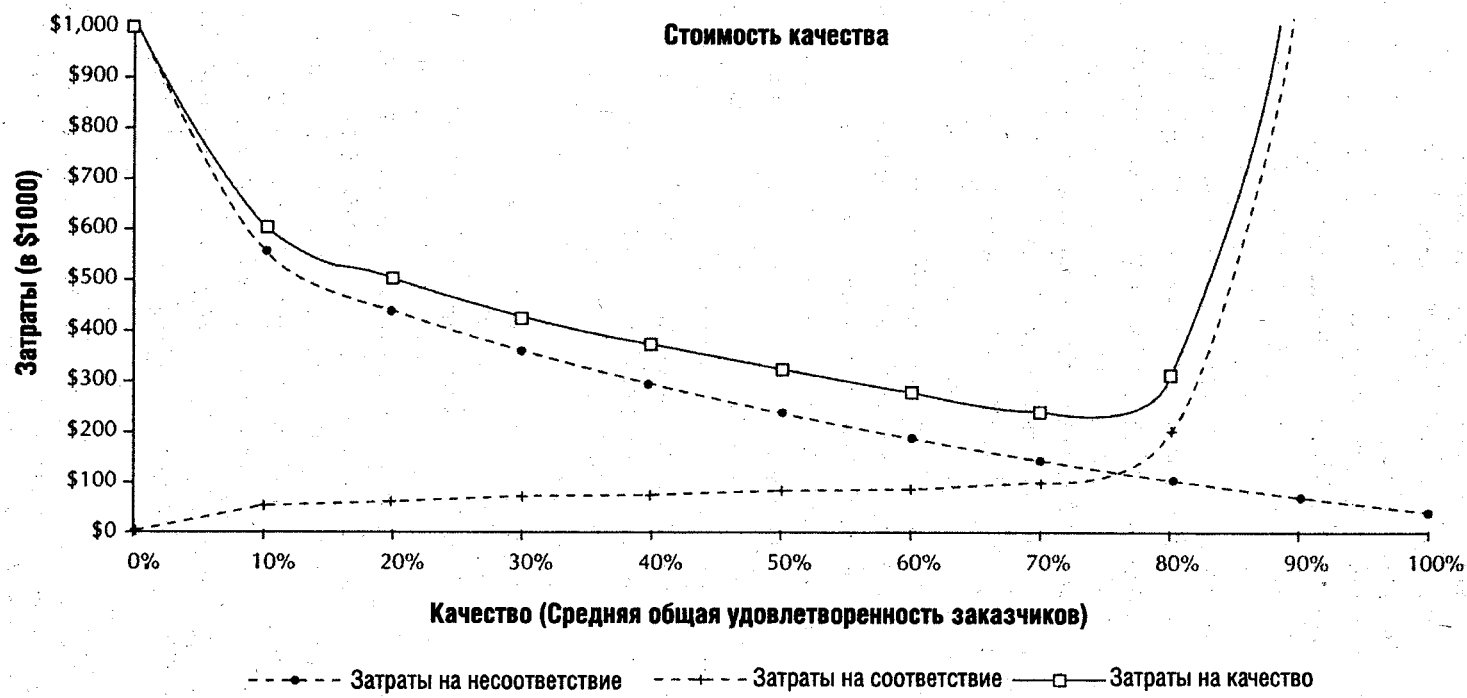


Рис. 4.1. Гипотетический график стоимости качества

проблемы в этих документах. Такого рода проблемы в противном случае превращаются хотя бы в одну ошибку, но обычно их больше. Тестировщики, как показывает мой опыт, особенно полезны как рецензенты, поскольку они постоянно задаются вопросом: "А что, если?" и способны предвидеть необычные ситуации.

Финансовая выгода от раннего вовлечения группы тестирования в проект – это стандартные преимущества, которые приобретаются в результате рецензирования документов с описанием требований и архитектуры. Их трудно измерить количественно, поскольку для этого требуется отслеживать обнаруженные проблемы и их исправление в документации. Компании, использующие формальные инспекции или точные модели удаления дефектов, располагают процессами для отслеживания такого рода проблем, но в большинстве компаний таких процессов нет. Независимо от возможности количественного измерения преимуществ рецензирования силами тестировщиков, надо честно сказать, что с учетом значительно меньшей стоимости исправления дефектов в документации по сравнению с исправлением ошибок в коде возврат затрат на час работы тестировщика по рецензированию требований и архитектуры обычно равен или превосходит возврат на час работы тестировщика на любой более поздней стадии проекта[1].

Неисправленные, но известные ошибки

Даже если обнаруженные ошибки не исправлены, процесс по-прежнему имеет ценность. Каким образом? Позвольте применить следующую аналогию. Предположим, вы собираетесь совершить длительное летнее путешествие на автомобиле. До начала путешествия вы, вероятно, отправитесь в ближайший автосервис проверить машину. Механик говорит, что под капотом все выглядит отлично, но про шины известно, что в условиях сильной жары, высокой скорости и низкого давления в них они могут отказать. Вы не можете позволить себе поменять шины, однако делаете пометку, что необходимо управлять транспортным средством таким образом, чтобы это не повлекло неприятностей с шинами, и принять необходимые предупредительные меры, например проверить наличие запасных частей, запаса питьевой воды, пищи, аптечки первой помощи и т.д. То же самое всякий раз происходит и с программными системами. Многие из ошибок, которые мы находим, не исправляются, а становятся известными ограничениями и предупреждениями пользователям.

В некоторых случаях мы можем принимать активные меры по недопущению пользователя в опасную зону системы, а не устранять эту зону. Например, рассмотрим дыру в программе SpeedyWriter: редактор не может работать с определенными версиями Java в браузере. Программы установки позволяют делать выбор обновления до новой версии с разрешения пользователя и предотвращают возникновение этой ситуации.

---

[1] В "*Systematic Software Testing*" авторы Крэйг (Craig) и Яскиль (Jaskiel) поясняют идею превентивного тестирования и методов его достижения. Модели удаления дефектов описаны в "*Metrics and Models in Software Quality Engineering, Second Edition*" Кана (Kan).

В других случаях мы и наши заказчики обречены жить с ошибками. В этой ситуации технические писатели могут документировать известные проблемы в описаниях отличий новой версии, руководствах пользователя, экранах помощи, сообщениях об ошибках и другой системной документации. Сотрудники отделов маркетинга и продаж могут передавать эту информацию пользователям до приобретения или поставки системы. А группа технической поддержки может загрузить открытую информацию о дефекте в базу знаний поддержки. Эта информация особенно полезна, если отчет об ошибке включает в себя описание обходных путей. Я работал в проекте, в котором два младших тестировщика потратили две недели, остававшиеся до выпуска версии, на работу с менеджером технической поддержки, инженером по сопровождению базы знаний технической поддержки и несколькими ведущими специалистами службы технической поддержки для того, чтобы убедиться, что последние понимают систему так же, как и тестировщики.

Насколько полезны все эти знания о том, как система может дать сбой? Довольно трудно получить точные цифры. Степень, в которой обходные пути и документированные предупреждения микшируют проблемы, в любом случае трудно измерить цифрами, поскольку заказчик, который решает свои собственные проблемы, применяя файлы помощи или руководство пользователя, не звонит в техническую поддержку, чтобы сообщить, что он решил не беспокоить техническую поддержку, поскольку имеет отменную документацию. Тем не менее скажем, что можно оценить время при наличии обходного пути для проблемы, если заказчик все же обращается в службу поддержки. Если система уже продается на рынке какое-то время, мы можем установить частоту звонков в техническую поддержку по поводу ошибки. (Если это новая система, мы можем на основе анализа рисков качества оценить частоту звонков по определенной ошибке на основе вероятности.) Таким образом, имея некоторые оценки того, сколько обнаруженных дефектов не будут исправлены, мы можем оценить экономию (в человеко-часах) времени работы сотрудника центра обработки вызовов технической поддержки.

Тесты, снижающие риски

К настоящему моменту мы определили выгоду только от найденных ошибок. Мы обнаруживаем множество ошибок и снижаем ущерб посредством некоторой комбинации ремонтных работ, предотвращения появления новых ошибок, обходных путей и предупреждений. Это — управление рисками качества в то смысле, что риски качества после выпуска версии уменьшаются. Мы обнаружили определенные условия, при которых качеству системы создает угрозу определенная, известная опасность. Нам известен вид ошибки, который оказывает влияние на определенный процент пользователей и заказчиков системы в течение определенного времени. Вероятно, мы не знаем эти проценты с большой точностью, однако мы можем оценить средние значения. Это позволяет измерить отдачу инвестиций, как указывалось выше.

Но верно ли, что отдачу приносит только то тестирование, которое ведет к обнаружению ошибок? Или, другими словами, тест, который вы-

полняется без ошибок, тратит впустую деньги компании? Во многих случаях основной целью компании является не столько обнаружение ошибок, сколько уменьшение беспокойства. Это означает, что отдача от тестирования вытекает не столько из знания того, сколько ошибок мы нашли, сколько из знания того, что работает, а что — нет.

Чтобы количественно измерить эту отдачу, можно воспользоваться экономической категорией *замещения* (substitution). Замещение — это использование альтернативного продукта или услуги, приносящей аналогичную пользу. В случае с уменьшением беспокойства обычным решением может стать страхование. (Страхование — это просто статистически обоснованный механизм распространения риска на большое число держателей полисов, т.е., с точки зрения теории, нет оснований полагать, что таким образом распределенный риск не может быть множеством рисков качества в проектах по созданию программных систем.) Во сколько обойдется менеджеру продукта "Суматра" Кейт застраховать проект от затрат на несоответствие?[1]

Предположим, что при выполнении предыдущего проекта, до того как компания Software Cafeteria прозорливо наняла на должность тест-менеджера Джамала, Кейт посетила ближайшего страхового агента. Она сказала: "Знаешь, Джимми, у меня есть проект системной разработки стоимостью в 1 миллион долларов, и я не знаю, каков уровень качества. Совсем. Он может быть как равным 0% удовлетворенности пользователей (все возвращают товар), так и равным 100% удовлетворенности пользователей (все восторгаются товаром). У меня нет группы тестирования, программисты просто тестируют свой код, т.е. никто не обладает информацией, которая мне нужна. Я бы хотела купить у тебя страховой полис, который покрыл бы все мои затраты на несоответствия".

Джимми ответил: "Конечно, Кейт, у меня есть стандартный полис страхования затрат на несоответствия. Подпиши там, где пунктирная линия. Это будет стоить $427 000, включая 10% моих комиссионных сверх 25%-ной надбавки страховой компании XYZ". Как страховая компания XYZ узнала, во сколько необходимо оценить стоимость полиса?

Если мы отбросим наценку (прибыль и накладные расходы) и комиссионные, реальная стоимость страхового полиса окажется равной $311 000. Эта цифра рассматривается как ожидаемая выплата, которую страховая компания предполагает заплатить по полисам, аналогичным данному, проданным большому числу полисодержателей. Страховая компания XYZ могла бы получить это число, если бы располагала статистической моделью, которая лежит в основе кривой затрат на несоответствия, показанной на рис. 4.1, поскольку ожидаемая выплата — это сумма, с учетом всех возможных исходов, произведений вероятности исхода и стоимости исхода. Если предположить, что в отсутствии другой информации о качестве каждый из 11 исходов от 0% до 100% удовлетворенности пользовате-

---

[1] Замещение предполагает одинаковую пользу, а не идентичную замену. Тестирование может дать информацию том, какие части системы работают, а какие нет, о каких рисках стоит беспокоиться, а о каких не стоит. Однако тестирование не возместит затрат компании на несоответствие в результате обнаружения ошибок в период эксплуатации, что под силу страховому полису.

лей одинаково вероятен, то мы вычислим ожидаемые выплаты следующим образом:

$$\frac{1000}{11}+\frac{553}{11}+\frac{437}{11}+\frac{355}{11}+\frac{290}{11}+\frac{236}{11}+\frac{188}{11}+\frac{145}{11}+\frac{106}{11}+\frac{70}{11}+\frac{37}{11}=\$1000\ \ \$311\,000$$

Где можно получить модель затрат и вероятностей различных уровней качества? Купит ли на самом деле кто-нибудь такой страховой полис, если предложить его по честной цене?

Страховые компании размышляют о моделях ожидаемых выплат на основе уровней риска. Для страхования жизни, например, риск связан с определенными результатами медицинских обследований, позволяющими установить состояние здоровья, определить типы поведения, увеличивающие вероятность травм, и возраст. Для страхования автомобиля риск определяется на основе стоимости вашей машины, вашей истории вождения и в некоторых случаях с учетом вашего криминального прошлого.

При разработке системы можно оценить вероятность и затраты определенных исходов, начиная с полного провала проекта и заканчивая удовлетворенностью всех пользователей в отрасли, в которой используется система, учитывая выбранную технологию разработки, уровень зрелости процессов в компании и навыки и умения проектной группы, в частности группы тестирования.

К сожалению, подробных данных в рассматриваемых областях имеется немного, что объясняет причины того, что страховые полисы для проблем в проектах по созданию систем, ориентированных на повышенные запросы к качеству, не являются общедоступным товаром. Любая модель является ограниченной, а при отсутствии данных — очень грубой. Однако, по крайней мере, мы сможем сделать довольно точный анализ в крайних точках. Мы знаем стоимость полного провала — весь проектный бюджет или в некоторых случаях стоимость всей компании. При условии, что мы можем надежно оценить число дефектов в продукте и средние затраты на исправление дефекта до выпуска версии, мы можем обоснованно оценить затраты на несоответствия при 100%-ной удовлетворенности пользователя.

Таким образом, если мы можем определиться с моделью даже при наличии исключительно линейной интерполяции от гибели проекта из-за *цунами* ошибок до полной удовлетворенности пользователя при нулевом количестве дефектов, и группа управления проектом примет эту модель как достаточно точную, значит ли это, что они заплатят за страховку на ее основе? Некоторые компании, вероятно, ответили бы утвердительно. Некоторые компании действительно купили — за очень высокую плату — страховку против компьютерных сбоев из-за проблемы 2000 г. Хотя все закончилось относительно благополучно, многие из тех, кто приобрел страховые полисы, возможно, до сих пор считают это разумным решением. Если бы ситуация складывалась по-другому, а о полисах бы не позаботились, неспособность групп менеджеров управлять определенными классами рисков могла бы привести их в суд, а то и в тюрьму.

В других организациях, особенно в небольших высокотехнологичных компаниях, в которых разработка систем является основным бизнесом, одним из способов получения прибыли является выполнение проектов с высокой степенью риска. Те, кто играл на бирже акций высокотехнологичных компаний в 90-е гг. прошлого века, знают, что большие риски могли привести к большим прибылям. (Так же, как и те, кто держал такие акции в 2000 – 2002 гг., знают, что те же решения могли вести к большим потерям.) Некоторые люди решают не приобретать полис страхования жизни, другие считают это разумной инвестицией. Компании ведут себя так же. Сколько они заплатят за управление рисками качества, зависит в большой степени от того, насколько они болезненно относятся к рискам.

Ведение проекта

Если тщательно проанализировать фразу *"управление рисками качества"* (quality risk management), можно обнаружить, что это также означает способность принятия обоснованного решения. Словарь *Merriam-Webster's "Collegiate Dictionary"* определяет слово "managing" (управление) как "контроль или руководство с определенным навыком", а "навык") – как "способность легко и эффективно использовать чьи-либо знания". Другими словами, для того чтобы действительно управлять рисками качества, нужно не только держать под контролем известные источники опасности, но и добывать информацию об общем качестве системы, и получать некоторое представление о том, что это может значить для системы в будущем.

Снова обратимся к рис. 4.1, уделив особое внимание кривой затрат на несоответствия. Итак, эта кривая является некоторой аппроксимацией, которая не подлежит непосредственному использованию ни в какой компании, но общая форма – убывание с повышением качества – применима повсеместно.

Большинство менеджеров проектов, с которыми мне довелось работать, понимали, что качество системы влияет на общую стоимость разработки, сопровождения или поставки системы в течение всего ее жизненного цикла. Поскольку часть менеджмента всегда занимается управлением затрат, эти менеджеры стремятся проникнуть в суть качества. Без точных, заслуживающих доверия и своевременно полученных результатов тестирования менеджеры проектов знают очень немного об уровне качества своих систем. Более того, они еще меньше понимают, как эволюционирует качество их систем, по мере того как проект приближается к важной контрольной точке, например первоначальной сборке или версии сопровождения.

Тестирование может позволить заглянуть внутрь качества. Чтобы это сделать, мы должны сначала понять общую картину рисков качества, построить окружение для тестирования этих рисков, выполнить тесты, чтобы оценить качество, а затем довести полученную в ходе тестирования информацию эффективно, точно, надежно и своевременно. Таким образом, менеджеры проектов могут узнать, что не работает, что работает, а что остается за рамками исследований тестировщиков. Хорошо организованный проект тестирования может также собрать информацию о тенденциях, позволяющую сделать осмысленные предсказания уровней ка-

качества при том же уровне представления о них, какой финансовые отделы имеют о рисках бюджета, отделы маркетинга — о рисках свойств системы, подразделения менеджеров проектов — о рисках сроков. Хотя есть еще непочатый край работы с моделями, которые позволили бы численно оценить точность оценки качества, имеет смысл включить некоторые оценки этой величины как часть анализа возврата вложений в тестирование.

Заключительное замечание об оценках возврата вложений

Цель проведения анализа возврата вложений состоит в показе бизнес-плана тестирования. В некоторых случаях, вероятно, есть смысл представить бизнес-план для проведения тестирования другими способами. Например, Фьюстер (Fewster) и Грэм (Graham) показывают построение бизнес-плана для автоматизации тестирования в "*Software Test Automation*".

Бизнес-план для автоматизации тестирования выявляет следующий полезный факт. Некоторые затраты могут быть разнесены по нескольким проектам. Это правило применимо не только к скриптам для автоматизированного тестирования, но и ко всей системе тестов, включая тестовую среду, затраты на ее конфигурирование, тестовые данные, многократно используемые тестовые сценарии для ручного тестирования, некоторые другие объекты. Нужно быть реалистами, но тем не менее иметь в виду это обстоятельство при оценке возврата инвестиций в тестирование.

Что касается данного замечания, важно иметь сдержанные оценки возврата вложений. Если вы переоценили свой случай и привели производящие впечатление цифры, и какой-либо менеджер вас поймал на этом, вы нанесли тем самым реальный ущерб доверию вам со стороны руководства. Следует опираться на источники, а не брать цифры с потолка. Лучший источник данных — ваш опыт и ранее выполненные проекты, поскольку эти данные наилучшим образом проанализированы и осознаны. В противном случае можно опираться на данные в доступных публикациях (например, в "*Estimating Software Costs*")) или на информацию от коллег. Однако, применяя данные, полученные от других людей, необходимо потратить время, чтобы понять контекст, из которого взяты эти данные, и как различие между этим контекстом и вашим окружением тестирования влияет на применимость данных для ваших целей. Наконец, оцените с точки зрения здравого смысла полученные оценки возврата вложений и будьте готовы

Амортизация (Amortize)

Распределение затрат и прибыли от инвестиций на серии временных периодов (например, ежемесячно) либо на совокупность проектов. Показывает средние затраты и среднюю прибыль от инвестиций на период времени или на проект.

чества в будущем, а также определить примерное время готовности системы к выпуску[1].

Поскольку мы можем обсуждать проблемы не только в терминах ошибок и тестовых сценариев, но и в терминах "опыта качества" пользователя, эта оценка позволяет понять менеджменту, каков уровень качества системы. Как можно количественно измерить, насколько значимо это понимание для менеджмента? Попробуем это сделать, рассмотрев следующие варианты.

Начнем с замечания, что наличие точных, своевременных, заслуживающих доверия и исчерпывающих оценок и прогнозов о качестве системы является фундаментальной частью отслеживания хода проекта. В книге "*Estimating Software Costs*" Кейперс Джоунз (Capers Jones) выделяет четыре основные корневые причины неудач проектов, одна из которых – небрежное отслеживание состояния проекта[2].

Джоунз сообщает, что выбор более надежных способов отслеживания хода проекта может снизить риск неудачи проекта примерно на 50%. (Он определяет неудачу проекта как нарушение сроков или бюджета не менее чем на 25% от плановых либо как полную остановку проекта.) Джоунз также говорит о том, что риск неудачи проекта в целом есть функция от величины проекта, изменяющаяся от 2% для очень маленьких проектов до 85% для очень больших проектов. Предположив что-то среднее, будем считать, что риск неудачи проекта 40% в случае плохого отслеживания его хода. Только улучшение отслеживания проекта снизило бы этот риск до 20%. Если результаты тестирования привносят только половину этой цифры в улучшение отслеживания проекта, то группа тестирования отвечает за 10%-ное снижение риска неудачи проекта за счет оценки и прогнозирования уровня качества системы и предоставления этой информации группе управления проектом.

Насколько значимо для проекта 10%-ное снижение риска? Это зависит от проекта. Если мы говорим о проекте стоимостью в миллион долларов, то это $100 000. (Заметьте, что мы снова применяем расчет ожидаемых выплат, как описано выше.) Эта цифра, очевидно, представляет грубую оценку. Даже Джоунз предупреждает о большом разбросе цифр, на которые он ссылается. Однако мое мнение о цифрах возврата вложений, полученных на основе предположений Джоунза, таково, что они обычно недооценивают значение результатов тестирования.

В конечном счете, информация, которую может предоставить группа тестирования в ходе доведения результатов оценки качества, имеет исключительное значение для компании. Я подозреваю, что значение этой информации может превышать значение информации, представленной в виде обнаруженных и исправленных до выпуска версии дефектов, поскольку она позволяет руководству принимать решения о рисках

---

[1] Более подробно о сборе, интерпретации и передаче результатов тестирования см. в главе 15. Также можно обратиться к главам 4 и 5 книги "*Managing the Testing Process, Second Edition*".

[2] Эта информация впервые появилась в "*Patterns in Software System Failure and Success*". В главе 7 книги "*Estimating Software Costs*" резюмируются результаты работы Джоунза. Оставшиеся три причины – это неточная предварительная оценка затрат, плохое планирование и недостаточное тестирование. Что касается влияния этих причин, особенно последней, они уже рассматривались выше.

говорить о деталях. Если вы намерены утверждать, что любая ошибка, подлежащая обязательному устранению, стоит $1500, если ее нашли в процессе эксплуатации системы, потрудитесь объяснить, почему.

Джамал готовится выставить счет и его обоснование

Изучив вопрос о возврате вложений в тестирование, Джамал вернулся к подготовке бюджета группы тестирования. Вооружившись декомпозицией работ, Джамал взялся в первую очередь за кадровые вопросы. Он решил использовать стратегию назначений в группе тестирования, основанную на плане-графике проекта. Таким образом, Лин Цу, Эмма, новый тестировщик и новый лаборант будут назначены в проект на полную загрузку, как только они должны будут приступить к своим обязанностям согласно плану-графику[1]. Поскольку у Джамала есть другие обязанности, в частности следить за тем, что тестирование версии сопровождения для других продуктов идет достаточно быстро, он выделил себе 50%-ную загрузку в проекте. Он также предусмотрел расходы на командировку, поскольку собирался нанести визит в лабораторию тестирования локализации, которая находится в другом штате.

Пока Джамал думал о лаборатории локализации, он вспомнил, что, проводя анализ покрытия, он отправил электронное сообщение коллеге, с которой познакомился на конференции по тестированию. В письме он спрашивал ее о квоте тестирования удобства использования. Он включил эту квоту в бюджет. Джамал не был уверен, что группа тестирования сможет провести тестирование удобства использования, поскольку очень большая часть функционального тестирования будет автоматизирована. (Будучи весьма эффективным, автоматизированное функциональное тестирование часто означает, что тестировщики имеют меньше времени на то, чтобы "потрогать руками" взаимодействие функциональных областей системы, что является ключевым при поиске проблем удобства использования.) Более того, ни один человек в группе тестирования не имеет достаточного представления об удобстве использования. Конечно, он всегда обращается к тестировщикам с просьбой "влезть в шкуру разумного пользователя" при выполнении тестов, но это не то же самое, что иметь соответствующего эксперта.

Исследуя аппаратное обеспечение для тестовой среды, он включил в список новые серверные кластеры (web-сервер, сервер приложений и сервер баз данных), а также 10 новых рабочих станций. Что касается программного обеспечения, ему также требуются лицензии для серверов, но это обычно не нужно включать в бюджет, поскольку Software Cafeteria уже имеет достаточно лицензий по договорам с различными производителями программ. Наконец, по инструментарию Джамалу требовалась еще одна лицензия на инструменты автоматизации тестирова-

---

[1] См. главу 8 моей книги "*Managing the Testing Process, Second Edition*".

ния графического интерфейса и нагрузочного тестирования, поэтому он добавил эти две лицензии в список.

№ этапа	Этап	Выполнено?
2.A	Извлечение из декомпозиции работ полного списка ресурсов. Определение для каждого ресурса первого и последнего дня работы в проекте. В случае если есть ресурсы, занятые сразу в нескольких проектах, уточнить процентную загрузку каждого из таких ресурсов в каждом из проектов в течение различных периодов времени.	☑
2.B	Выявление необходимых дополнительных ресурсов для поддержки ресурсов, занятых сразу в нескольких проектах.	☑
2.C	Распределение ресурсов по категориям: персонал, командировки, инструментарий, тестовая среда и, если есть, затраты на аутсорсинг. Суммирование по периодам времени и категориям.	☑

Джамал посмотрел на первый черновик своего бюджета и понял, что забыл добавить 20% на непредвиденные обстоятельства, что было его обычной практикой. С этой добавкой общий бюджет тестирования составил более $800 000 (см. рис. 4.2). По сравнению с предыдущими проектами он был довольно велик. Однако он посмотрел на среднемесячные затраты, которые составили $119 000. Эта цифра оказалась лишь чуть больше, чем в предыдущих проектах. "Это просто еще одно проявление раннего вовлечения тестировщиков в проект и более интенсивного тестирования в случае инкрементного подхода к разработке", – думал он.

	SpeedyWriter Release 3.1 (w/DMS) IST Budget							
	Sept	Oct	Nov	Dec	Jan	Feb	Mar	Total
Lin-Tsu Wu (MTE)	4,531	9,063	9,063	9,063	9,063	9,063	9,063	$58,909
Emma Moorhouse (ATE)	5,094	10,188	10,188	10,188	10,188	10,188	10,186	$66,222
TBH (ATE)	0	10,000	10,000	10,000	10,000	10,000	10,000	$60,000
Technicians	0	30,800	30,800	30,800	30,800	30,800	23,100	$177,100
Total Staff	**$9,625**	**$67,550**	**$67,550**	**$67,550**	**$67,550**	**$67,550**	**$59,850**	**$410,975**
Travel Allowance	**$0**	**$0**	**$0**	**$0**	**$2,000**	**$2,000**	**$0**	**$4,000**
Tools								
GUI Functional	$10,000	$0	$0	$0	$0	$0	$0	$10,000
Load Testing	10,000	0	0	0	0	0	0	$10,000
Total Tools	**$ 0,000**	**$0**	**$0**	**$0**	**$0**	**$0**	**$0**	**$20,000**
Test Systems								
Web Servers	$50,000	$0	$0	$0	$0	$0	$0	$50,000
App Servers	50,000	0	0	0	0	0	0	50,000
Database Servers	50,000	0	0	0	0	0	0	50,000
Clients	25,000	0	0	0	0	0	0	25,000
UPSes	6,000	0	0	0	0	0	0	6,000
Miscellaneous Supplies	1,000	0	0	0	0	0	0	1,000
Total Test Systems	**$182,000**	**$0**	**$0**	**$0**	**$0**	**$0**	**$0**	**$182,000**
External Labs								
Localization	$0	$0	$0	$0	$25,000	$25,000	$0	$50,000
Usability	0	0	0	0	0	25,000	0	25,000
Total External Labs	**$0**	**$0**	**$0**	**$0**	**$25,000**	**$50,000**	**$0**	**$75,000**
Contingency	**$43,075**	**$13,510**	**$13,510**	**$13,510**	**$18,910**	**$23,910**	**$11,970**	**$138,395**
Grand Total	**$258,450**	**$81,060**	**$81,060**	**$81,060**	**$113,460**	**$143,460**	**$71,820**	**$830,370**
Average Monthly Cost								**$118,624**

Рис. 4.2. Первый черновик бюджета тестирования

Полные затраты на штатную единицу (Fully Burdened Staff Rate)

Затраты на одну штатную единицу, включая заработную плату, премии, затраты на инфраструктуру и прочие связанные затраты на персонал.

Джамал не видел бюджета Дженни, но он знал, что в ее группу назначены 11 программистов и два менеджера. При средней ставке технического персонала с учетом всех затрат в $120 000 в год только затраты на персонал разработчиков примерно равен всему бюджету Джамала. Просматривая некоторые ранее выполненные бюджеты разработки, он прикинул, что Кейт ожидает, что его бюджет будет в пределах от одной четверти до одной трети всего бюджета проекта. "Я намереваюсь слегка превысить верхнюю границу", – думал он, поэтому мне следует подготовиться к защите бюджета.

№ этапа	Этап	Выполнено?
2.D	Проверка, с точки зрения здравого смысла, бюджета и общих сумм. По возможности сопоставить профессиональные знания и интуицию с данными предыдущих проектов, измерениями в отрасли и т.д., выявить и, если возможно, разрешить расхождения между бюджетом подпроекта тестирования и общим бюджетом проекта. Там, где это невозможно, документировать проблемы.	☑

Первым шагом решения этой задачи могла бы стать количественная оценка возврата вложений в тестирование. Просматривая бюджет, он понимал, что сможет амортизировать инструментарий и экземпляры тестовых сред в течение следующих 18 месяцев, в течение которых будут выполнены три проекта, включая нынешний. Кроме того, он предполагал списать работы по созданию тестов и конфигурированию оборудования для тестирования, каждая из которых потребует около двух человеко-месяцев работы Эммы, Лин Цу и нового тестировщика. Это составило бы около $60 000, и Джамал исключил бы все, кроме $20 000, из анализа возврата вложений.

№ этапа	Этап	Выполнено?
2.E	Уточнение амортизации объектов, которые являются предметом долговременных инвестиций, документирование возможностей их повторного использования и определение периода, в течение которого планируется возместить затраты.	☑

Чтобы проанализировать возврат вложений для обнаруженных и исправленных ошибок, Джамалу необходима оценка числа ошибок. У него нет точных данных о размере системы, например о количестве строк кода или функциональных точек, поскольку Software Cafeteria не пользуется такой статистикой. Тем не менее после некоторого исследования ранее вы-

полненных проектов и базы ошибок периода эксплуатации он выяснил, что в среднем в системе обнаруживалось 35 ошибок на человеко-месяц работы программиста, что включало ошибки, найденные как в ходе тестирования, так и после выпуска версий. В проекте «Суматра» запланировано 75 человеко-месяцев работы программистов. Чтобы оценка была умеренной, Джамал принял 2500 за ожидаемое число ошибок. Поскольку примерно 80% ошибок будут исправлены в ходе жизни проекта, он запланировал, что 2000 ошибок подлежат обязательному исправлению.

Из предыдущих проектов Джамал знал, что в среднем группе тестирования удается найти около 80% таких ошибок в системе. (Пользователи обычно находят оставшиеся 20% и сообщают о них группе технической поддержки.) Он надеялся, что удастся сработать лучше в новом, не столь спешном режиме, но он решил не учитывать эти ожидания в анализе возврата инвестиций. Таким образом, планировалось найти 1600 дефектов, подлежащих обязательному исправлению. Для быстрой проверки он умножил это число на средние затраты на обнаруженную ошибку, подлежащую обязательному исправлению, в его предыдущих проектах – $500. Он с радостью обнаружил, что цифры совпали. Таким образом, можно было считать, что будет обнаружено 1600 ошибок, подлежащих обязательному исправлению.

Цифра $500 на ошибку, подлежащую обязательному исправлению, учитывает все инвестиции в тестирование и затраты на обнаружение ошибок и затраты на внутренние несоответствия, связанные с тестированием. Теперь Джамал решил подсчитать средние затраты на внутренние и внешние несоответствия для дефектов.

Чтобы подсчитать затраты на внутренние несоответствия для группы тестирования, он сначала оценил повторное тестирование. Он запланировал 11 тестовых раундов. В шести из них будет проверяться новая функциональность в рамках очередного инкремента. Следовательно, пять тестовых раундов будут посвящены повторному тестированию для проверки исправления ошибок и регрессии. Продолжительность тестирования определена в 22 недели. На основе этих расчетов затраты на персонал на неделю тестирования примерно равны $17 000. Это означает, что суммарные затраты на тестирование внутренних несоответствий составляют примерно $170 000, причем затраты на тестирование для каждой ошибки, подлежащей обязательному исправлению, равны $105 (см. рис. 4.3).

Что касается разработки, Джамалу было известно от Дженни, что в течение 22 недель тестирования она планирует, что половину времени группа тестирования будет тратить на исправление ошибок, а другую половину на работу над новыми свойствами очередного инкремента. Используя среднюю ставку в $120 000 в год на 13 человек группы разработки, Джамал получил $215 затрат разработчиков на работу с одним дефектом, подлежащим обязательному исправлению. Таким образом, полные затраты на внутренние несоответствия на дефект достигают $320.

После выпуска версии в продажу затраты переносятся на программистов сопровождения, которые будут исправлять ошибки и усовершенствовать программу, и на сотрудников группы технической поддержки, которые будут отвечать на звонки, относящиеся к ошибкам в системе и ошибкам пользователей. Из разговоров с менеджерами этих подразделений Джамалу было известно, что около 75% времени в каждой группе

	A	B	C	D	E
1	**Internal Failure**			**External Failure**	
2					
3	**Test Team Retest Costs**			**Maintenance Team Fix Costs**	
4	Retest Proportion	45%		Fix Effort Percentage	75%
5	Test Weeks	22		Maintenance Weeks	52
6	Test Staff Cost/Week (7.5 People)	$ 16,868		Developer Staff Cost/Week (2 People)	$ 4,800
7	Test Cost of Internal Failure	$ 168,679		Maintenance Cost of External Failure	$ 187,200
8	Test Cost per Bug	$ 105		Maintenance Cost per Bug	$ 468
9					
10	**Development Team Fix Costs**			**Technical Support Team Costs**	
11	Fix Effort Percentage	50%		Bug Effort Percentage	75%
12	Test Weeks	22		Support Weeks	52
13	Development Staff Cost/Week (13 People)	$ 31,200		Technical Support Staff Cost/Week (4 People)	$ 9,600
14	Development Cost of Internal Failure	$ 343,200		Technical Support Cost of External Failure	$ 374,400
15	Development Cost per Bug	$ 215		Technical Support Cost per Bug	$ 936
16					
17	**Cost of Internal Failure**	**$ 320**		**Cost of External Failure**	**$ 1,404**
18					
19	**Cost Savings Per Bug Found in Testing**				**$ 1,084**

Рис. 4.3. Подсчет затрат на внутренние и внешние несоответствия в проекте "Суматра")

тратится на работу с ошибками системы. После выпуска версия поддерживается в течение года. Основываясь на размерах групп и ставках персонала, Джамал смог оценить затраты на внешние несоответствия в $1404 на один дефект, подлежащий обязательному исправлению.

Таким образом, каждая ошибка, подлежащая обязательному исправлению и обнаруженная в ходе тестирования, экономит компании примерно $1084. Подставляя в уравнение стоимости качества, получаем прибыль в размере около 1.7 миллиона долларов. "Неплохо, — подумал он, — мы уже в плюсе при возврате вложений в тестирование, а еще не все подсчитано".

Теперь рассмотрим оставшиеся 500 (или около того) найденных и не исправленных ошибок. Обычно около трети — это ошибки тестировщиков, ошибки в конфигурации системы и другие проблемы, не являющиеся ошибками системы. Остается примерно 300 ошибок, являющихся действительно проблемами системы, которые по разным причинам были отложены. В результате обсуждения с менеджером группы поддержки пользователей Косимо Неграевым выяснилось что, наличие информации об этих ошибках в базе знаний группы поддержки обычно экономит до 15 минут на звонок. Более того, Косимо упомянул, что на каждую ошибку в системе обычно приходится пять звонков. Снова используя среднюю ставку в $120 000 в год, Джамал получает $60 в час. Таким образом, выигрыш по обнаруженным, но неисправленным ошибкам составляет примерно $25 000.

Далее Джамал обратил внимание на значение снижения риска. В процессе анализа рисков качества участники определили вероятность возникновения проблем как очень высокую в следующих категориях рисков качества:

- Нагрузка, производительность, объем
- Надежность и стабильность

Из разговора с Мухаммедом Аджами, отвечающим за продажи компаниям Fortune 1000 и государственным организациям, Джамал узнал, что проблемы в этих областях системы могут обойтись, по крайней мере, в 2 миллиона долларов дохода от продаж. Крупные заказчики наиболее болезненно воспримут эти проблемы и не будут их терпеть. С учетом 50%-ной вероятности проблем такого рода снижение риска только их возникновения с помощью тестирования принесет 1 миллион долларов.

Наконец, Джамал рассмотрел прибыль, которую приносит отслеживание хода проекта. Прочитав книгу Джоунза, он решил воспользоваться эмпирическим правилом "10% бюджета проекта". Очень грубая оценка его доли бюджета проекта в 40% дает общую стоимость проекта 2 миллиона долларов. Информация о проверке системы и прогнозы, которые группа тестирования планирует предоставлять группе управления проектом, снижая риск неудачи проекта на 10%, приносит $200 000 прибыли.

Таким образом, Джамал получил суммарный доход от предлагаемой им программы тестирования примерно 3 миллиона долларов, 350% возврата вложений в тестирование (см. рис. 4.4). Хотя, согласно корпоративным правилам Software Cafeteria, тест-менеджеру запрещено обсуждать цифры бюджета с сотрудниками группы тестирования, он чувствовал себя очень комфортно, сумев обосновать умеренный, но верный возврат вложений в тестирование в проекте. Он с нетерпением ожидал возмож-

Sumatra Test ROI

Testing	Costs	Benefits
Staff (test development amortized)	$ 370,875	
External Labs	75,000	
Travel and Contingency	142,395	
Environments (amortized)	60,667	
Tools (amortized)	6,867	
Total Investment	**$ 655,793**	
Defect Estimates		
Estimated Bugs Delivered to Test	2,500	
Estimated Must-Fix Bugs Delivered to Test	2,000	
Estimated Defect Detection Percentage	80%	
Estimated Found-Not-Fixed Bug Rate	20%	
Estimated Found-Not-Fixed Bugs	500	
Estimated Found-Not-Fixed Non-Bug Rate	67%	
Bugs Found and Fixed		
Must-Fix Bugs Found	1,600	$ 1,734,400
Bugs Found and Not Fixed		
Deferred Bugs Found	333	$ 25,000
Tests that Mitigate Risks		
Sales Subject to LCV or Reliability Loss	$ 2,000,000	
Likelihood of Sales Loss w/o Testing	50%	$ 1,000,000
Guiding the Project		
Project Cost at Risk for Bad Tracking	$ 2,000,000	
Reduction in Risk due to Testing	10%	$ 200,000
Total Benefits		**$ 2,959,400**
Return on Investment		**351%**

Рис. 4.4. Расчет возврата вложений в тестирование для первого варианта бюджета Джамала

ности представить обоснование бюджета руководству проекта. Он был счастлив получить разрешение двигаться дальше.

№ этапа	Этап	Выполнено?
2.F	Если это необходимо или желательно, провести анализ возврата инвестиций. Если сальдо отрицательное, проверить допущения об амортизации, сделанные на шаге 2.E, и при необходимости повторить его. Если сальдо по-прежнему отрицательное, проанализировать те компоненты декомпозиции работ, которые требуют наибольших затрат с наименьшей отдачей, сверившись с рисками качества. Документировать активности, приводящие к потере средств.	☑
2.	Использовать декомпозицию работ и план-график для подготовки бюджета.	☑

Всегда ли быстрый побеждает медленного?

Всегда ли сроки на рынке побеждают качество? До возникновения бума дот-комов в 2000 г. многие говорили о времени Интернета часто в связи с вбрасыванием версий неизвестного качества во Всемирную паутину, которые подвергались лишь незначительному тестированию или не тестировались вовсе. Однако очень опасно придерживаться догматичной и негибкой позиции, считая, что риски, связанные со сроками, всегда перевешивают риски качества на рынке программного и аппаратного обеспечения. Позвольте мне рассказать следующую историю.

Рынок карманных персональных компьютеров (КПК) уже огромен, и ожидается, что будет расти дальше. Одним из первых участников на этом рынке была компания Apple Computer, выпустившая Newton несколькими годами раньше, чем появился Palm Pilot. Если непреложным правилом является победа скорости над качеством на рынке, Apple должна была занять весь рынок КПК. Почему этого не произошло?

Когда появилась первая версия Newton, она имела серьезные проблемы, связанные с возможностями распознавания почерка. Так случилось, что один из экземпляров Newton приобрел Гарри Трюдо (Garry Trudeau), автор комиксов "Doonesbury", печатающихся во многих изданиях Северной Америки. Он высмеивал слабости в распознавании почерков в Newton в большой серии комиксов в течение нескольких недель.

В этой главе мы рассмотрели способ учета затрат, связанных с таким видом ущерба репутации компании. Эти затраты в основном относятся к потерям от снижения продаж из-за несоответствия требованиям пользователей. Если компании утверждают о наличии реализованных требований в продуктах, которым эти продукты визуально или, что еще хуже, комически не соответствуют, компании рискуют повторить опыт Apple. Сложно оценить количественно затраты на несоответствия для внешних проблем Newton, но примерную цифру можно было бы получить, исходя из представлений о том, какие ежегодные финансовые показатели характеризовали бы ситуацию, будь Apple лидером продаж КПК.

Двигаясь вперед, оглянись назад

В этой главе очерчен, а затем показан на примере Джамала способ трансформирования декомпозиции работ в бюджет и анализа возврата вложений, соответствующего предлагаемым затратам на тестирование. Теперь читатель получил обоснованную оценку времени и ресурсов, направленных на контроль критических рисков качества, наряду с обоснованием выбранных действий не только в терминах качества, но и с точки зрения финансов. Остается только положить весь пакет полученных документов на стол менеджера проекта и ожидать положительной оценки деловой хватки и немедленного предоставления всех запрошенных ресурсов, не так ли? К сожалению, обычно это далеко не так.

В следующей главе мы завершим обсуждение оценок и перейдем к процессу обсуждения бюджета. Подготовлено сильное предложение, но его еще нужно "продать" руководству. Надо убедить их захотеть приобрести предлагаемые вами продукты и услуги по управлению рисками качества и преодолеть неизбежные возражения относительно сроков их предоставления и соответствующих затрат. После завершения обсуждения бюджета будет показана ретроспектива процесса оценки затрат. Прежде чем перейти к разработке планов тестирования, мы должны посмотреть на показатели хороших оценок затрат на тестирование, проблемы при подготовке таких оценок и планы развития системы, которые можно применять для продвижения процесса оценки затрат.

ГЛАВА 5

От оценки затрат к базовой версии: получение одобрения реалистичных, работоспособных и правдивых оценок

В этой главе завершается обсуждение оценки затрат на проект тестирования. Сначала мы посмотрим на то, что является, по всей вероятности, самым сложным этапом процесса — на получение одобрения подготовленных оценок затрат у руководства. Чтобы уговорить менеджеров выделить время и деньги на достойное тестирование, необходимы доверие и дар убеждения. Правдоподобные оценки и убедительное описание бизнес-плана являются ключевыми частями решения проблемы. Этапы, показанные в двух предыдущих главах, дают некоторые идеи о том, как получить эти части. Однако, когда вы представляете оценки руководству проекта, проверке подвергаются только ваш политический капитал и навыки.

После завершения исследования этого непростого процесса мы снова посмотрим на весь процесс оценки затрат. Как узнать, что оценка затрат сделана успешно? Какие факторы способствуют получению хороших оценок? Какие проблемы существуют? И как можно ввести в практику более удобные и эффективные методы оценки затрат? Ответы на эти вопросы будут даны во второй части главы.

"Продажа" оценки затрат

Получение хорошей, надежной оценки, включая окупаемость вложений, — непростая задача, получение поддержки руководства — задача другого рода. Даже наилучшая оценка затрат подпроекта тестирования с наиболее основательным бизнес-планом может столкнуться с препятствиями. Некоторые из этих препятствий заставят уменьшить исходные границы тестирования. Иногда этому нет альтернативы, поскольку ком-

пания действительно не может позволить себе покрыть все риски качества системы, которые в ходе анализа рисков определены как важные. Итак, рассмотрим некоторые из препятствий, с которыми мы сталкиваемся, а затем исследуем, какими способами их можно преодолеть.

Позвольте начать с препятствий, которые мы создаем сами как тестировщики. Мы обычно смотрим на тестирование и качество как на действительно важные вещи. За годы работы некоторые постулаты, связанные с тестированием и качеством, стали настолько очевидными для нас, что это оказывает влияние на способность взаимодействия с теми людьми, которые не смотрят на проект теми же глазами. В список этих людей попадают менеджеры, которые принимают решение о бюджете тестирования. В результате мы не можем разговаривать с руководителями на одном языке. Как пишет Джеймс Баллок (James Bullock): "Когда мы пытаемся обосновать тестирование, то часто приводим аргументы, исходя из неверного представления, что тестирование — *это самое правильное, что нужно делать*"[1].

Мы рассматриваем тестирование как важную часть разработки качественного продукта. Однако значение тестирования не всегда очевидно менеджеру. Это особенно верно в том случае, когда менеджер видит, что предлагаемый бюджет съедает существенную часть денег, необходимых на весь проект, а график тестирования откладывает дату выпуска продукта. Еще хуже, когда некоторые тестировщики реагируют на проблемы, связанные с потребностью тестирования, исходя из того, что никто, кроме них, не отстаивает качество системы. Такое отношение оказывает непоправимый вред их усилиям по подготовке и защите бюджета тестирования.

Чтобы эффективно взаимодействовать с менеджером проекта и старшими менеджерами, я пытаюсь увидеть, как тестирование встраивается в объемлющий проект. Не считая себя единственным сторонником качества, я обрисовываю свою роль как специалиста по тестированию. Я оказываю помощь моим коллегам по управлению рисками качества системы, которую мы разрабатываем, сопровождаем или поставляем. Какие из услуг управления рисками качества и сведения о рисках имеют ценность для моих коллег? Могу ли я думать о способах передачи информации о том, почему указанные продукты и услуги будут для них более ценными, чем им кажется, в терминах, которые поймут мои менеджеры? Могу ли я убедить людей в необходимости тестирования с точки зрения бизнеса?

Как только вы начинаете смотреть на тестирование в более широком контексте, реалии бизнеса в тестировании становятся яснее, раскрывая следующую группу препятствий. "Хорошо, — может сказать старший менеджер, — тестирование полезно, но посмотри, какие будут затраты и сроки". В зависимости от проекта тестировщики составляют от 15 до 50% всего персонала. Тестовые среды, копирующие среду пользователя, что существенно для системного тестирования, могут быть большими, доро-

---

[1] Эта и другие мысли найдены в статье Джеймса Баллока «Calculating the Value of Testing», опубликованной в "*Software Testing and Quality Engineering*", Volume 2, Issue 3 (May/June 2000) и доступной в настоящее время на www.stickyminds.com.

гими и трудными в сопровождении. Каждый тестовый цикл, кроме первого, откладывает поставку системы. Эти задержки возмущают руководство, поскольку для проектов время — деньги: опоздания с поставкой означают штрафы по контрактам, упущенные возможности, связанные с несвоевременным выходом продукта на рынок и т.д.

В некоторых компаниях менеджеры высшего звена могут увидеть политический ущерб в предложении отложить сроки или увеличить бюджет. Начальные план и бюджет, даже если они выглядят скороспелыми и скудными для тестировщиков и других сотрудников, выполняющих реальную работу, могут быть обеспечены только при условии наличия здравого смысла у части группы управления проектом. Меня беспокоит вариант, когда менеджер проекта начинает свирепеть при защите разумных вложений в тестирование. Я понимаю, что мне нужно предвидеть заранее, будет ли моя оценка затрат, если ее утвердят, вынуждать руководство проекта вновь открывать дискуссию о сроках и бюджете проекта. Если это так, я добавляю дополнительные обоснования, чтобы не допустить новых изменений.

Большинство менеджеров проектов — это опытные профессионалы. Они обычно находятся в гуще системных проектов и могут вспомнить плановую короткую фазу тестирования, которая увеличилась на неделю, потом на две, потом на четыре и так далее, каждый раз усиливая панические настроения. Следовательно, опытный менеджер проекта будет изучать план-график и бюджет в поисках того, что ему нужно: заниженные ограничения, минимальные вложения в тестирование, небольшая начальная цена[1].

"Подождите, — скажете вы, — не вы ли всю прошлую главу рассказывали о том, что разумно проводимое тестирование дает положительное сальдо на инвестиции в него? И о том, что качество бесплатно?" Да. Но это верно лишь в долгосрочной перспективе, в ходе всего жизненного цикла системы. В то же время затраты на разработку, сопровождение и поставку системы обычно финансируются, исходя из проекта, а не из жизненного цикла системы. На любой проект выделяется ограниченный объем средств. Следовательно, с одной стороны, инвестиции в тестирование имеют положительное сальдо, но, с другой стороны, они также представляют, как и любые инвестиции, неизбежные расходы.

В некоторых случаях мы должны убеждать менеджеров, которые не разбираются в тестировании или не знакомы с успешно идущими проектами тестирования. Это порождает дополнительные препятствия для получения одобрения бюджета руководством. Во-первых, нам приходится знакомить менеджеров с тем, в чем ценность тестирования для проекта. Во-вторых, у нас мало времени, чтобы это сделать, поскольку большинст-

---

[1] В 1975 г. Фред Брукс (Fred Brooks) писал в "*Мифическом человеко-месяце*": «Особенно катастрофические последствия может иметь недостаток времени для системного тестирования. Поскольку задержка происходит в конечной части графика, никто не подозревает о том, что график находится под угрозой срыва вплоть до дня сдачи продукта. Плохие вести, полученные поздно и без предупреждения, обескураживают менеджеров и имеют особенно тяжелые материальные и психологические последствия. Проект осуществляется при полной укомплектованности работниками и максимальных финансовых издержках. Еще важнее, что вторичные издержки могут быть выше, чем все прочие. Поэтому очень важно в изначальном графике работ отвести достаточно времени для системного тестирования».

во менеджеров проектов заняты на бесконечных совещаниях, где они занимаются тем, что вводят друг друга в заблуждение. В-третьих, некоторые тест-менеджеры считают трудным объяснение особенностей и реальностей тестирования людям, которые не знакомы с кухней проекта тестирования. Как тестировщики-профессионалы мы сконцентрированы на растущей, изменяющейся и приводящей нас в восхищение профессии в рамках разработки системы. Это означает, что многие из наших коллег, включая тех, кто руководит нами и утверждает наши запросы ресурсов, не могут уследить за всеми изменениями в теории и практике тестирования. В результате они часто не могут понять детали того, что мы делаем и почему это делаем, или, другими словами, динамику процесса тестирования.

Как можно преодолеть эти препятствия? Ответ зависит от конкретной ситуации, поскольку это вопрос, связанный исключительно с "продажей" бюджета. Как продать что-нибудь? Это зависит не только от продукта или услуги, но и от покупателя, а все они разные. Однако весь процесс прост: сделать продукт или услугу желанными для человека, которого вы пытаетесь убедить потратить деньги, а затем работать с его возражениями. Я обычно использую следующий метод.

Сначала я пытаюсь продать мои оценки на основе рисков. Опираясь на основательный анализ рисков качества, я могу создать мощную защиту своих предложений, поскольку могу увязать их с вопросом о том, что могут означать для бизнеса проблемы в период эксплуатации. Каждая задача в плане-графике, каждый доллар в бюджете сопоставляется определенному критическому риску качества. Что я стараюсь сделать, так это показать очевидную ценность тестирования: "Если мы беспокоимся об этом риске, вот сколько будет стоить его смягчение с помощью тестирования". Опишите в общих чертах катастрофические последствия для прибыли, целей компании, даже жизни, которые можно предотвратить до выпуска версии, особенно для дорогих бюджетных единиц, например серверов или сотрудников. Не втянута ли компания-конкурент или другая компания, работающая в аналогичной отрасли и развернувшая похожую систему, в судебные разбирательства из-за проблем качества системы?

Кроме того, поскольку я прошу кусок уже готового и неизменного пирога — часть бюджета всего проекта, который часто зафиксирован к этому моменту, — я не могу делать это в вакууме. Если я попытаюсь это сделать, то ничего не выиграю в соревновании с менеджерами моего уровня. Таким образом, часть моего метода продажи включает в себя поиск союзников. В главе 1 говорилось о проблемах выстраивания крепких взаимоотношений с внутренними заказчиками: программистами, сотрудниками отделов продаж и маркетинга, специалистами по поддержке пользователей, сетевыми администраторами. Сейчас подходящий момент для использования этих отношений. Определите конкретные способы, которыми бюджет поддерживает услуги и продукты для нужд заказчиков. Определите, какие из этих продуктов и услуг вы не сможете поставить в случае урезания бюджета. Попросите ваших коллег о поддержке. Не забывайте также о человеческих аспектах при получении поддержки. Поскольку тестировщики исполняют роль поставщиков услуг, они должны

обладать точкой зрения, ориентированной на предоставление услуг. Я прилагаю усилия к тому, чтобы выстроить партнерские позитивные взаимоотношения с людьми из проектной группы, дабы быть уверенным в том, что группа тестирования и ее работа рассматриваются как поддерживающий и положительный фактор для успеха проекта.

Также я использую результаты анализа инвестиций. В то же время я не ограничиваюсь только демонстрацией цифр. Если я приведу цифры: "Мы намерены получить 375% окупаемости вложений на тестирование", – это прозвучит не очень убедительно. Это лишь краткое заключение, требующее подробных пояснений, как получена цифра и что она означает[1].

Для пояснения окупаемости вложений я выхожу за рамки сухих тактических категорий – обнаружение дефектов, поддержка устранения дефектов, снижение рисков и управление проектом, чтобы увязать этот вопрос с тем, почему менеджеры вкладывают деньги в тестирование. Например, насколько предлагаемый подпроект тестирования позволяет компании снизить общие затраты на систему, связанные с проблемами качества? Какую степень доверия к надежности эксплуатации системы мы можем гарантировать руководству? Не бойтесь также говорить о нематериальных активах компании. Как раз потому, что вы не можете количественно оценить их, спрашивайте менеджеров, насколько будет выгодно улучшить репутацию компании за счет повышения качества системы. Насколько выгодно снижение хаоса, повышение морального духа и снижение текучести персонала?

Разумные менеджеры проектов часто отвечают на эти замечания: "Да, мне нужны преимущества тестирования, но почему они так дороги и требуют столько времени?" Это как раз тот момент, когда нужно уметь четко разъяснить план проекта тестирования, тему предыдущих двух глав. Наша способность объяснить логику и взаимосвязи между тестированием и остальной частью проекта позволяет менеджерам увидеть проблемы, с которыми мы сталкиваемся при тестировании. Кроме того, мне нужны данные за продолжительный период времени, лучше о похожих проектах, чтобы показать, почему требуется примерно столько времени, насколько я оцениваю усилия группы проекта по поиску и устранению важных ошибок.

В некоторых случаях присутствует излишний оптимизм в отношении качества системы, проецируемый на тестирование и общий ход проекта. В этих случаях я считаю, что сокращение *утвержденного* графика тестирования и урезание *утвержденного* бюджета тестирования может в краткосрочной перспективе *показать* уменьшение времени и затрат на тестирование. Однако сокращение адекватной проверки качества на самом деле увеличивает риск превышения бюджета и нарушения сроков проекта, часто очень значительно по отношению к реалистичным бюджету и срокам, которые были выработаны в начале проекта. Дефекты, обнаруженные на ранних стадиях проекта за счет эффективного и целесообразного

---

[1] В забавной книге Дугласа Адамса (Douglas Adams) "*The Hitchhiker's Guide to the Galaxy*" компьютер вычисляет значение жизни, равное 42. Этот результат обманывает и разочаровывает людей, которые запрограммировали компьютер, чтобы узнать, что такое жизнь. Меняющееся значение окупаемости инвестиций может в равной степени ввести в заблуждение ваших менеджеров.

тестирования (включая статическое тестирование требований, архитектуры и просмотр кода), могут быть легко и дешево устранены с незначительным влиянием на сроки и бюджет проекта. Однако серьезные ошибки, обнаруженные в последние две недели, могут иметь фатальные последствия.

Применяя свой метод "продажи" бюджета, я предпочитаю выступать в роли адвоката, а не понтифика. Если я общаюсь с менеджерами, непонимающими тестирования и скептически относящимися к бюджету и графику, мне нужно мобилизовать силы и выдвинуть в качестве предложения положительный и убедительный вариант. Я могу быть эмоциональным, но избегаю поучений. Продавая оценку затрат, я считаю, что несу свет знаний в массы коллег, старших и высших руководителей. Если я найду правильные мотивы для них или, другими словами, сделаю желаемым то, что я продаю, они захотят понять, что я предлагаю[1].

Джамал готовит свой вариант

В пятницу во второй половине дня, завершив оценку затрат, Джамал позвонил своему боссу, вице-президенту поддержки разработок Джону Албертсону. "Привет, Джон, — сказал он. — Я почти готов к обсуждению с тобой предложений по графику и бюджету тестирования Суматры в понедельник. Есть ли у тебя время с утра?")

"Конечно, — ответил Джон, — я записываю тебя на 10.00".

Джамал отправился на поиски менеджера продукта SpeedyWriter Кейт Эрнандес с копиями графика и бюджета. "Привет, Кейт, есть минутка?")

"Да, заходи, что у тебя?")

"Вот, — сказал Джамал, — у меня примерный план-график и оценка бюджета тестирования, которые я собираюсь предложить для проекта. Думаю, что должен тебе их показать". Он быстро разложил перед Кейт различные активности, затраты и анализ окупаемости инвестиций. "На основе этого бюджета и графика, я думаю, мы сможем вести действительно тщательное тестирование и предоставлять тебе очень точные данные о состоянии проекта и о том, как он идет, на еженедельных совещаниях".

Кейт почесала затылок. "Это больше, чем я думала. Я смотрела предыдущие проекты и решила, что полмиллиона на тестирование будет достаточно".

"Да, но те подпроекты тестирования продолжались только 4–5 месяцев. Наши среднемесячные затраты лишь немногим больше, чем в предыдущих проектах. Я нанимаю в основном лаборантов-контрактников для экономии расходов".

После непродолжительного обсуждения Кейт согласилась посмотреть, сможет ли он частично урезать бюджет группы маркетинга и бюд-

---

[1] Несколько других идей о том, как продавать тестирование руководству, можно найти в статье Джеффа Пэйна (Jeff Payne) "Quality Meets CEO", которая была впервые напечатана в журнале "*Software Testing and Quality Engineering*", Volume 1, Issue 3 (May/June 1999), а сейчас доступна на www.stickyminds.com.

жет разработки Дженни Кауфман для частичного возмещения избытка в $300 000. Оставшуюся сумму она пообещала поискать вместе с генеральным директором Раджешем и вице-президентом по маркетингу Максом.

Возвращаясь в свой кабинет, Джамал сделал остановку у кабинета Дженни, чтобы показать график и бюджет на верхнем уровне детализации и предупредить ее о возможном урезании бюджета группы разработки. Дженни не сильно обрадовалась уменьшению бюджета, особенно когда поняла, что это может выразиться в потере одного сотрудника. Джамал предложил заключить сделку, выразив желание принять участие в разработке тестов для компонентного тестирования. "Слушай, нам тоже нужны эти тесты, Дженни, поэтому не будет проблем, если мы разработаем вместе с твоими ребятами некоторые из них".

"Знаешь, — сказала Дженни, — это, возможно, даст лучшие результаты. Мы сможем воспользоваться поддержкой специалистов при создании и выполнении тестов".

Поучаствовав еще немного в светской беседе, Джамал пошел к себе, спросив, уже уходя, Дженни, не может ли она сообщить Кейт уже сегодня, что в целом поддерживает оценки графика и бюджета. Дженни согласилась подготовить и отправить короткое сообщение по итогам обсуждения с Джамалом, в частности, упомянув, что она не возражает против небольшого уточнения бюджета ее группы с учетом того, что группа тестирования окажет ей дополнительную помощь при подготовке и проведении компонентного тестирования.

В понедельник утром Джамал отправился на встречу с Джоном. В дополнение к основательным оценкам затрат Джамал решил опираться на свою репутацию склонного к недооценкам, надежного и заслуживающего доверия менеджера. Когда он пришел, вице-президент по системной разработке Гарольд как раз завершал разговор с Джоном. "Доброе утро, Джамал! — приветствовал его Гарольд. — Рад тебя видеть. Дженни сообщила, что вы вдвоем обсудили примерные график и бюджет тестирования, а Кейт сказала, что у вас есть неплохой совместный план действий. Ты не против, если я поучаствую в обсуждении? Мне хотелось бы побольше узнать о твоих оценках".

Ни Джамал, ни Джон не возражали. Втроем они разложили таблицу анализа видов ошибок и их влияния, результаты анализа тестового покрытия, график Гантта, таблицы с бюджетом и с описанием окупаемости вложений в тестирование на длинном столе в кабинете Джона и принялись за работу. Джамал начал с результатов анализа рисков качества и тестового покрытия, сказав: "Давайте сначала посмотрим на то, как я получил эти оценки и какие риски мы собираемся снижать". Он показал различные риски, включая комментарии, почему они имеют значение, и тестовые сценарии, которые их покрывают. В результате он перешел к декомпозиции работ, показав сопутствующие задачи. Затем он представил бюджет и таблицу окупаемости инвестиций в тестирование. В ходе всего обсуждения Джон и Гарольд задавали различные уточняющие вопросы, но ничего не комментировали. Наконец Джамал сказал: "Итак, что вы думаете?")

№ этапа	Этап	Выполнено?
3.A	Презентация преимуществ подпроекта тестирования.	☑
3.B	Обрисовать временные и финансовые ресурсы, необходимые для получения описанных преимуществ.	☑

Кивнув, Джон сказал: "Я думаю, что ты проделал хорошую работу, получив отлично документированную оценку затрат. Я также думаю, что ты организовал хорошую презентацию оценки. Но...")

"Я-то надеялся, что не будет никаких "но")", – отшутился Джамал.

"Боюсь, что без них не обойтись. Гарольд, Кейт и я рассмотрели оценку со всех сторон, и мне кажется, что мы не можем принять цифру 800К. Я думаю, что 700К более реально".

Джамал отвечал: "Могу урезать непредвиденные расходы, чтобы дать эту цифру, но не думаю, что это реально. Мы все равно в конце получим что-то около 20%, я просто не знаю, где взять еще".

Джон кивнул, а Гарольд вставил: "Знаешь, мне кажется, что непредвиденные расходы предназначены для покрытия непредвиденных сдвигов графика, не так ли? В нашем случае ты закладываешься на два дополнительных месяца финансирования персонала. С учетом того, что мы используем итерационный подход, я думаю, ты мог бы сократить половину. Я уверен, что мы не отстанем больше чем на месяц".

"Звучит довольно привлекательно, – сказал Джамал, – при условии, что мы все соглашаемся, что, если график сдвинется больше чем на месяц, нам придется выйти за пределы бюджета". Джон и Гарольд согласились. "Теперь нам нужно сэкономить еще $60 000, чтобы не выйти за $700 000. Нам придется урезать часть тестирования".

"Что если не нанимать дополнительного тестировщика? Это даст почти то, что нужно".

"В этом случае мне придется сдвинуть начало тестирования примерно на месяц, и я думаю, что тогда мы не сможем выполнить договоренность с Дженни о помощи в подготовке и проведении компонентного тестирования".

"Хм, – заметил Гарольд, – это не очень хорошо. Мы очень на это рассчитываем".

"А что если сделать вот так? – предложил Джамал. – Вместо параллельного выполнения тестов на надежность и перегрузку на двух отдельных конфигурациях мы купим только один дополнительный сервер и будем чередовать тестирование надежности и перегрузки через раунд? Это, конечно, создаст небольшой разрыв в регрессионном тестировании, поскольку мы будем получать меньше информации о том, как каждая сборка влияет на надежность и производительность системы, но это позволит сэкономить $75 000, предназначенных на дополнительную тестовую среду".

Гарольду и Джону очень понравилась эта идея. Джамалу тоже. Это на самом деле не только снизит затраты, но и позволит высвободить часть времени Эммы, поскольку ей не нужно будет каждый раз организовывать выполнение обоих комплектов автоматизированных тестов одновременно. Джон и Гарольд предложили Джамалу перейти к следующим этапам и

продолжить работу с данной оценкой. Джамал положил документы с планом-графиком и бюджетом в репозиторий проекта и мог теперь переходить к разработке планов тестирования.

№ этапа	Этап	Выполнено?
3.C	Осознание и принятие мер по разрешению любых возражений по поводу сделанных оценок посредством повторения шагов 3.A и 3.B.	☑
3.D	Если не удается получить согласие руководства на предлагаемые план-график и бюджет, обсудить, какие области тестирования могут быть исключены, установить, какие ограничения по срокам и бюджету позволят получить поддержку руководства.	☑
3.	Утвердить у руководства план-график и бюджет.	☑
4.	Повторить этапы 1–3, если необходимо. Уточнить график работ и бюджет с целью добиться адекватности ресурсов рамкам тестирования (возможно уточненным) и одобрения руководства.	☑
5.	Поместить принятые документы, фиксирующие план-график и бюджет, в проектную библиотеку или систему управления конфигурацией проекта. Организовать контроль изменений документов.	☑

Построение успешного процесса оценки затрат

В последних трех главах был представлен большой объем информации об оценке затрат. Возможно, читатель думает, что вся сложность процесса оценки затрат состоит в трудности построения грамотного процесса. Однако, немного попрактиковавшись, вы обнаружите, что это не так сложно. Мы можем правильно построить процесс оценки затрат, поскольку он позволяет рабочей группе проекта делать следующее.

Определение факторов, влияющих на оценку

Системная разработка, сопровождение, поставка, включая тестирование, — это сложные мероприятия с высоким риском. Важно комбинировать полезные методы оценки затрат с пониманием факторов, которые влияют на затраты, сроки, зависимости и ресурсы. Некоторые из этих факторов могут и увеличить, и сократить сроки, тогда как другие все только замедляют.

Некоторые из этих факторов вытекают из процесса выполнения работы:

- Степень, в которой активности тестирования пронизывают проект или присоединяются в конце
- Четко определенные точки передачи управления между тестированием и остальной частью проекта или компании
- Число изменений в проекте, особенно при использовании автоматизированных тестов или детально подготовленных тестовых скриптов для ручного тестирования

- Хорошо управляемые процессы управления изменениями для графиков разработки и тестирования, требований к продукту, архитектуры, разработки и тестирования
- Выбранный жизненный цикл системы, включая эффективность, производительность и степень организации процессов тестирования и разработки в рамках жизненного цикла
- Реалистичные и работающие графики и бюджеты разработки и тестирования
- Своевременное поступление высококачественных передаваемых результатов тестирования
- Своевременное и надежное устранение обнаруженных дефектов, особенно при нарушении критериев начала тестирования
- Надлежащее проведение ранних фаз тестирования (модульного, компонентного, комплексного)

Некоторые из факторов имеют материальную основу и обусловлены природой проектной работы, имеющихся инструментов, доступных ресурсов и т.п.:

- Наличествующие, освоенные высококачественные инструменты и средства автоматизации тестирования и процесса
- Качество системы тестов, под которой подразумеваются тестовая среда, процесс тестирования, тестовые сценарии, инструменты тестирования и т.п.
- Системы тестов и документация многократного применения из предыдущих похожих проектов
- Адекватная, отдельная и безопасная тестовая среда
- Отдельная, адекватная среда отладки у группы разработки
- Наличие надежного тестового оракула (как мы можем узнать ошибку, когда видим ее)
- Доступная проектная документация высокого качества (понятная, лаконичная, точная и т.п.), в том числе требования, архитектура, планы и т.п.
- Похожие проекты и тестирование, выполнявшиеся ранее

Некоторые факторы вытекают из особенностей состава группы тестирования, и это, возможно, наиболее важные из всех:

- Надлежащие навыки, опыт и отношение к работе в проектной группе, особенно у менеджеров и ключевых сотрудников, а также в обучении и поиске новых людей
- Организационные изменения, включая массовые реорганизации в середине проекта
- Чувство обиды и другие проблемы взаимоотношений в проектной группе
- Сверхурочная работа, работа по выходным и рабочие смены вне пределов основного рабочего времени

- Усталость и моральные проблемы, особенно накопившиеся в предыдущих проектах
- Корпоративная культура и привычки
- Компетентность и опыт группы управления проектом
- Вдохновляемые и вдохновляющие менеджеры и технические лидеры, особенно подготовленная группа управления, являющаяся поборником определенных уровней качества и тестирования
- Понимание всеми участниками проекта необходимости тестирования, подготовки версий, системного администрирования и других неярких, но имеющих большое значение работ (т.е. неприятие культуры "отдельных героев"))
- Реалистичные ожидания со стороны всех участников проекта, включая непосредственных исполнителей, менеджеров и заинтересованных лиц
- Стабильность проектной группы, отсутствие текучести персонала
- Выстроенные позитивные взаимоотношения в рамках проектной группы, включая непосредственных исполнителей, менеджеров и заинтересованных лиц
- Компетентная и своевременная поддержка тестовой среды
- Использование квалифицированных контрактников и консультантов для заполнения вакансий
- Честность, обязательность, прозрачность, открытые планы, доступные непосредственным исполнителям, менеджерам и заинтересованным лицам

Наконец, несколько усложняющих факторов, которые всегда увеличивают сроки и затраты:

- Большая сложность процесса, проекта, технологии, структуры компании или тестовой среды
- Большое число лиц, заинтересованных в тестировании, качестве системы или самом проекте
- Внешние зависимости, особенно от незаинтересованных лиц
- Вопросы местоположения: географическое расположение проектных групп или помещений офисов, требующее частых поездок или увеличения времени на обеденный перерыв
- Большое количество подгрупп, особенно если они разделены географически
- Необходимость поднимать, обучать и вводить в проект растущую группу тестирования или всего проекта
- Необходимость осваивать или разрабатывать новые инструменты, методы и технологии как в тестировании, так и в проекте в целом
- Наличие заказного аппаратного обеспечения
- Любое требование новой системы тестов, особенно для автоматизированного тестирования, как часть работ по тестированию

- Любое требование разработки непротиворечивых тестовых сценариев с высокой степенью детализации, особенно в незнакомом стандарте документирования
- Сложное расписание поступления компонентов, особенно для комплексного тестирования и разработки тестов
- Хрупкие тестовые данные, например чувствительные ко времени

Специфические аспекты проекта в каждой из категорий влияют на ресурсы и сроки, необходимые для выполнения определенных действий. При подготовке оценки затрат на тестирование тест-менеджер и те сотрудники группы тестирования, которые помогают ему провести оценку, должны учитывать, насколько каждый из этих факторов влияет на оценку.

Пренебрежение или пропуск одного из факторов может превратить реалистичную оценку в неверную и бесполезную. Опыт – это часто единственный учитель, но толковые тест-менеджеры могут научиться задавать умные вопросы и самим себе, и проектной группе о том, насколько каждый из факторов влияет на ход проекта.

Представление реалистичной картины будущего проекта

Главное достоинство любого хорошего прогноза в том, что он сбывается. Предсказали ли мы будущее правильно с определенной степенью точности? Конечно, будет какое-то отклонение, ошибка на какой-то процент времени и денег, но, если мы грамотно провели оценку, половина оценок будет завышенной, а половина заниженной. Также ошибка не должна превышать 10–20%, и ни в коем случае оценка не должна отличаться от фактического результата в несколько раз.

К сожалению, многие официальные графики и бюджеты проектов превышаются, часто на значительные величины. Любой наудачу взятый экземпляр делового журнала или газеты, наверняка, содержит какую-нибудь "посмертную" историю о проекте в области информационных технологий или системной разработки, который вышел за рамки бюджета и сроков. По моему мнению, это происходит по нескольким причинам.

- *Новые технологии.* В некоторых случаях затраты в проектах, в которых применяются новые технологии, недооцениваются. Никто реально не знает, сколько времени нужно на выполнение работы, поскольку никто никогда ее не делал.
- *Производительность рабочей группы.* Некоторые проектные группы не достигают необходимого уровня эффективности или производительности из-за текучести кадров, усталости основных исполнителей, отсутствия обучения и т.п. (См. некоторые идеи о работе с этими проблемами в проекте в главах 8 и 9).
- *Отсутствие знаний.* Некоторые группы управления проектом не способны применять подходящие методы оценки затрат.
- *Отсутствие согласия.* В некоторых проектах серьезные и непрерывные споры между ключевыми заинтересованными лицами мешают достижению согласия относительно того, что же все-таки создается. В результате мы не можем оценить затраты на тестирование в

постоянно меняющемся проекте, не может быть успешным выполнение всех работ проекта.

- *Чрезмерные внешние ограничения.* Некоторые группы управления проектом создают графики и бюджеты в рамках цифр, диктуемых маркетологами, пользователями и другими людьми, не участвующими в процессе разработки системы, ее сопровождения или поставки, что является дополнительной проблемой, если множество функций системы и уровень ее качества также ограничиваются извне.
- *Растянутые цели.* Некоторые группы управления проектом объявляют цели, которые, как им заранее известно, не будут достигнуты, для того чтобы увеличить давление на непосредственных исполнителей и заставить их работать напряженнее.
- *Финансирование.* В некоторых случаях спонсирующие организации или кредиторы компании (например, вкладывающие свой капитал с риском) отказываются финансировать проект, если он стоит меньше, чем определенная сумма, или не выполняется по определенному графику, что сказывается на проектах с такими же объемами бюджетов и рубежами графика.

За исключением первых двух факторов, это проблемы, с которыми мы можем и, я убежден, должны работать при подготовке оценки затрат. Я ставлю цель подготовить реалистичную, работающую и правдивую оценку затрат, а затем обсудить соответствующие изменения в рамках проекта и в тестовом покрытии как необходимые компоненты встраивания графика тестирования в общий график проекта.

Реалистичность в подходе к оценке означает полноту. Другими словами, Должны быть определены все задачи, необходимые для успеха проекта. Я работал в большом проекте разработки, который был распределен по шести группам в трех городах на двух континентах. В начальный график проектная группа забыла включить фазы, действия и даже задачи интеграции компонентов и комплексного тестирования. Никаких ресурсов не было выделено на объединение составляющих систему компонентов, получаемых от различных групп, и на проверку этого объединения. Это нереалистично.

Чтобы оценка была реалистичной, она должна учитывать каждую фазу, активность и задачу, необходимые для успеха. Строгое следование правилам декомпозиции работ, которые обсуждались ранее, – это один из способов обеспечить полноту включения всей проектной группы, особенно опытных исполнителей, в процесс оценки затрат.

Подготовка реалистичного графика также означает получение и использование разумных оценок затрат и ресурсов на каждую задачу. В другом большом проекте одним из ключевых моментов был переход от хранилища плоских файлов к реляционной базе данных. На разработку инструментов и процесса для конвертирования всех данных из старой архитектуры в новую было выделено всего две недели. Это тоже нереально.

Для получения реалистичной оценки необходимо, чтобы она была основана на разумной продолжительности задач. Здесь снова помогают следование проверенным практикам составления декомпозиции работ, метод экспертных оценок и другие методики, основанные на экспертизе внутри группы.

Назначение ответственных за выполнение

Работающая оценка — это такая оценка, когда ответственные за все составляющие задачи определены, согласны их выполнять и утверждены менеджерами. Однажды я видел график, в котором декомпозиция работ была сделана только до уровня активностей, причем за каждую активность отвечало до шести человек, а продолжительность составляла две-три недели. Как разобраться каждому из участников, что же ожидается от них? Чтобы дополнительно все запутать, группу тестирования назвали группой контроля качества. Они полагали, и им об этом говорили, что обязанность группы — внедрение изменений в процесс. Однако все остальные участники проекта думали, что группа будет тестировать продукт.

Я говорил куратору проекта из числа топ-менеджеров, что не понимаю, как он будет определять, в каком месте графика он находится, те ли задачи выполняют участники проекта. С работающей оценкой все понимают, какие задачи попадают в их сферу ответственности и как можно оценить степень завершенности каждой из задач в терминах передаваемых результатов.

Работающие оценки также учитывают ограничения привлекаемых ресурсов. Например, классической ошибкой является планирование использования одного серверного кластера как для функционального, так и для нагрузочного тестирования в одно и то же время. Другая проблема возникает, когда планируют использование тестовых сред, расположенных в других офисах, не имея практической возможности контролировать конфигурирование и работу этих сред. Существуют зависимости между задачами тестирования, связанные с используемыми ими ресурсами.

Работающая оценка должна выявлять не только зависимости между ресурсами, но и все другие зависимости, взаимосвязанные цепочки задач и критические пути. Необходимо понимать, что оценки затрат не достигнут совершенства. Следовательно, необходимо пытаться предсказывать последствия позднего завершения задачи или ее выхода из бюджета. Это особенно важно для тест-менеджера, который должен понимать, когда, кем и в результате какого процесса каждый из передаваемых результатов будет предоставлен его группе.

Зависимости, взаимосвязанные цепочки и критические пути также являются элементами правдивых графиков. Сочетание рисков вдоль критического пути — удивительный, но важный факт для осознания тес-

Критический путь, критическая сеть (Critical Path, Critical Network)

Совокупность задач (в терминах длительности активностей и задач) в структурной декомпозиции проектных работ (с учетом всех межзадачных взаимозависимостей), которые должны завершиться к определенному моменту для того, чтобы завершить проект в срок.

тировщиками и менеджерами. Посмотрим на сочетание рисков с математической точки зрения.

Пусть имеется 12 задач на критическом пути в проекте. Вероятность завершения в срок любой из задач равна 90%. Вероятность завершения раньше срока любой из задач равна 0%. Заметим, что своевременное завершение каждой задачи зависит от своевременного завершения предыдущих задач. Статистическое представление, носящее название теоремы Байеса, имеющей дело с условной вероятностью, говорит о том, что вероятность завершения в срок последней задачи в перечне меньше 30%:

$$0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 \times 0.9 = 0.9^{12} \quad 0.28$$

Если вероятность завершения каждой из задач снизить до 75%, суммарные шансы своевременного завершения всех задач падают до 3%. А что если, как часто это бывает, последняя задача на критическом пути — это рубеж, означающий завершение проекта? Сочетание рисков на критическом пути дает нам лишенное эмоций логическое, чисто математическое объяснение, почему использование графиков как растянутых целей или как средства принуждения людей работать напряженнее, является столь непродуктивным и часто безуспешным.

Планы на случай непредвиденных обстоятельств, резерв времени, контрольные точки и критерии начала фаз могут способствовать разрешению проблем графика, связанных взаимозависимостями задач. Грамотные оценки учитывают проблемы в выполнении задач в ограниченные периоды времени. Они позволяют реагировать и регулярно пересматривать график. Они позволяют увидеть нарушение сроков и затрат, связанных с задачами критического пути или близкими к нему, и управлять ими до того, как в конце проекта разразится кризис графика.

Честные прогнозы

Правдивые оценки предполагают полное доверие группе. Когда менеджеры запрашивают оценки затрат у сотрудников своих групп, они должны воспринимать полученные оценки как реалистичные, а не как коварные попытки лентяев получить удобный график. Вспоминаю совещание, на котором менеджеры проектов признали, что действующий график является фикцией, но согласились оставить его без изменений, чтобы установить растянутые цели для группы разработки.

Такой стиль руководства производит на меня впечатление нечестного и основанного на предположении, что люди ленивы от природы. На самом деле большинство оценок затрат, которые я получаю от своих сотрудников, обычно слишком оптимистичны. Мне приходится обсуждать эти оценки с людьми и обычно изменять их в сторону увеличения. Иногда исполнители не видят других "изменяющихся частей", которые оказывают влияние на их производительность.

Мы должны быть честными и правдивыми также в наших собственных суждениях, особенно имея дело с тем, что мы пока не знаем. В "*Апологии Сократа*" Платон цитирует философа, который предполагает, что знание собственного уровня незнания есть сущность человеческой мудрости.

Я встречал оценки, которые в два раза отличались от реальных затрат. Но я не помню случая, чтобы какие-либо из них были недостаточными.

Существует предел знаний о ходе проекта в первые несколько недель. Снова отмечу, что резерв времени, планы на случай непредвиденных обстоятельств, контрольные точки и критерии начала и завершения фаз позволяют снизить риск неточных оценок и своевременно отреагировать на расхождения действительности и оценок. Когда мы видим, что это происходит, честность требует смело говорить об этом вслух. При честной оценке затрат все менеджеры и непосредственные исполнители не боятся указывать группе управления проектом на расхождения в случае их возникновения.

Наконец, подготовка реалистичных, работающих и правдивых оценок затрат также требует, чтобы мы согласились с контекстом и ограничениями проекта. Если я представлю руководству график и бюджет подпроекта тестирования, который сдвигает дату выпуска на 50% и превышает бюджет на 200%, с моей стороны нереально ожидать утверждения менеджментом таких оценок. Да, это верно, что тестирование обычно увеличивает продолжительность проектов. Часто действительно необходимо заложить больше тестирования, но это не означает непреклонного отказа работать в разумных рамках.

Я извел море чернил на эти три главы, посвященные оценке затрат, поэтому вы, вероятно, поняли, что я очень беспокоюсь об этом. Почему оценка затрат так важна? Я участвовал в нескольких проектах, в которых оценка была выполнена хорошо, и работать в этих проектах было истинным наслаждением. Проектные группы сверху донизу ощущали гордость, когда своевременно достигались рубежи и проект успешно завершался. Ключевые заинтересованные лица, не входящие в проектную группу, были рады получить системы в срок и в рамках бюджета. Я также работал в проектах, в которых, надо честно признать, напряженно работающих программистов, тестировщиков и других непосредственных исполнителей уговаривали, ругали и обвиняли в результатах неудачных оценок затрат, сделанных руководством. В двух таких проектах неудачные оценки проявлялись как раз в конце проекта, в ходе тестирования, заставляя меня бороться за то, чтобы сделать что-то полезное, в то время как другие менеджеры тайно ворчали, рассуждая, сколько еще продлится тестирование. Неверная оценка затрат обрекает проект и проектную группу на неудачу.

Таким образом, я думаю, что мы, специалисты по разработке программного обеспечения, должны начать настраивать себя на успех. Успех проекта опирается на баланс свойств, сроков, бюджета и качества. Реалистичные, работающие и правдивые оценки затрат непосредственно в начале проекте — важная часть построения этого баланса. Если имеются честные оценки графика проекта и мы честны с нашими руководителями, коллегами, группами и самими собой при обсуждении оценок, это настраивает нас на успех. Если нам известно, кто, что и к какому сроку должен сделать и какие существуют зависимости и риски, это также настраивает нас на успех. Подготовка оценки с учетом всех взаимозависимостей — это не скучная рутинная бумажная работа, с трудом выполненная во второй половине дня в Microsoft Project и затем забытая. Оценка затрат — это важнейшая функ-

ция управления, направленная на получение одобрения заинтересованных лиц, определение реалистичных ожиданий, честное обсуждение с сотрудниками возможных достижений и управление успехом проекта.

Быстрая оценка затрат

Оценка затрат полезна, поскольку является прогнозом. Оценка затрат позволяет нам предсказать будущее и соответствующим образом строить планы. Оценка проекта тестирования — график, бюджет, окупаемость инвестиций — это прогнозы на будущее в терминах сроков и финансовых затрат.

Как и все оценки, ваша оценка будет неточной и до некоторой степени неверной. Это нормально. Вы можете прекрасно предсказать погоду на завтра, если подождать до послезавтра. Но ценность предсказания в том, что вы заглядываете в будущее и получаете возможность предпринять своевременные действия, чтобы повлиять на это будущее.

Как при управлении морским судном, чем раньше вы начинаете действовать, тем менее усердными и исключительными должны быть ваши усилия. Если вы начинаете слишком поздно, никакие корректирующие действия вам не помогут. (Продолжая морскую тему, можно вспомнить, что попытка вырулить "Титаник" от судьбоносного айсберга началась слишком поздно.) Таким образом, значение любой оценки состоит в том, чтобы она была в наличии, когда требуется принимать меры, а также в том, что реализация плана, вытекающего из оценки затрат, добавляет стоимость описанными выше способами.

Решение проблем

В рассматриваемом нами конкретном случае приведен пример того, что происходит, если процесс оценки был успешным. К сожалению, этот процесс не всегда идет гладко. Предусмотрительные тестировщики готовятся к проблемам, которые порождают оценки затрат.

Тестирование как объект инвестиции

При обсуждении окупаемости инвестиций в тестирование я упростил картину за счет вычисления затрат и доходов, как будто они имеют место примерно в один и тот же период времени. Очевидно, что затраты на тестирование распределяются по всему графику проекта. Это также верно для информации, которая направляет проект, поскольку значение этой информации осознается в ходе проекта.

Тем не менее стоимость, добавленная за счет обнаружения дефектов (независимо от их устранения) и за счет снижения рисков, относящихся к ошибкам, которые могли быть внесены, накапливается в течение некоторого периода времени после завершения проекта. Это приводит к двум оговоркам, касающимся подхода к окупаемости вложений, которые нужно иметь в виду, если метод применяется с целью обоснования предлагаемого бюджета.

Во-первых, многие проекты используют учетную политику и стимулирование, работающие против окупаемости вложений в тестирование. В большинстве случаев проекты имеют установочный бюджет, включающий в себя только затраты, связанные с выпуском текущей версии. Затраты, понесенные за счет неверных решений, принятых в ходе проекта, и проявившиеся при сопровождении, не берутся в расчет бюджета проекта так же, как и полученные доходы, которые проявляются позднее. Таким образом, группа управления проектом, как это ни печально, имеет стимулы находить и исправлять только те ошибки, которые никак нельзя оставить в продукте для его выпуска.

Вновь обратимся к гипотетической модели стоимости качества (см. главу 4), воспроизведенную для удобства на рис. 5.1. Предположим на время, что мы живем внутри комикса про Дилберта, и допустим, что проектная группа должна исправить только половину из 1000 ошибок, имеющихся в системе, чтобы продукт нормально работал перед поставкой. Это даст на основе модели 10%-ный средний рейтинг удовлетворенности пользователя.

(Дилберт – известный в США рисованный персонаж, созданный художником Скоттом Адамсом. Дилберт – самый известный офисный работник в США, постоянно попадающий в различные забавные ситуации вместе со своим лысым боссом. Пример комикса с русским переводом можно найти на http://obzor.com.ua/multfilm/misc/2001-05-24-7.shtml. – *Прим. пер.*)

Предположим, что менеджер проекта достаточно циничен, чтобы считать, что все ошибки, обнаруженные после выпуска версии являются проблемами менеджера службы технической поддержки. Допустим, что ежемесячно обнаруживается половина из оставшихся ошибок, и, таким образом, доля обнаруженных дефектов уменьшается до ничтожно малой величины примерно через шесть месяцев после выпуска версии. Сейчас модель показывает затраты на несоответствие примерно $550 000 при удовлетворенности пользователя в 10%. На рис. 5.2 показаны затраты на несоответствие, сходящиеся к этому числу в первые шесть месяцев. Более того, заметим, что до выпуска версии окупаемость инвестиций в тестирование на самом деле отрицательная, поскольку затраты на соответствие превосходят затраты на несоответствие.

Хорошо, покинем комикс Дилберта – он слишком мрачный. Предположим, что есть менеджер проекта, заботящийся о пользователях. Он нанимает независимого тестировщика для обнаружения большинства дефектов до поставки версии. Он также определяет цель для разработчиков: устранить две трети ошибок до выпуска версии, примерно на 150 ошибок больше, чем в предыдущем варианте. На рис. 5.3 показано, как работает этот подход в начальный период эксплуатации версии. В перспективе стоимость качества снизится примерно до $200 000, что составляет треть величины, полученной в предыдущем примере (рис. 5.2). Это существенный прогресс.

Однако в ближайшей перспективе, которую видят те, кто ведет учет затрат проекта, – это более дорогой проект. Для менеджеров проекта, которые получают премии от прибыли проекта, а не от долгосрочной прибыли, получаемой от разработки системы в течение всего ее жиз-

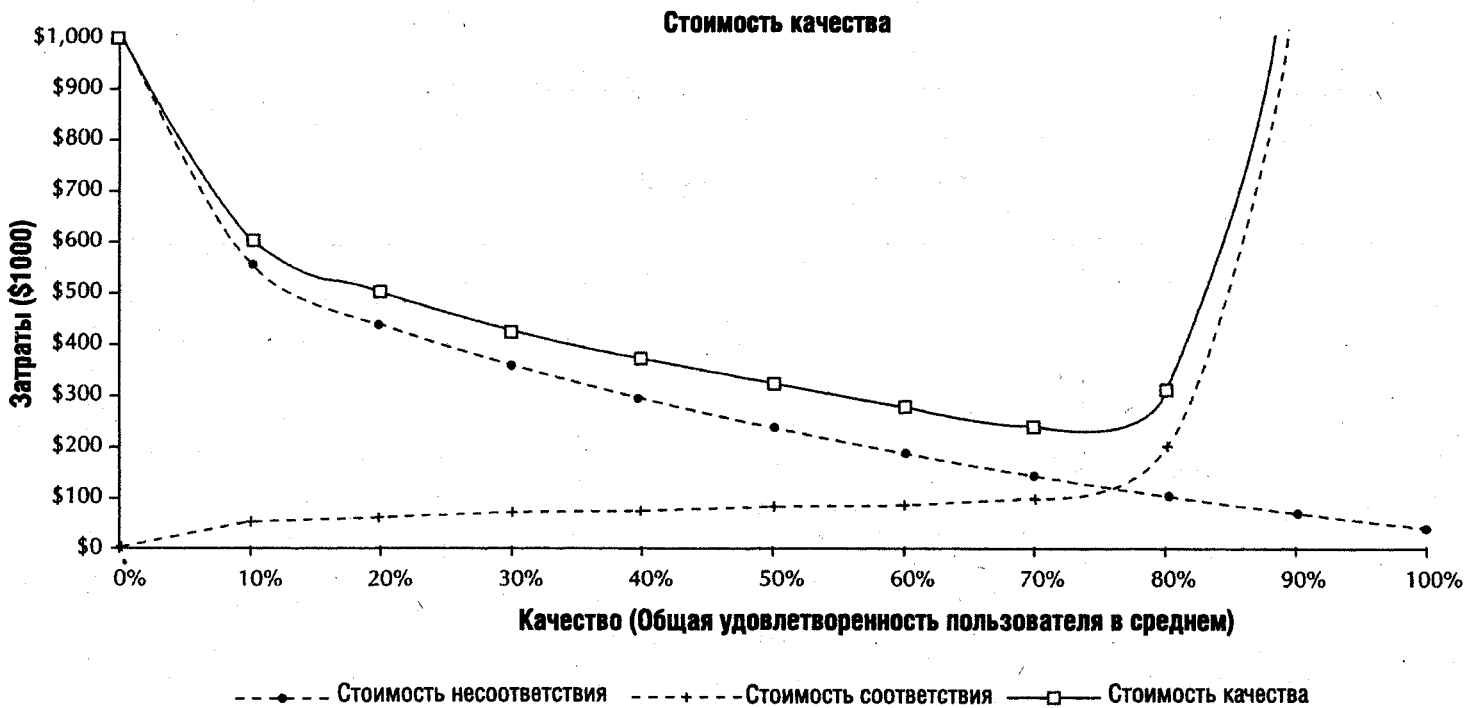


Рис. 5.1. Гипотетический график стоимости качества

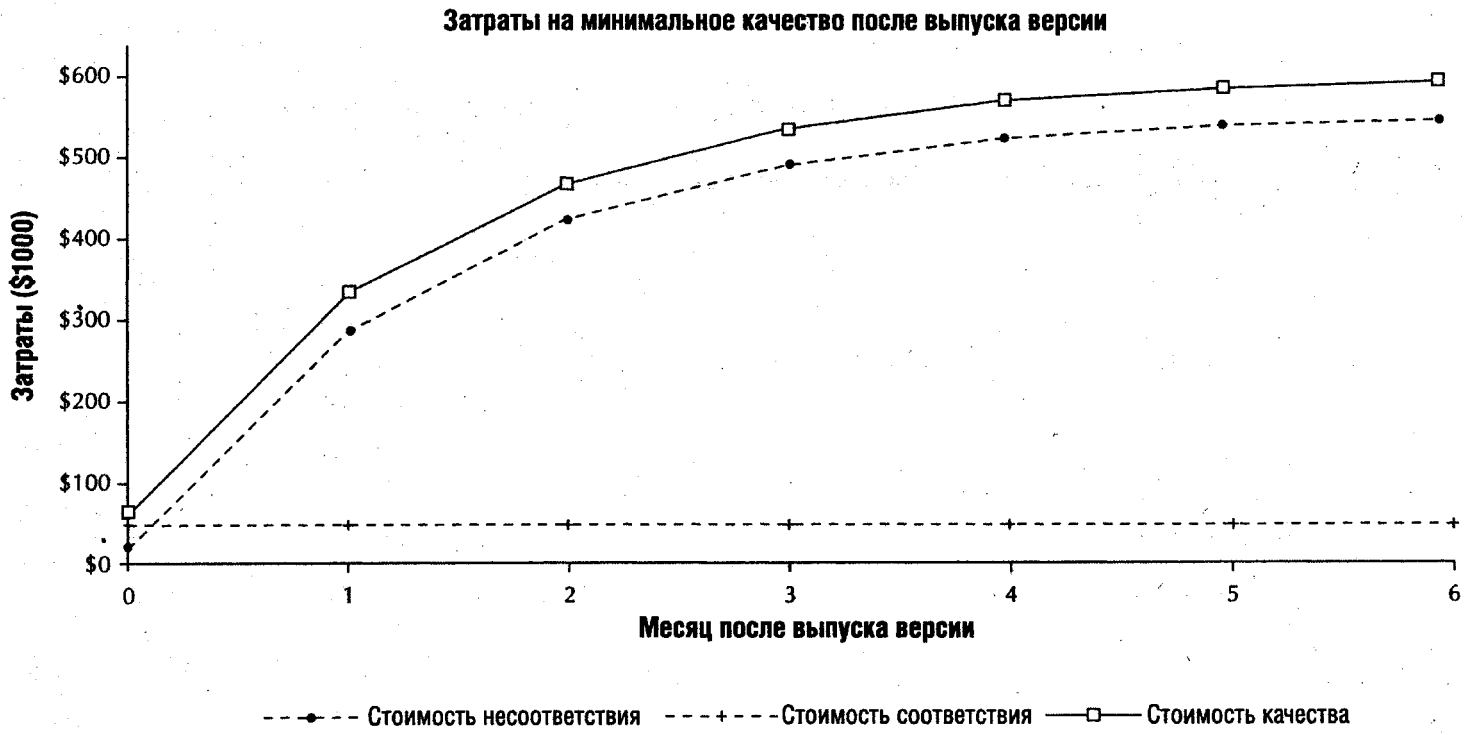


Рис. 5.2. Затраты на минимальное качество после выпуска версии

ненного цикла, затраты на тестирование всегда будут неизбежной помехой[1].

Во-вторых, поскольку некоторая экономия затрат на несоответствия проявляется со временем, продолжительность этого времени может уменьшить окупаемость вложений в тестирование. В бухгалтерском учете это понятие носит название "*чисто приведенная денежная стоимость*" (net present value of money). Чисто приведенная денежная стоимость говорит о том, что $1000 через шесть месяцев будут стоить меньше чем $1000 сегодня, поскольку мы можем вложить $1000 сегодня и получить больше чем $1000 через шесть месяцев. Насколько больше? Это зависит от процента окупаемости альтернативных вложений.

Вернемся к основному примеру книги, чтобы проиллюстрировать этот тезис. Для каждой исправленной ошибки, найденной группой Джамала, анализ окупаемости вложений, сделанный им, предполагает возврат около $1100. В то же время экономия в $1100 не имеет места одновременно с выпуском версии системы. Она происходит в некоторый момент в будущем, когда некоторое число пользователей встретят эту ошибку и потребуют ее исправить. Поскольку нам неизвестно точно ни для одной ранее исправленной ошибки, когда она могла бы проявиться, так как более некому объявить о ее отсутствии, момент, в который будет зафиксирована экономия, неизвестен и не может быть выявлен в принципе. Однако мы можем посмотреть на итоговые результаты.

В главе 4 было отмечено, что поддержка версии "Суматра" завершается по прошествии одного года после ее выпуска. Предположим, что менеджер поддержки Косимо Нграев говорит Джамалу, что в каждый месяц первого года он получает сообщения о половине ошибок, ранее не обнаруженных в версии. Если Джамал принимает альтернативный процент окупаемости вложений как 18% в год, или 1.5% в месяц, он может проанализировать вторую черновую версию бюджета и подсчитать возврат вложений на основе чисто приведенной стоимости, как это показано на рис. 5.4. (Заметим, что цифры, показанные на рисунке, учитывают два направления экономии затрат, согласованные Джамалом с Гарольдом и Джоном в основном примере книги.)

Сейчас возврат инвестиций в тестирование, равный 420%, не делает хуже бизнес-план Джамала. Но являются ли 18% в год разумной альтернативой вложений?

Более вероятно, что вся сумма или часть ее, которую запрашивает Джамал, может быть израсходована на наем новых программистов для реализации дополнительных свойств или ускорения сроков выполнения проекта. Каков процент окупаемости вложений при различных альтернативах, существующих в диапазоне от 0 до 40% средств (как предлагает Джамал), выделяемых на тестирование?

---

[1] Согласно гипотетической модели устранение примерно 900 ошибок минимизировало бы стоимость качества. Модель дает эту цифру на основе затрат на соответствие и затрат на несоответствие (при различных уровнях удовлетворенности пользователя) и соотношения между дефектами в поставленной версии и удовлетворенностью пользователя. В реальности лишь немногие менеджеры проектов имеют под рукой эту информацию, что делает выбор оптимального объема тестирования и исправления дефектов трудным даже для настоящих сторонников возврата стоимости качества.

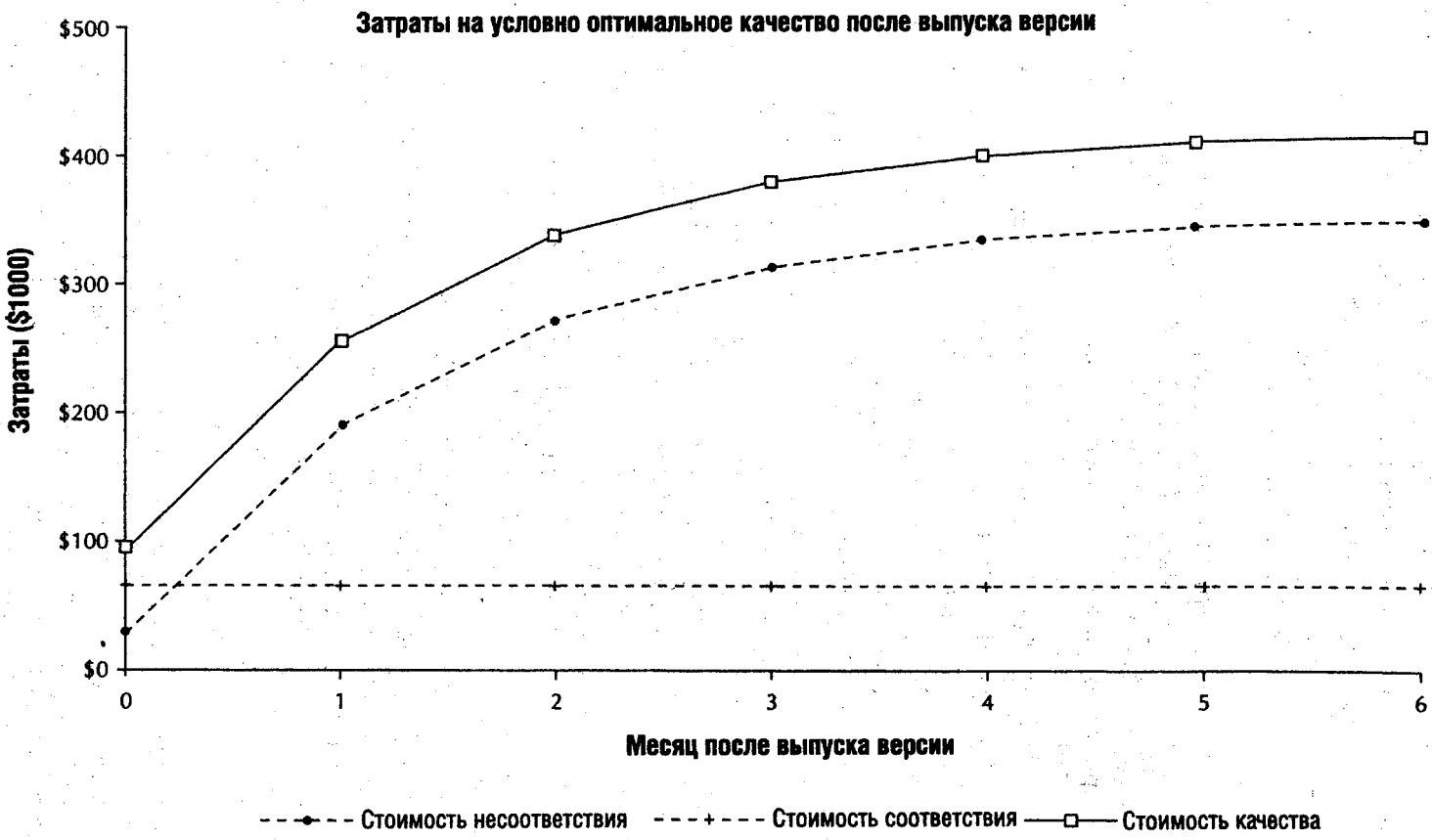


Рис. 5.3. Затраты на улучшенное (но не оптимальное) качество после выпуска версии

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Sumatra Test ROI (Using NPV Analysis)														
								Benefits Received in Period							Total Benefits
Testing	Costs	Release (R)	R+1M	R+2M	R+3M	R+4M	R+5M	R+6M	R+7M	R+8M	R+9M	R+10M	R+11M	R+12M	NPV Dollars
Staff (test development amortized)	$370,375														
External Labs	75,000														
Travel and Contingency	65,698														
Environments (amortized)	35,667														
Tools (amortized)	6,667														
Total Investment	**$554,007**														
Estimated Bugs Delivered to Test	2,500														
Estimated Must-Fix Bugs Delivered to T	2,000														
Estimated Defect Detection Percentage	80%														
Period	Total	0	1	2	3	4	5	6	7	8	9	10	11	12	
Estimated Field Bugs Avoided	1,600	0	800	400	200	100	50	25	13	6	3	2	1	0	
Estimated Found-Not-Fixed Bug Rate	20%														
Estimated Found-Not-Fixed Bugs	500														
Estimated Found-Not-Fixed Non-Bug R	67%														
Estimated Field Bugs Known	333	0	167	83	42	21	10	5	3	1	1	0	0	0	
Bugs Found and Fixed															
Must-Fix Bugs Found		$0	$867,200	$433,600	$216,800	$108,400	$54,200	$27,100	$13,550	$6,775	$3,388	$1,694	$847	$423	$1,683,540
Bugs Found and Not Fixed															
Deferred Bugs Found		$0	$12,500	$6,250	$3,125	$1,563	$781	$391	$195	$98	$49	$24	$12	$6	$24,267
Tests that Mitigate Risks															
Sales Subject to LCV or Reliability Loss	$ 2,000,000														
Likelihood of Sales Loss w/o Testing	50%	$0	$500,000	$250,000	$125,000	$62,500	$31,250	$15,625	$7,813	$3,906	$1,953	$977	$488	$244	$970,676
Guiding the Project															
Project Cost at Risk for Bad Tracking	$2,000,000	$200,000													
Reduction in Risk due to Testing	10%	$200,000													$200,000
Total Benefit (NPV)															**$2,878,482**
Return on Investment (NPV)															**420%**

Рис. 5.4. Окупаемость инвестиций в тестирование, уточненная в соответствии с чисто приведенной стоимостью

Я поднимаю эти вопросы не для того, чтобы отговорить читателя от проведения анализа окупаемости вложений в тестирование. Для тестировщиков-профессионалов, особенно тест-менеджеров, важно понимать финансовые последствия того, что они делают. Тем не менее при подсчете окупаемости вложений в тестирование я помню об ограничениях, присущих этим расчетам. Модели являются интуитивными на высоком уровне абстракции, но некоторые детали по-прежнему остаются проблемами. Эти проблемы могут вызвать недоверие к полученным цифрам, а также породить различные политические вопросы. В некоторых случаях лучше отказаться от подсчета окупаемости инвестиций в тестирование[1].

Возможно, это похоже на чудо, но это только оценка

Поскольку окупаемость инвестиций вычисляется приближенно, это только оценка. (Это то же самое, что и прогноз погоды или другое предсказание.) Продемонстрированный процесс оценки затрат, несмотря на все доверие к цифрам и данным, не является точной наукой. Мы смотрим на предыдущие проекты. Мы изучаем некоторые цифры. Мы манипулируем этими цифрами различными способами. Мы обсуждаем с другими людьми сделанные допущения. Из этого процесса возникают несколько документов, которые, возможно, предсказывают будущее. Совсем не волшебным образом мы можем предсказать, что солнце скрывается за горизонтом каждую ночь. Тем не менее есть что-то чудесное в том, что мы можем предвидеть итог такого сложного и уникального предприятия, как разработка, сопровождение или поставка системы хотя бы с некоторой точностью.

Я помню, как был очарован моими ранними оценками затрат в проектах. Я согласен, что часто не останавливался ни перед чем, чтобы постараться сохранить мои оценки точными, подгоняя проект под мои оценки иногда довольно бессмысленным образом. Если я утверждаю в графике, что задача займет пять дней, а это на самом деле не так, разумно ли говорить о том, что задача завершается через пять дней? Не нужно ли еще поучиться оценке затрат, если очевидно, что оценка неверна, и требуется вновь анализировать ситуацию?

Признание ошибок и извлечение из них уроков всегда лучше в долгосрочной перспективе, но делать это — не является сущностью человеческой природы. Часто более естественно обвинить кого-либо другого. Мы можем думать, что оценка была верной, но группа все испортила и делала дольше, чем, согласно оценке, должна была делать.

При разумном управлении мы должны пересматривать оценки, а не продолжать следовать ошибкам, вводя в заблуждение и обвиняя всех и вся. Если согласно карте вы должны ехать *сквозь* дерево, а *не объезжать*

---

[1] Например, Джоанна Ротман (Johanna Rothman) в ходе переписки со мной указала, что цифрами окупаемости вложений в тестирование легко манипулировать. Более того, анализ окупаемости вложений может привести к тому, что будут проигнорированы важные риски, как отмечает Сем Канер (Cem Kaner) в статье "Quality Cost Analysis Benefits and Risks" на www.kaner.com.

его, вы же не будете настаивать на точности карты и не попытаетесь пробить машиной дерево?

Есть предел волшебства в оценке затрат. Если вы заботитесь о получении хороших оценок, у вас в значительной степени больше шансов на это, нежели в случае, если вы берете цифры с потолка или колдуете над графиками Гантта и таблицами, не имеющими ничего общего с реальностью, но сообщаете старшим менеджерам информацию, которую они хотят услышать. Тем не менее процесс оценки затрат в проектах разработки, сопровождения или поставки систем полон потенциальных ошибок. Кажущиеся незначительными ошибки при составлении декомпозиции работ могут привести к серьезным неточностям в конечных оценках. Пожалуйста, не поймите меня неправильно: я делаю оценки затрат и делаю это тщательно. Если я не отношусь к оценке затрат тщательно, то я не управляю процессом и, следовательно, рискую. Но я научился не видеть волшебства в предвидении там, где есть только очень искусно выстроенная и тщательно и многократно перепроверенная приблизительная оценка.

Правило девяти месяцев

Хорошо известный афоризм управления гласит, что неважно, сколько пар участвует в процессе, потребуется, как минимум, девять месяцев плюс-минус несколько недель, чтобы родить ребенка. Другими словами, есть определенные задачи, имеющие определенную присущую им минимальную продолжительность независимо от числа людей, ответственных за их выполнение. Пренебрежение этим афоризмом — одна из классических ошибок при оценке затрат.

Одна из ловушек связана с тем, что Фред Брукс назвал мифическим человеко-месяцем. Некоторые при оценке проектов по созданию систем подсчитывают общее число необходимых человеко-месяцев, а затем считают, что могут произвольно сократить сроки проекта, добавив нужное количество людей. Брукс указал, что это миф, поскольку само добавление дополнительного числа людей может увеличить общее число человеко-месяцев, необходимых для выполнения проекта. Вот сформулированный по-другому знаменитый закон Брукса: "Если проект не укладывается в сроки, то добавление рабочей силы задержит его еще больше". Меньшее число людей могут работать более эффективно, а накладные расходы, связанные с координацией работ большего числа людей, перекроют все преимущества[1].

В некоторых случаях тестирования закон Брукса не работает. Например, в случае, если имеются скрипты для ручного тестирования, обычно можно использовать младших тестировщиков, потратив не более недели на их обучение и ввод в проект и поделить работу непосредственно на

---

[1] Книга Брукса "*Мифический человеко-месяц*" остается прекрасным исследованием тонкостей управления системными проектами. Впервые она была опубликована в 1975 г., а в 1995 г. вышло исправленное и дополненное издание. Мерилом общности и неистребимости проблем, возникающих перед группами разработки систем, является то, что книга по-прежнему остается актуальной, а новое издание было только дополнено и практически не изменилось.

уровне тестовых сценариев. Однако разработка тестовых данных, конфигурирование тестовой среды, написание тестовых сценариев, разработка автоматизированных тестовых скриптов или инструментов тестирования так же сложны для решения, как и программирование, и подчиняются закону Брукса.

Еще одна ловушка возникает из-за нереалистичного планирования на проектном уровне. Обычно тестировщикам довольно трудно понять это без достаточного опыта в программировании. Тем не менее можно поискать некоторые признаки того, что вас пытаются ввести в заблуждение.

Одним из таких признаков является декомпозиция работ проекта, не соответствующая критериям, определенным в главе 3. Пока остаются участники проекта, которые могут неверно понять свои задачи в проекте, график остается похожим на сочинительство.

Другим признаком является информация о том, что процесс составления графика ведется от произвольной даты поставки продукта. Еще хуже, если люди понимают, что для выполнения проекта в указанный срок необходимо существенное улучшение процессов, т.е. они просто полагают, что улучшение процессов произойдет само собой без приложения сил в этом направлении.

Если имеются подозрения насчет упущений в графике, нужно особенно тщательно определять связь задач и критерии начала фаз тестирования. В противном случае проблемы графика незаметно проникнут в тестирование и вас посчитают ответственным за них. (Более подробно эти вопросы рассматриваются в главах 6 и 7 при обсуждении вопросов планирования.) Более того, насколько ни были бы важны при оценке затрат планы на случай непредвиденных обстоятельств и резервы времени, они имеют еще большее значение при оценке подпроекта тестирования, который существует в рамках неадекватного графика или дефицитного бюджета для всего проекта разработки системы.

Осторожно! Незаинтересованные лица!

Быть честным перед самим собой означает быть честным относительно рамок подпроекта тестирования в объемлющем проекте и относительно того, как другие люди видят проект, в котором я участвую. Подпроект тестирования обычно прочными нитями связан с тем, что происходит с проектом вне тестирования. Обычно это связи с другими участниками проектной группы. Эти люди непосредственно заинтересованы в успехе проекта. Поэтому, хотя нужно тщательно управлять взаимосвязями с логической точки зрения, по всей видимости, у вас не будет проблем в мотивации взаимодействия. Как заметил однажды один толковый менеджер: "Пока мы все в одной лодке, не вижу большого смысла выбрасывать на скалы нос или корму".

Однако иногда мы зависимы от людей, не вовлеченных в проект или, возможно, даже враждебных ему. Это мощный фактор, влияющий на оценку затрат на тестирование.

В одном из проектов моя группа тестирования имела три таких зависимости, которые оказались исключительно проблемными. Во-первых,

была группа системного администрирования, которая не сделала своим приоритетом настройку тестовой лаборатории. Они работали над многими другими проектами. Им казалось, что наш проект будет отменен. Они просто не хотели связываться с настройкой десятка тестовых рабочих станций с одинаковыми конфигурациями, которые моделировали конфигурации развернутой системы.

Во-вторых, была группа, отвечавшая за администрирование существующей системы, для которой мы готовили замену. Нам нужен был доступ к той системе в качестве *эталонной платформы* или *тестового оракула* для определения корректности результатов работы системы, предназначенной для тестирования. (Тестовый оракул подробно рассматривается в главах 10 и 11.) Однако поскольку операторы существующей системы оставались без работы в случае успешного завершения проекта, в лучшем случае они оказывали нам помощь с выражением неприязни.

Наконец, группа информационных систем, занимавшаяся поставкой программных продуктов, для которых эта (большая) компания имела лицензии, не считала, что нам нужны десятки лицензий для системы отслеживания дефектов. Эта проблема особенно обострилась, когда мы попросили настроить двух удаленных клиентов для компании, которая разрабатывала большую часть кода.

Ни одна из задач, которую должны были выполнить для нас эти незаинтересованные стороны, не заняла бы больше чем один рабочий день тестировщика, включая разъяснение необходимости, помощь в обеспечении соответствующего доступа и проверку завершения задачи после выполнения тестов. На деле группа тестирования потеряла, по крайней мере, три человеко-недели, пытаясь разобраться с кумулятивным эффектом задержек и пассивного сопротивления. Все время между планированием и выполнением тестов занимало решение этих проблем. Проблемы были бы несерьезными, если бы люди в других группах, от которых мы зависели, просто выполняли свои обязанности. Поскольку они не были заинтересованными лицами проекта, у них отсутствовали стимулы делать это. Такое поведение незаинтересованных лиц имело исключительно разрушительные последствия для подпроекта тестирования и проекта в целом.

Рассмотренный пример демонстрирует, что может произойти в том случае, когда имеются зависимости вне проекта. Определение контекста проекта очень важно, поскольку позволяет рассмотреть тех, кто способствует выполнению проекта и проведению в нем тестирования, а кто — нет. Если я вижу, что возможны задержки со стороны незаинтересованных лиц, я ограничиваю зависимость от тех людей, чьи интересы не соответствуют успешному выполнению проекта, до возможного минимума, лучше всего до нуля. Если по какой-либо причине я не могу этого сделать, то прилагаю усилия, чтобы задачи с такими зависимостями не попадали на критический путь, и предупреждаю руководство о том, что я ожидаю возникновения проблем.

Для любых задач с такими зависимостями я создаю приличный резерв времени. Я также рассчитываю на то, что придется предпринять значи-

тельные усилия для выполнения моей части этих и всех зависимых задач. Наконец, я получаю солидный однозначно сформулированный документ с обязательством участников и менеджеров выполнить все зависящее внутри проекта и с определением быстрого способа эскалации проблем на необходимый уровень руководства. В следующих двух главах по планированию тестирования будет показано, как я это делаю.

Когда "продают" неудачные оценки

Предположим, что мы завершили процесс оценки затрат, включающий в себя создание бизнес-плана и определение реалистичной оценки, и получили заключение: "Мы не можем себе это позволить, и у нас нет столько времени. Вот сколько у вас есть денег, вот вероятные сроки; теперь идите и тестируйте систему". Что теперь? Здесь я бы попытался найти разумный компромисс между глубиной и широтой покрытия. В качестве вспомогательного средства я бы взял результаты анализа риска качества.

Если группа управления проектом предлагает пойти на риски, которые мне кажутся неразумными, я стараюсь объяснить свою позицию. Когда я это делаю, то имею в виду, что предприниматели и технологи, как правило, больше привержены риску, чем тестировщики. Обычно мне отвечают примерно так: "Да, Рекс, мне понятно твое мнение, но я не собираюсь тратить больше денег и откладывать сроки, чтобы снизить эти риски". Если по какой-либо причине я прекращаю тестирование раньше, чем должен, то тщательно документирую риски, которые остались непокрытыми тестами.

Стремление к компромиссу по поводу объема тестирования и достаточного бюджета и графика для поддержки такого тестирования — не то же самое, что отказ от своих оценок. Другими словами, если я защищаю реалистичный бюджет и оценку затрат, руководство может либо поддержать эту оценку, либо урезать рамки тестирования и попросить меня согласиться с меньшим бюджетом и более короткими сроками. Но иногда вместо этого меня просят сделать то, о чем ранее я говорил так: "Возьмите X долларов и Y месяцев за две трети стоимости в половину времени и посмотрите, что получится". Тим Коомен указал мне при рецензировании предыдущей главы: "На совещании [посвященном обсуждению оценок] никогда не соглашайся безвозмездно убрать часы, например: "Мы будем работать немного напряженнее и, я думаю, сможем завершить все в указанный вами срок",– поскольку в этом случае ты теряешь доверие". Я искренне согласен с его утверждением.

Наконец, может оказаться отдельной трудной задачей убедить руководство в реалистичности оценок, если у него уже имеется опыт работы с фиктивными оценками других менеджеров или предшественников в отделе тестирования. Если предыдущий тест-менеджер, скажем, брал цифры с потолка, а затем не мог достичь поставленных целей по срокам и бюджету, руководство могло придти к мнению, что ни один тест-менеджер на самом деле не знает, как проводить оценку затрат. Я работал в ком-

паниях, в которых вся группа технических менеджеров обвинялась в этом. Справедливо или нет, но если вы начинаете в условиях дефицита доверия к оценкам, необходимо преодолеть эти обстоятельства. Как говорится в старой пословице, если вы оказались в яме, первым делом нужно прекратить копать ее дальше. В этой ситуации нужно особенно тщательно готовить только наилучшие, наиболее аккуратные оценки, но вы же всегда так делаете, верно?

Системы с повышенными требованиями к безопасности или к выполняемой целевой задаче

Конечно, положительное сальдо окупаемости инвестиций само по себе неплохо, однако в жизни не все решают деньги. В некоторых случаях системы, которые мы разрабатываем, имеют большое значение для безопасности людей или выживания бизнеса, что выходит за пределы возможностей любых финансовых моделей оценки.

Одна из сильных сторон формального анализа рисков качества, в частности метода анализа видов ошибок и их влияния, состоит в том, что он отличается выявлением действительно ключевых рисков качества и соответствующих видов ошибок для систем, от которых зависит жизнь человека или непрерывное функционирование бизнеса. Если вы работаете в такой среде, необходимо тщательно следить за тем, чтобы не потерять достижения анализа рисков качества в ходе оценки затрат, составления графика и планирования бюджета. Перед обсуждением активности, задачи или ресурса еще раз нужно проверить, что при исследовании этих критических типов рисков необходимо учитывать не только затраты. Если вы уверены, что сокращение тестирования выльется в дополнительный риск для человеческой жизни, ваша моральная, юридическая и профессиональная обязанность — предпринять решительные меры.

Я бы порекомендовал для начала опробовать все возможные способы для разрешения проблем, эскалируя их на разные уровни руководства компании. Вы подвергнете опасности весь свой политический капитал, если не будете использовать его тонко и осторожно. Важно также понимать, что вы можете ошибаться. На самом деле, все участники проекта могут до некоторой степени заблуждаться в своем понимании рисков. Вы, возможно, особенно опасаетесь рисков, в то время как другие не столь осторожны. Открытость к компромиссам и склонность поступаться принципами — это две разные вещи. Для эффективного взаимодействия необходимо слушать и понимать, что говорят другие, а также уметь объяснить свою позицию.

Если вы чувствуете, что риски непреодолимы, а другие не разделяют вашу озабоченность, вероятно, вы не донесли свою точку зрению надлежащим образом. (В главе 15 рассматривается трагический пример неудачного взаимодействия при выполнении полета американского космического челнока "Челленджер" в 1986 г.) Люди могут понимать и принимать вашу *озабоченность*, однако это не значит, что они понима-

ют, *почему* вы озабочены. Если вы чувствуете, что остаются риски, угрожающие жизни, ваша обязанность — донести проблему с помощью данных в не оставляющей вопросов форме людям, которые могут принимать решения.

Если вы исчерпали способы разрешить проблему обычными путями, что дальше? Возможно, у вас есть право сообщить об определенных действиях, которые могут угрожать жизни или общественной безопасности. Вероятно, вас заставит действовать ваша совесть. Тем не менее выступать в качестве доносчика — не самый легкий выбор. Рекомендую в этом случае перед принятием решения в спокойной обстановке посоветоваться с надежными друзьями, семьей или взять юридическую консультацию.

Внедрение усовершенствований

Научиться создавать реалистичные, точные, правдивые оценки затрат и убеждать руководство поверить в них, а также управлять подчиненными и взаимодействовать с коллегами было, вероятно, наиболее трудным для меня при переходе от непосредственного исполнителя к роли тест-менеджера. Люди стремились к предвидению с давних времен, мифология полна рассказов о тех, кто умел предсказывать будущее. Я не претендую на то, чтобы достичь возможностей Пифии, главной жрицы Дельф, но я бы порекомендовал следующие шаги для улучшения методов оценки затрат.

1. Изучите тему оценки затрат. Это важно для тех из нас, кто должен проводить оценку затрат на тестирование систем, для того, чтобы иметь основные инструменты под рукой. Следовательно, важно научиться составлять хорошо организованную декомпозицию работ. Также важно понимание, как соединить бюджет и бизнес-план. Поскольку бизнес-план тестирования основан в значительной степени на стоимости качества, изучение этой темы также важно. Вероятно, вы обнаружите, что по мере изучения этих тем вы станете не только более компетентным как тестировщик, но и заслужите больше доверия. Учитесь говорить на языке менеджеров проектов. Вырабатывайте навыки, необходимые для представления и защиты проекта тестирования понятными им словами. Учитесь обсуждать предложения по урезанию бюджетов тестирования иначе, чем просто утверждать: "Вы бы не предложили этого, если бы беспокоились о качестве". Нужно уметь говорить о конкретных рисках, затратах и альтернативах, а также вносить контрпредложения, которые позволяют достигнуть взаимного удовлетворения и поддержки руководства без озлобления сторон.

2. Учитесь на прошлых проектах, на нынешних проектах и у других людей. Основные методы оценки затрат на тестирование — это пустые сосуды, которые вы можете и должны заполнить данными.

Посмотрите на предыдущие проекты и замените гипотетические данные в таблицах и декомпозиции работ, которые приводятся в этой книге, данными из вашего опыта. Сбор данных из текущего и будущих проектов поможет оценкам в будущем. Отслеживайте текущие проекты и сравнивайте фактические результаты с оценками. Исследуйте мертвые зоны в оценках. Какие задачи вы забыли включить? Почему? Для каких действий вам неясно, как провести оценку затрат? Что нужно узнать и какие дополнительные данные собрать для более точной оценки? Использование механизма базовой версии, имеющейся во многих инструментах для управления проектами, является хорошим способом измерить точность оценки. Наконец, поделитесь мыслями, идеями и вопросами с другими. Разработайте модели для получения хороших оценок и отправьте их коллегам. Покажите их на конференциях по тестированию и управлению проектами. Опубликуйте их в статьях и книгах. Это поможет вам не только стать лучше, но и оказать сообществу тестировщиков услугу, поскольку мы все нуждаемся в улучшении оценок затрат.

3. Обдумайте, какие есть препятствия. В предыдущих разделах определено несколько возможных препятствий для получения хороших оценок. Не все из них применимы в вашем случае. Рекомендую потратить время на выявление проблем в рамках вашего контекста, которые могут помешать выбору эффективных практик оценки затрат. Выработайте идеи по преодолению этих препятствий. Если некоторые идеи не срабатывают так, как должны, не сердитесь и не опускайте рук (по крайней мере, надолго); просто измените подход и попытайтесь применить его в следующем проекте. Поскольку книга Фреда Брукса "*Мифический человеко-месяц*" по-прежнему актуальна, спустя 25 лет после первого издания, значит, описанные им проблемы непросты, и вы не решите их за один подход.

4. Практикуйтесь, практикуйтесь, практикуйтесь. С каждым разом, когда вы начинаете составлять декомпозицию работ, бюджет, строить бизнес-план на основе окупаемости инвестиций в тестирование, обсуждать время и ресурсы с руководством для получения оценок и управлять проектом в соответствии с этими оценками, вы обнаруживаете, что это получается все лучше и лучше. Настойчиво делайте это и после завершения каждого проекта спрашивайте себя, что вы узнали нового об оценке затрат.

5. Используйте по максимуму то, что есть в наличии. Если вам нужна практика, вы должны *хотеть* практиковаться. Если вас просят провести тестирование с недостаточными ресурсами, не терзайтесь при отказе. Практически всегда ищется компромисс. Если от вас не хотят ничего противозаконного или неэтичного, вероятно, в ваших интересах держаться в проекте до конца. Я научился многому

в оценке затрат и в проектах, которые закончились катастрофой, и в проектах, которые прошли успешно.

Располагая утвержденной оценкой затрат, можно перейти к следующему этапу. В следующих двух главах мы рассмотрим планирование тестирования. Каково то множество деталей, которые необходимо собрать вместе для успешного проекта? Если рассматривать оценку затрат как карту, то план проведения тестирования — это направления движения по ней. Но в этом случае направления движения имеют отношение к группе водителей, которые должны приехать в одно и то же место, не столкнувшись друг с другом или с кем-нибудь еще. Как этого добиться?

ГЛАВА 6

Изучение и обсуждение содержания проекта: планирование тестирования

Некоторые менеджеры уверены, что планирование завершается, как только они подготовили и утвердили у руководства декомпозицию работ, график и бюджет. Хотя я согласен с тем, что качественная оценка затрат, которую поддерживает руководство, имеет большое значение для написания эффективного плана для любого проекта, в рамках процесса оценки затрат множество деталей остается скрытым, непознанным и не доведенным до заинтересованных лиц. При подготовке оценки мы делаем допущения, определяем зависимости, перескакиваем через конкретную работу и принимаем за аксиому, что определенные процессы низшего уровня просто имеют место. Это опасно, если мы берем за основу непосредственно подпроект тестирования в контексте рабочей группы проекта, который определяет, что мы делаем, почему мы это делаем и как мы зависим друг от друга.

Но откуда возьмется тревога в рабочей группе проекта, если мы сначала не проникнемся ею сами и затем не просветим наших коллег? Подпроекты тестирования состоят из сети сложных, взаимозависимых временных активностей, которые увязаны со многими другими активностями в объемлющем проекте. Даже опытные тест-менеджеры сталкиваются с трудностями, предугадывая детали того, что будет происходить. Чтобы работать в реальном мире, необходимо вникнуть в суть мелочей. Как только мы это сделаем, мы должны обсудить наше понимание с другими и получить их поддержку.

В некоторых случаях процесс планирования тестирования имеет осязаемый результат – зафиксированный на бумаге план тестирования. Этот план может быть частью информации, которую группа проекта рассчитывает получить от группы тестирования. Для удобства понимания документированных планов тестирования проектная группа может предполагать, что план тестирования будет разработан в соответствии с некоторым шаблоном, разработанным внутри компании либо являющимся стандартом в отрасли. В других случаях план тестирования мо-

жет появиться в виде небольшого документа, а то и вовсе не фиксироваться на бумаге. В этой главе будем предполагать, что мы намерены подготовить документированный план тестирования. В противном случае наибольшее значение имеет то, что группа тестирования понимает, что делать, а проектная группа согласна поддержать план и приняла результаты планирования как достаточные.

Процесс планирования тестирования

Процесс 5 – это процесс, который я использую для планирования тестирования. Шаблон плана тестирования, который вы применяете, или спо-

Процесс 5. Процесс планирования тестирования

№ этапа	Этап	Выполнено?
1.	Исследовать, продумать, собрать и документировать стратегию, тактику и составляющие работы подпроекта тестирования.	☐
2.	Обсудить и документировать совместные работы подпроекта тестирования и объемлющего проекта разработки.	☐
3.	Завершить подготовку и документирование оставшихся логических деталей и подробностей плана, в том числе рисков самого подпроекта тестирования и определений глоссария тестирования и проекта. Указать все цитируемые документы. Подготовить одностраничное резюме подпроекта тестирования.	☐
4.	Разослать план для закрытого (автономного) рецензирования, обычно сначала сотрудникам группы тестирования, а затем на более широкое обсуждение среди заинтересованных лиц и участников проекта. Собрать отзывы и пересмотреть план, повторив этапы 1 – 3 по необходимости. Изучить любые изменения в имеющемся графике и бюджете (превышающие оговоренные резервы времени и планы на случай непредвиденных обстоятельств), внесенные в результате процесса планирования, и получить поддержку этих изменений у руководства.	☐
5.	Разослать план для открытого рецензирования. Провести совещание по итогам рецензирования с участием всех заинтересованных лиц. Сделать все необходимые заключительные уточнения и удостовериться, что с учетом всех согласованных на совещании изменений план для подпроекта тестирования зафиксирован на бумаге и утвержден.	☐
6.	Пересмотреть оценку графика и бюджета на основе новой информации, полученной в ходе процесса планирования, включая использование ресурсов. Если в результате происходит сдвиг сроков или увеличение бюджета за пределы резервов или планов на случай непредвиденных обстоятельств, эскалируйте проблемы для решения на соответствующий уровень руководства. Обсуждение новых бюджета и графика может повлечь повторение этапов планирования или переработку оценки затрат.	☐
7.	Поместите план в библиотеку проекта или в систему управления конфигурацией. Ведите контроль изменений документа.	☐

Система тестов (Test System)

Тестовая среда, средства тестирования и процессы выполнения тестов, которые группа тестирования будет применять для проверки качества системы, предназначенной для тестирования.

соб, который вы выбираете для документирования плана тестирования, может определенным образом повлиять на содержание и порядок шагов. Например, некоторые процессы тестирования требуют разработки генерального плана тестирования (master test plan) для всего тестирования в проекте и отдельных частных детализированных планов тестирования для каждой фазы или уровня тестирования (например, комплексного и системного тестирования). Информация, попадающая в каждый из документов, оказывает влияние на ход процесса планирования тестирования и может означать, что в результате мы получаем два-три процесса планирования тестирования, проходящих одновременно.

У Джамала есть план

Джамал начал готовить план 12 сентября, но 17 сентября, утвердив пересмотренную финальную версию оценки затрат, он взялся за план всерьез. Он начал с обзора результатов анализа рисков качества и документирования рамок тестирования в таблице с колонками ДА/НЕТ. Каждый риск, для которого заинтересованные лица рекомендовали полное и сбалансированное тестирование, отправлялся в колонку ДА, а риски более низкого приоритета помещались в колонку НЕТ.

Затем Джамал приступил к выбору подходящей стратегии тестирования для каждого риска качества, попадающего в рамки тестирования. Он уже принял решение по поводу некоторых стратегий при составлении декомпозиции работ. Например, его группа будет использовать автоматизированное тестирование для большей части функционального тестирования и для тестирования производительности. Однако некоторые детали оставались невыясненными, и теперь он приступил к их обдумыванию. Тестирование обработки ошибок и восстановления системы после сбоев, по его представлению, должно включать в себя полный обзор сообщений об ошибках, определенных для серверной и клиентской сторонах системы, и, вероятно, должно основываться на просмотре исходных файлов сообщений, а не на попытках исчерпывающим образом смоделировать каждый вид сбоя. Тестирование нагрузки, производительности и объема, вероятно, потребует инструментов для работы с API-интерфейсом, чтобы общаться с API на стороне сервера, и потребует смоделировать определенные виды нагрузки в дополнение к готовым коммерческим инструментам нагрузочного тестирования на основе графического интерфейса.

По мере решения некоторых проблем он начал аккумулировать состав тестовой среды. Бюджет позволял ему приобрести один дополнительный серверный кластер, который Джамал планировал добавить к трем уже имеющимся кластерам, оставшимся от предыдущего проекта для SpeedyWriter версии 3.0. (Он планировал оставить один из четырех кластеров для проекта SpeedyWriter 3.0, предназначенный специально для тестирования сопровождения.) Поскольку предполагалось выполнять большие объемы ручного и автоматизированного тестирования одновременно, он запланировал приобрести десять дополнительных клиентских систем помимо тех пяти, которые можно было забрать из подпроекта тестирования SpeedyWriter 3.0. Джамал получил эти цифры, основываясь на представлении, какой объем тестирования будет выполняться в каждый определенный момент и можно ли раздельно использовать серверные кластеры для разных видов тестирования или нужно для каждого вида тестирования иметь свой собственный кластер.

Джамал просмотрел исходные оценки, подробно исследуя, какие тесты на каких конфигурациях будут исполняться. Используя тот же метод анализа, что и при получении оценок, но фокусируя внимание на загрузке серверов и рабочих станций в часах, а не на общей продолжительности работ, Джамал еще раз убедился, что ему нужны четыре серверных кластера и восемь дополнительных рабочих станций (см. рис. 6.1). В вычислениях в нижней части таблицы он проверил, что четыре кластера учтены только один раз.

Джамал очень обрадовался, обнаружив, что детальные расчеты показывают, что предварительные грубые оценки затрат на аппаратуру достаточно точны и умеренны. Но какая конфигурация (аппаратное, программное и сетевое обеспечение) будет у тестовой среды? Джамал решил построить диаграмму, чтобы визуально представить тестовую среду.

При построении диаграммы конфигурации тестовой среды Джамал пришел в замешательство. Дополнительный серверный кластер поместится в лаборатории, поскольку три системы можно расположить в стой-

Overlapping Phase Period (excl. Localization & Usability)	Effort	Duration	Count
Total Person Hours/Hours	255	494	
Est. Person Weeks/Weeks	13	6	
Proposed ATE Staffing	2		
Proposed MTE Staffing	1		
Proposed Tech Staffing	4		
Test Pass Duration (Weeks)	2		
Test Environments (Clusters, Wkstns, Nets) Required (w/ Some Sharing)			4
Server Cluster 1 (East) Usage (Taking Sharing into Account)			1
Server Cluster 2 (West) Usage (Taking Sharing into Account)			1
Server Cluster 3 (North) Usage (Taking Sharing into Account)			1
Server Cluster 4 (South) Usage (Taking Sharing into Account)			1
Workstation Usage			13

Рис. 6.1. Начальный подробный план использования оборудования

ке, но пять из десяти дополнительных рабочих станций некуда ставить. Дополнительного места в лаборатории не было, а Джамалу не хотелось разносить оборудование по разным помещениям. Если попытаться разместить все рабочие станции в лаборатории, сотрудникам не останется пространства для работы. "Хм, – думал он, – что же делать?" Невозможно, чтобы тест-менеджер, три тестировщика и четыре младших тестировщика работали одновременно на десяти рабочих станциях, поскольку в этом случае нельзя заниматься локализацией дефектов, написанием отчетов о них и читать электронную почту, пока выполняются тесты, и т.п. Его опыт показывал, что две рабочие станции на сотрудника – хорошо работающее эмпирическое правило. Как получить это соотношение?

Он вспомнил обучающий курс по руководству тестированием, на котором рассказывалось про использование младших тестировщиков, работающих по контракту, во вторую смену. "Предложите дополнительную оплату за час-два работы в неудобную смену, – говорил инструктор. – И у вас не должно быть проблем с поиском контрактников для работы в вечерние часы". "Хм, – подумал Джамал, – посмотрим, что из этого получится". Он переработал план использования оборудования из расчета 80-часовой (а не 40-часовой) рабочей недели. (В среднем тестировщик тратит 30 часов в неделю на выполнение тестов из-за совещаний, обсуждений в группе и т.п., но сами рабочие станции должны быть доступны для проведения тестирования в течение всего рабочего времени в обеих сменах.) Результат, как показано на рис. 6.2, позволяет достичь цели за счет уменьшения количества необходимых рабочих станций примерно до 10. В любой момент число работающих в тестовой среде тестировщиков не превысит пяти человек, что в точности соответствует отношению 2:1 между числом рабочих станций и числом тестировщиков. Пересмотрев бюджет, Джамал увидел, что экономия на покупке рабочих станций практически покроет премии за работу в вечернюю смену, которые он заплатит младшим тестировщикам, работающим по контракту. Перерасход

Overlapping Phase Period (excl. Localization & Usability)	**Effort**	**Duration**	**Count**
Total Person Hours/Hours	255	494	
Est. Person Weeks/Weeks	13	6	
Proposed ATE Staffing	2		
Proposed MTE Staffing	1		
Proposed Tech Staffing	4		
Test Pass Duration (Weeks)	2		
Test Environments (Clusters, Wkstns, Nets) Required (w/ Some Sharing)			4
Server Cluster 1 (East) Usage (Taking Sharing into Account)			1
Server Cluster 2 (West) Usage (Taking Sharing into Account)			1
Server Cluster 3 (North) Usage (Taking Sharing into Account)			1
Server Cluster 4 (South) Usage (Taking Sharing into Account)			1
Workstation Usage			9

Рис. 6.2. Пересмотренный подробный план использования оборудования с учетом двухсменной работы

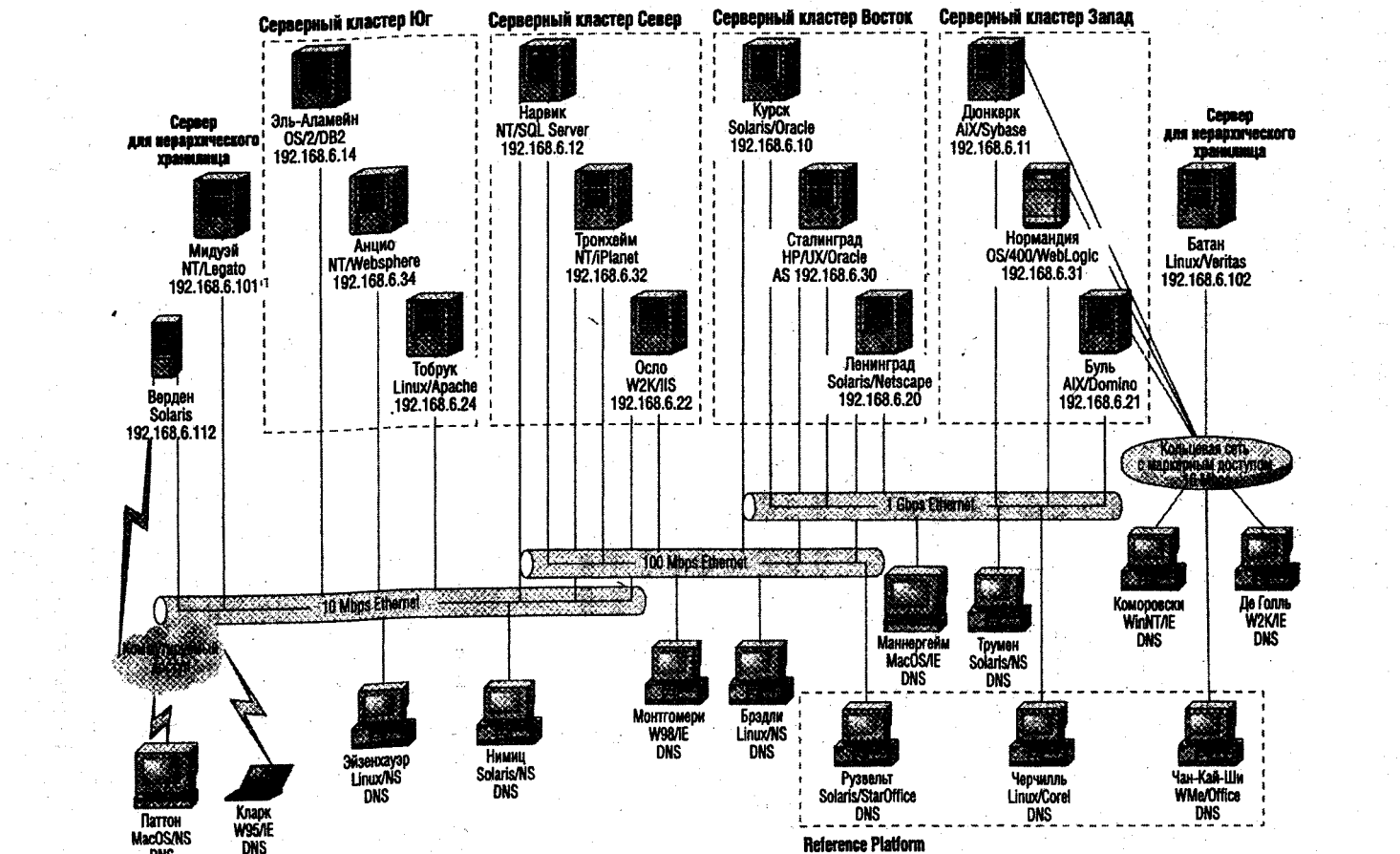


Рис. 6.3. Тестовая среда проекта "Суматра"

средств составит около $4000, что вписывается в 10% средств на непредвиденные расходы. Джамал вновь переработал диаграмму конфигурации тестовой среды (см. рис. 6.3).

"Хорошо, я разобрался со средой, – думал он, – но кто будет выполнять тесты?" К этому моменту он определил двух штатных сотрудников (Лин Цу и Эмму), одного нового тестировщика, которого предстоит нанять, и четырех младших тестировщиков, которых возьмут на работу по контракту на период тестирования в проекте. Теперь было уже понятно, что младшие тестировщики будут работать в две смены. Джамал зафиксировал эту информацию, а также различные следствия из нее в плане тестирования.

Джамал вновь обратился к декомпозиции работ. Во-первых, он нашел в графике Гантта все основные контрольные точки, уделяя особое внимание зависимостям от других групп, например получение версий на тестирование, и включил их в план тестирования. Он исправил ошибку в начальной версии декомпозиции работ, связанную с некоторыми тестовыми циклами системного тестирования продолжительностью более одной недели. Сначала он немного огорчился, но затем понял, что даже экспертная оценка и тщательная проверка не способны устранить все ошибки ни в исходном коде, ни в планах проекта.

Он начал подробно обследовать каждую задачу, каждую активность, каждый процесс, включенные в декомпозицию работ. В разделе, посвященном разработке тестов, он упомянул не только конкретные области, на которые необходимо нацелить новые тесты, но и подробные сведения о том, как эти тесты разработать. В частности, он выделил основной элемент работ по созданию тестов – разработку тестов безопасности и секретности. Эти тесты предназначены для тестирования вручную. Их будет выполнять Лин Цу, используя графический интерфейс (браузер), различные API-интерфейсы, командные строки NT и UNIX и интерфейсы баз данных. Тесты будут нацелены на проверку как структуры кода, так и поведения системы. Лин Цу будет разрабатывать тестовые данные, тестовые сценарии, макросы, скрипты, процедуры и т.п. Он также рассчитывает, что Лин Цу исследует известные уязвимые места в системах, построенных на базе браузеров. Такая степень детализации вместе со стратегиями тестирования, представленными ранее, по расчетам Джамала, позволит Эмме, Лин Цу и новому тестировщику получить достаточно информации о том, какие виды тестов необходимо разработать. И, таким образом, он может оставить им вопросы детального проектирования и разработки тестов.

Из плана тестирования для SpeedyWriter 3.0 Джамал скопировал в новый план разделы об отслеживании дефектов и тестовых сценариев. Этот раздел описывает используемые поля системы отслеживания, но не содержит никакого подробного описания процессов выполнения тестов и написания отчетов о дефектах. Чтобы удостовериться в том, что вновь принятые тестировщик и младшие тестировщики понимают, как устроены эти критические процессы, Джамал решил провести однодневное обучение, сразу после того как они приступят к работе, с раздачей слайдов и с записями лекций.

Младшие тестировщики, вероятно, не очень хорошо разбирающиеся в тестировании, получат кураторов из числа инженеров-тестировщиков. (Он пометил для себя: не забыть раздать инженерам-тестировщикам заметки о том, как стать хорошим куратором, а также перечитать их самому, поскольку он будет курировать вновь принятого на работу инженера-тестировщика.) Куратор должен проверять все результаты тестирования и отчеты о дефектах, полученные младшим тестировщиком. Джамалу нравилось, как шла работа в этом направлении в проекте SpeedyWriter 3.0. Сделав несколько небольших изменений, чтобы следовать линии предыдущего проекта, Джамал завершил первый большой этап планирования тестирования — документирование внутренних аспектов тестирования.

№ этапа	Этап	Выполнено?
1.	Исследовать, продумать, собрать и документировать стратегию, тактику и составляющие работы подпроекта тестирования.	☑

Теперь переходим к трудоемкой части: подготовке беспрепятственной передачи результатов и процессов совместной работы с остальной частью группы проекта. Для начала Джамал посчитал полезным нарисовать картинку (см. рис. 6.4), иллюстрирующую, каким образом происходит взаимодействие тестировщиков с заинтересованными лицами, включая разнообразные потоки информационных продуктов и услуг вне группы тестирования и передаваемые материалы и услуги тестирования внутри группы.

"Хорошо, — думал Джамал, — сейчас я просто должен договориться о том, как будут идти потоки внутрь и за пределы группы тестирования". Сначала он позвонил Кейту Ли, менеджеру системных операций и администрирования. У него было два вопроса к Кейту, один требовал немедленного ответа, другой предполагал более длительные раздумья.

"Привет, Кейт, как дела?" — сказал Джамал, как только Кейт ответил ему на звонок.

"Новый день, новый пожар. Ребята из техподдержки просят помощи в конфигурировании странного оборудования заказчика, и вся моя группа собралась там".

"Ясно, у тебя есть минутка?"

"Конечно, со временем всегда беда", — засмеялся Кейт.

"Поможешь мне в двух вопросах: один срочный, другой нет. Ты же знаешь, что мы должны довольно скоро начать тестирование первого инкремента Суматры".

"Скоро начать тестирование? Ой, проект только начался в этом месяце!"

"Да, но мы используем итерационный подход. Больше тестируем и раньше начинаем. У меня уже есть утвержденные график и бюджет. Бюджет включает в себя закупки оборудования. Если я тебе предоставлю документацию на этой неделе, сможешь ли ты со своей командой начать работы по конфигурированию прибывающего оборудования для тестовой лаборатории, скажем, в среду?" — Джамал сверился со своим графиком.

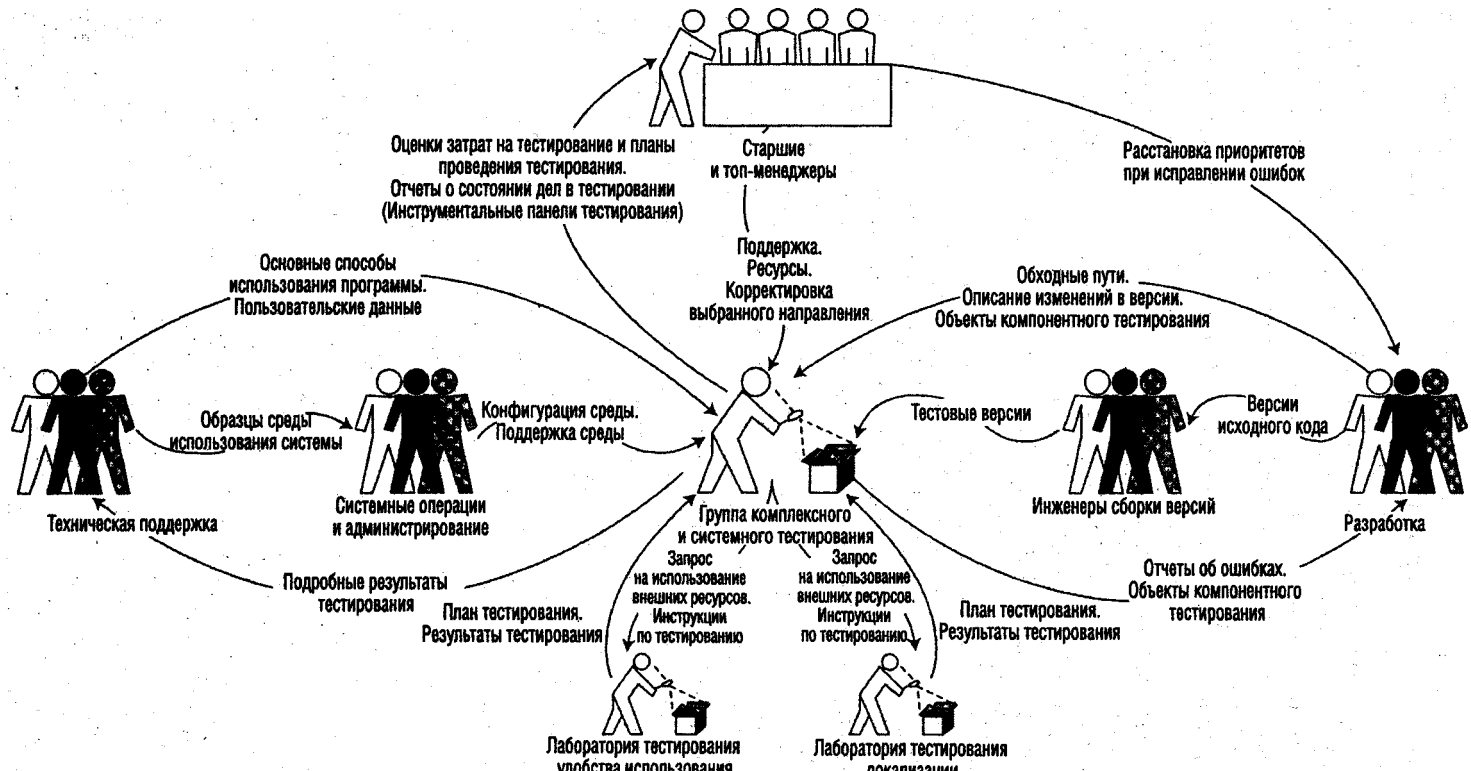


Рис. 6.4. Контекст комплексного и системного тестирования проекта “Суматра”

"Да, часть работы начнем. Серверы требует больше времени, а клиентские машины — однозначно начнем. Когда мы должны завершить настройку?"

Джамал снова взглянул на график Гантта. "1 октября. Начинаем в следующий понедельник. Я рассчитываю, что Эмма и Лин Цу приступят к выполнению некоторых тестов для комплексного тестирования в тестовой среде, поэтому мне потребуется помощь твоих людей".

"Чем дальше, тем лучше".

"Перехожу к следующему вопросу. Я бы хотел, чтобы ты выделил двух человек из своей группы в качестве ключевых контактных лиц для разрешения проблем тестовой среды. Это позволило бы иметь четкое направление для эскалации проблем".

"Так, для серверов я бы выделил Петру Фейхимян (Petra Fahimian). Она недавно работает, но уже зарекомендовала себя. Для клиентских машин, что ты думаешь по поводу кандидатуры Индера Ванешватана (Inder Vaneshwatan)?" — спросил Кейт.

"Хм, — осторожно сказал Джамал, — знаешь, Кейт, в проекте SpeedyWriter 3.0 был случай, когда Индер неверно настроил все клиентские машины, и мы потеряли неделю тестирования. Буду откровенен с тобой, Кейт. Тот случай и то, как он отреагировал на наше беспокойство, произвело сильное впечатление на меня и на других сотрудников моей группы".

"Понимаю тебя. У меня был с ним серьезный разговор обо всей ситуации и о его реакции. Это снова не повторится. Индер обещал мне, и я обещаю тебе. Конечно, если что-то действительно произойдет, передай проблему мне. Пожалуйста, проверь, что твои люди знают об этом. Если Индер или Петра не займутся твоей проблемой в течение получаса после твоего звонка или сообщения на мой пейджер, сразу звони мне".

"Хорошо, Кейт, ты был довольно откровенен. Я включу процесс эскалации в план тестирования. Посмотришь на этой неделе первую версию плана? Сообщи, если обнаружишь проблемы".

"Будь уверен, брат. Вернусь к своим сегодняшним проблемам," — язвительно хихикнул Кейт.

"Пока", — сказал Джамал. Он повесил трубку и подумал, что одно дело сделано, осталось еще два. Он снова снял трубку и позвонил Дженни Кауфман, чтобы обсудить порядок эскалации блокирующих ошибок и передачи отчетов об ошибках. Потом он позвонил Ясухиро Канагаве (Yasuhiro Kanagawa) и обсудил процесс передачи версии на тестирование.

После отработки интерфейсов с группами своих коллег Джамал перешел к изучению вопросов о критериях начала и завершения тестирования, а также о подготовке отчетов о состоянии дел. Джамал работал под началом Джона Албертсона чуть более года, поэтому он знал, что Джон будет настаивать на таких отчетах. Гарольд Джоунз пришел в компанию недавно, поэтому Джамал не знал, что ожидает Гарольд от тестировщиков. Но Джамал решил не спрашивать его, а взять на себя подготовку проекта критериев и затем отправить его вместе с инструментальной панелью состояния дел в тестировании из проекта SpeedyWriter 3.0. (Подготовка отчета о ходе тестирования и инструментальная панель рас-

Критерии начала комплексного тестировани

Комплексное тестирование должно начинаться при выполнении следующих условий:

1. Система отслеживания ошибок установлена и доступна для использования группам поддержки разработок и системных разработок.
2. Группа системных операций и администрирования завершила настройку клиентских рабочих станций и серверов для тестирования. Группа тестирования получила необходимый доступ к этим системам.
3. Группа разработки подготовила по крайней мере два взаимодействующих компонента системы для передачи на комплексное тестирование под формальным автоматизированным контролем системы управления конфигурациями.
4. Группа разработки или группа сборки версий подготовила тестовую версию, содержащую по крайней мере два взаимодействующих компонента, прошедших компонентное тестирование.
5. Группа разработки подготовила перечень изменений, которые описывают функциональность тестовой версии.

Критерии продолжения комплексного тестирования

Комплексное тестирование должно быть продолжено при условии выполнения следующих критериев:

1. Каждая версия, предназначенная для комплексного тестирования, содержит только компоненты, находящиеся под формальным автоматизированным контролем системы управления конфигурациями.
2. Группа разработки или группа сборки версий подготовила тестовые версии из взаимодействующих компонентов, прошедших компонентное тестирование.
3. Группа разработки сопровождает каждую тестовую версию перечнем изменений, которые описывают функциональность версии.
4. Существует тестовая версия, собранная из относительно нового кода, которая может быть установлена в тестовой среде таким образом, что она будет стабильно функционировать. Более того, группа тестирования может проводить плановое и исследовательское тестирование тестовой версии в относительно эффективной манере, позволяющей получить в дальнейшем поддающиеся интерпретации результаты тестирования.

Критерии завершения комплексного тестирования

Комплексное тестирование считается успешно завершенным при выполнении Следующих условий:

1. Группа тестирования выполнила все запланированные тесты относительно всех запланированных версий для комплексного тестирования.
2. Заключительная версия, предназначенная для комплексного тестирования, содержит все компоненты, которые войдут в версию Суматры, предназначенную для пользователей (General Availability).
3. Группа управления проектом «Суматра» считает, что выполненное комплексное тестирование достаточно, и дальнейшее продолжение комплексного тестирования, по всей вероятности, не выявит новых дефектов, связанных с интеграцией компонентов.
4. Группа управления проектом «Суматра» провела совещание, посвященное завершению фазы комплексного тестирования, и приняла решение о том, что критерии завершения тестирования выполнены.

Критерии начала системного тестирования.

Системное тестирование может быть начато при выполнении следующих условий:

1. Система отслеживания ошибок установлена и доступна всем участникам проекта.
2. Группа системных операций и администрирования завершила настройку клиентских рабочих станций и серверов для системного тестирования. Группа тестирования получила необходимый доступ к этим системам.
3. Группа разработки завершила разработку всех функций и исправление всех ошибок, запланированных для инкремента 1, и поместила все указанные компоненты исходного кода под формальный автоматизированный контроль системы управления конфигурациями.
4. Группа разработки завершила компонентное тестирование всех функций и исправление всех дефектов, запланированные для инкремента.
5. Открыто менее 50 ошибок, подлежащих обязательному исправлению (согласно решению группы управления проектом), включая ошибки, обнаруженные в ходе компонентного и комплексного тестирования.
6. Группа управления проектом «Суматра» провела совещание по поводу начала фазы системного тестирования и приняла решение, что инкремент 1 готов к системному тестированию.
7. Группа сборки версий подготовила полную версию с контролируемыми изменениями для группы тестирования, согласно описанию в разделе «Версионное управление».

Рис. 6.5. Критерии начала, продолжения и завершения комплексного и системного тестирования

Критерии продолжения системного тестирования

Системное тестирование должно быть продолжено при условии выполнения следующих критериев:

1. Все программное обеспечение, передаваемое на тестирование, сопровождается документированным перечнем изменений. Перечень изменений должен содержать список отчетов о дефектах, которые, по мнению группы разработки, исправлены в каждой из версий.
2. В системе "Суматра" не производится никаких изменений, т.е. не меняются исходный код, файлы конфигурации, пользовательская документация и другие инструкции по настройке и процессы, без сопровождающего сообщения об ошибке или как элементы планируемых свойств или исправлений ошибок в следующей версии.
3. Группа сборки версий еженедельно предоставляет контролируемую полную версию программного обеспечения группе тестирования, созданную на основе относительно нового исходного кода, согласно описанию в разделе "Версионное управление". Эти версии могут быть установлены в тестовой среде таким образом, что они будут стабильно функционировать. Более того, группа тестирования может проводить плановое и исследовательское тестирование тестовой версии в относительно эффективной манере, позволяющей получить в дальнейшем поддающиеся интерпретации результаты тестирования.
4. Открыто менее 50 ошибок, подлежащих обязательному исправлению (согласно решению группы управления проекта), включая ошибки, обнаруженные в ходе компонентного и комплексного тестирования.
5. До завершения фазы системного тестирования дважды в неделю проводятся совещания, посвященные управлению незавершенными работами по устранению дефектов и временем закрытия дефектов.

Критерии завершения системного тестирования

Системное тестирование успешно завершено в случае выполнения следующих критериев:

1. В предшествующие четыре недели не произошло ни одного сбоя, остановки, неожиданного прекращения процесса или другой остановки обработки на программном и аппаратном обеспечении серверов.
2. Инкремент, являющийся кандидатом в золотую сборку (в настоящее время планируется, что это будет инкремент 5), подвергался системному тестированию в течение шести недель.
3. Группа тестирования выполнила все запланированные тесты относительно тестовой версии системы, являющейся кандидатом в золотую сборку.
4. Группа тестирования исправила все ошибки, подлежащие обязательному устранению.
5. Группа тестирования удостоверилась, что все проблемы в системе управления дефектами либо закрыты, либо отложены и, где необходимо, проверены с помощью регрессионного и проверочного тестирования.
6. Кривая качества продукта на инструментальной панели показывает, что Суматра достигла приемлемого уровня качества, стабильности и надежности.
7. Значение покрытия рисков качества на инструментальной панели показывает, что все риски качества покрыты адекватным образом.
8. Группа управления проектом согласна, что продукт, как это определено в ходе заключительного цикла системного тестирования, будет удовлетворять разумным ожиданиям качества пользователей и заказчиков.
9. Группа управления проектом провела совещание, посвященное завершению системного тестирования, и согласилась, что критерии завершения системного тестирования выполнены.

Рис. 6.5. (*продолжение*)

сматриваются в главе 15.) Перечень предлагаемых Джамалом критериев приводится на рис. 6.5. Он отправил сообщение с критериями и описанием инструментальной панели Джону и Гарольду с копией вице-президенту по продажам Сесиль Райс (Cecile Rice), Дженни Кауфман, Кейт Эрнандес, Кейту и Ясухиро.

В тот же день Джамал получил ответ от Гарольда, направленный также в качестве копии всем остальным получателям письма Джамала.

Джамал!

Проверил прикрепленные файлы. Инструментальная панель – отлично, но объясни мне, пожалуйста, разницу между графиками выполнения работ (Fulfillment) и хода работ (Progress).

Что касается критериев: они выглядят разумно. Хорошо, что ты понимаешь необходимость принятия решений относительно начала и завершения тестирования на уровне управления проектом. На предыдущем месте работы я имел дело с менеджером по контролю качества, использовавшим ряд измерений, которые вели к непродуктивным спорам о серьезности дефектов, о значении слова "является" и т.п. :-)

Два вопроса относительно прекращения тестирования.

Что это за волшебное число 50 открытых дефектов, при котором происходит остановка тестирования?

Как ты планируешь измерять эффективность, чтобы мы поняли, что такая эффективность подходящая?

Отличная работа, Г.

Джамал отправил ответ с копией всем получателям.

Привет, Гарольд!

Рад, что тебе понравилось. Если ты не против, поговорим об инструментальных панелях завтра?

Относительно решений уровня управления проектом. Да, критерии начала, продолжения и завершения фаз тестирования должны учитывать не только вопросы качества, но и варианты свойств, бюджета и сроков.

По поводу замечания, 50 не является волшебным числом, но совещания, посвященные расстановке приоритетов исправления дефектов, становятся неуправляемыми при наличии примерно такого числа открытых дефектов. Мой опыт показывает, что нет никакого смысла продолжать тестирование, если невозможно управлять основным результатом работы тестировщиков – отчетами о дефектах. Верно?

Что же до измерения эффективности, это часть того, что нам покажет график хода работ. Если нам потребуется для прогона тестов в три-четыре раза больше времени, чем мы планировали, что мы, безусловно, увидим на графике, думаю, мы захотим рассмотреть другие варианты дальнейшей работы. Снова подчеркну, это должно быть бизнес-решение о балансе качества, свойств, бюджета и сроков. Я бы лучше не прекращал тестирование, однако, если не будет достигнуто никакого продвижения вперед, по-моему, мы должны обсудить на уровне управления проектом, имеет ли смысл продолжать тратить наши ресурсы.

Надеюсь, я ответил на твои вопросы.

С уважением,

Джамал

Через 15 минут Джамал получил ответ на свое сообщение, с копией всем участникам дискуссии.

Джамал,

Все совершенно понятно. Пожалуйста, рассматривай критерии как "официально одобренные Гарольдом".

Пока,

Г.

№ этапа	Этап	Выполнено?
2.	Обсудить и документировать совместные работы подпроекта тестирования и объемлющего проекта разработки.	☑

"Отлично, — думал Джамал, — я уже почти закончил план, но что может пойти не так? Что я могу сделать для смягчения рисков, и если случится что-нибудь плохое, могу я запустить план на случай непредвиденных обстоятельств, чтобы снизить ущерб?" Он снова обратился к предыдущим проектам, просмотрел базы данных управления изменениями некоторых последних его проектов и добавил несколько рисков, а также план на случай непредвиденных обстоятельств и план смягчения рисков. Он некоторое время сидел, уставившись в стену, а затем добавил еще несколько рисков. Джамал пролистал несколько книг о тестировании, которые лежали на его столе, и затем дописал еще риски[1]. В таблице 6.1 показано, над чем размышлял Джамал.

№ этапа	Этап	Выполнено?
3.	Завершить подготовку и документирование оставшихся логических деталей и подробностей плана, в том числе рисков самого подпроекта тестирования и определений глоссария тестирования и проекта. Указать все цитируемые документы. Подготовить одностраничное резюме подпроекта тестирования.	☑

После составления списка рисков Джамал приступил к включению задач и напоминаний в программу для составления плана-графика работ, чтобы можно было выяснить, наступили ли события рисков. Вычисление местоположения этих мин замедленного действия — одна из лучших частей плана, думал он. Как бы было хорошо, если бы мне удалось не наступить на эти мины! Джамал завершил подготовку плана определением ряда терминов, перечислением ссылок на цитируемые и вспомогательные документы и написанием краткого резюме. Тем же вечером, 18 сентября, в среду, перед уходом домой Джамал отправил первый вариант плана тестирования в качестве прикрепленного файла Джону Албертсо-

[1] База данных управления изменениями рассмотрена в главе 7 моей книги "*Managing the Testing Process, Second Edition*". Кроме того, в главе 11 той же книги обсуждаются факторы, влияющие на ускорение и замедление тестирования. К последним относятся также риски, которые должны быть смягчены.

Таблица 6.1. Риски, непредвиденные обстоятельства и их смягчение для проекта SpeedyWriter

Риск	Непредвиденное обстоятельство/Смягчение
Сбой в процессах решения или эскалации проблем	Попытаться найти обходное решение проблемы, учесть неэффективный ход работ и задержки в выполнении планов. Или Остановить тестирование в соответствии критерием продолжения тестирования, решить процессуальную проблему. Или Продолжить тестирование обходным путем, разрешить проблему при следующем ее возникновении.
Внешние исполнители не могут установить Суматру.	Предотвратить возникновение проблемы за счет тщательного тестирования установки перед отправкой версии внешним исполнителям. Или Подготовить отчет о дефекте в процессе установки. И Отправить опытного специалиста для установки.
Серьезные изменения в требованиях, архитектуре, свойствах или других частях плана разработки, сделанные в последнюю минуту.	Учесть возрастание рисков качества из-за неполного тестирования изменения. Или Принять риск увеличения бюджета из-за добавления сотрудников или аутсорсинга для выполнения достаточного тестирования в последний момент. Или Принять риск увеличения сроков за счет откладывания дат выпуска.
После начала тестового цикла обнаружилось, что тестовая версия не готова к тестированию.	Ввести автоматизированное приемочное тестирование на уровне подготовки сборки для определения неготовых версий. Или Прекратить тестирование в ходе "неудавшегося" цикла, вернуться к "последней хорошей" версии и продолжить тестирование с учетом сокращения выполнения запланированных тестов и снижения темпов хода работ.
Подготовка тестовой среды не завершена к дате начала комплексного или системного тестирования.	Начать тестирование с использованием тех тестов, которые могут исполняться в доступной среде, учитывая ограниченные возможности выполнения тестов, невысокий темп хода работ и значительные пробелы в покрытии тестами. Или Отложить плановые сроки начала и завершения фазы тестирования в соответствии с реальными сроками готовности тестовой среды.

Таблица 6.1 *(продолжение)*

Риск	Непредвиденное обстоятельство/Смягчение
Тестовая среда не является полной для одной или двух фаз.	Переработать план с учетом необходимого продления фаз, исключить все тесты, зависящие от неготовых конфигураций, выявить возрастающие риски. Или Установить третью (ночную) смену. (NB! Это решение необходимо принять до найма младших тестировщиков. Оно не гарантирует полного смягчения рисков.)
Поддержка тестовой среды недоступна или не является квалифицированной.	Учесть снижение темпов хода работ и невыполнение запланированных тестов. Или Нанять на короткий период контрактника для получения квалифицированной и своевременной поддержки.
Передаваемые материалы, содержащие ошибки, препятствуют ходу тестирования, или снижают тестовое покрытие, или и то и другое вместе.	Предотвратить проблемы за счет тщательного компонентного тестирования силами группы разработки. Или Придерживаться критериев продолжения и остановить тестирование, если проблемы станут серьезными. Учесть задержку работ. Или Попытаться продолжить тестирование в соответствии с текущим графиком и бюджетом с учетом ухудшения качества версии для заказчика.
Пробелы в тестовом покрытии.	Исследовательское (партизанское – guerrilla) тестирование позволяет протестировать области, не покрытые запланированными тестами. И Использовать по возможности методы анализа структурного покрытия, анализ данных об отказах при эксплуатации, данные пользователей для определения пробелов, которые необходимо заполнить. Этот вопрос должен, по всей вероятности, решаться на основе доступности ресурсов в ходе проектирования и разработки тестов.
Неясность с вариантами использования системы, неполнота информации о средах применения системы или неверное тестирование.	Получить информацию о реальных вариантах использования системы от группы поддержки пользователей, о результатах альфа- и бета-тестирования, организуемого службой маркетинга, сведения от группы продаж, а затем разработать тесты для покрытия этих областей. Или Учесть в соответствии с ограничениями сроков и ресурсов возможность тестирования, несогласованного с вариантами использования системы.
Незавершенность разработки, невыполнение критериев готовности к назначенному времени.	Нарушить критерии готовности, сократить объем функциональности, предоставляемой для тестирования, и внести коррективы в план-график проекта. Или Нарушить критерии начала тестирования и учесть возрастание риска низкого качества из-за недостаточного времени на обнаружение и устранение ошибок, включая риск полного провала проекта.

Таблица 6.1 (*продолжение*)

Риск	Непредвиденное обстоятельство/Смягчение
Непредвиденная потребность в дополнительных ресурсах для завершения тестирования.	Увеличить человеческие ресурсы и бюджет. Или Исключить часть тестирования и перераспределить ресурсы, предназначавшиеся для тестирования пропущенных областей. (NB! Не во всех ситуациях это возможно.) Или Принять решение не выполнять тесты, для которых не выделено достаточно ресурсов, и учесть возрастание рисков качества, связанных с пробелами в тестовом покрытии.
Усталость тестировщиков.	Принять к сведению временное замедление выполнения запланированных тестов. И Заполнить как можно быстрее вакансии за счет тестировщиков-контрактников, имеющих подходящую квалификацию.
Усталость ключевых участников (вне группы тестирования).	Определить альтернативные варианты или дублеров для каждого ключевого участника, не являющегося сотрудником группы тестирования. Или Заполнить место сотрудником из группы тестирования, что влечет за собой неэффективность работы и более медленное выполнение запланированных тестов.
Невозможность нанять подходящего инженера по автоматизированному тестированию.	Использовать все доступные средства для поиска идеального кандидата, включая рекрутинговые компании. Или Увеличить продолжительность раунда тестирования, чтобы иметь возможность выполнять тесты более медленно. Или Отправить одного из тестировщиков на ускоренную переподготовку в учебный центр производителя инструмента тестирования[a].
Отладка в тестовой среде.	Предоставить конфигурации в соответствии с требованиями пользователей как для разработки, так и для тестирования, вести управление конфигурациями, для того чтобы можно было воспроизвести ошибки в средах разработки. Или Учесть замедление или остановку выполнения план тестирования наряду со снижением эффективности работ из-за необходимости восстановления тестовой среды после отладки.

a. Я благодарю одного из рецензентов, Дебору Маккандлесс (Deborah McCandless) из Nolo.com, за вклад в эту тему. Она написала: "[Для этого вам нужен] кто-нибудь, у кого есть все основные умения и навыки, а также желание пройти учебный курс и начать сразу давать отдачу. Это может быть отличной возможностью для кого-нибудь существенно продвинуться в карьере. Я проделала этот опыт с одним из младших программистов, и все получилось отлично".

ну, Гарольду Джоунзу, Дженни Кауфман, Кейт Эрнандес, Кейту Ли, Ясухиро Канагаве, Косимо Неграеву, Речел Вудз, Максу дель Оро, Джейму Иноджосе и Бобби Макдениэлсу. Он также послал копии Раджешу Гупте, Сайсли Райс и Тине Браун.

Привет всем!

К письму прикреплена первая версия плана комплексного и системного тестирования Суматры. Сопровождающие документы с описаниями графика, бюджета и результатов анализа рисков можно найти на сервере Jupiter в папке Sumatra/Test.

Понимаю, что все мы заняты, однако прошу прислать отзывы в пятницу до конца рабочего дня. Я учту ваши отзывы в плане за выходные и подготовлю новую версию к понедельнику. Вы все получите приглашение на совещание во вторник, чтобы изучить пересмотренную версию плана и принять решение. Будет подан обед. (Тина, спасибо за заказ совещания.)

С уважением,

Джамал

Довольный тем, что неплохая первая версия плана предложена для обсуждения вне группы тестирования, Джамал отправился домой. Он знал, что, хотя большая часть трудной для подготовки писанины завершена, еще предстоит сложная работа по согласованию и утверждению плана.

№ этапа	Этап	Выполнено?
4.	Разослать план для закрытого (автономного) рецензирования, обычно сначала сотрудникам группы тестирования, а затем на более широкое обсуждение среди заинтересованных лиц и участников проекта. Собрать отзывы и пересмотреть план, повторив этапы 1 — 3 по необходимости. Изучить любые изменения в имеющемся графике и бюджете (превышающие оговоренные резервы времени и планы на случай непредвиденных обстоятельств), внесенные в результате процесса планирования, и получить поддержку этих изменений у руководства.	☑

За рамками процесса: основные соображения по поводу плана тестирования

Большая часть того, что попадает в план тестирования, — либо результаты анализа рисков качества или оценки затрат, либо итог обсуждений с ключевыми участниками проекта. В первом случае все необходимое мы получаем за счет тщательного отбора информации из таблицы анализа видов ошибок и их влияния, графика и бюджета и перевода ее в текст или картинки. Во втором случае ключевые участники проекта свободны в выборе ролей и обязанностей, по крайней мере, в рамках ограничений проекта. В обоих случаях план тестирования просто собирает и переформулирует существующую информацию и дает возможность обмениваться этой информацией.

Стратегия тестирования (Test Strategy)

Совокупность выбранных подходов и решений, вытекающих из целей и задач подпроекта тестирования и группы тестирования. Целью обычно является эффективное и качественное тестирование, а стратегия — это общие правила и принципы, способствующие достижению цели. Тактика тестирования — это частные правила, методики, процессы и способы проведения тестирования.

Однако тестировщик или тест-менеджер, готовящий план, должен тщательно обдумывать некоторые темы. Он должен предлагать идеи, выводы и решения по этим направлениям таким образом, чтобы люди, прочитавшие план, очень четко понимали, что от них требуется. Две такие темы, управление подготовкой тестовых версий и отчетность о состоянии дел в тестировании, будут рассмотрены в последующих главах в качестве самостоятельных ключевых процессов. (Важно помнить, что эти проблемы должны быть учтены в планах.) Сейчас я предлагаю рассмотреть четыре темы, не относящиеся к ключевым процессам: стратегию тестирования, планирование ресурсов тестирования, управление конфигурациями и управление рисками, которые обязательно нужно отразить в планах.

Выбор стратегий тестирования

Начнем со стратегий тестирования. Что такое стратегия тестирования? Разные люди по-разному понимают это выражение. Мое определение приведено выше. Для придания смысла контексту стратегии тестирования и ее взаимосвязи с целями и тактикой группы тестирования и проекта в целом предлагаю обратиться к рис. 6.6.

На рис. 6.7 показана стратегия тестирования проекта "Суматра". Эта стратегия, без сомнения, универсальная. Это довольно обычная вещь, по крайней мере, как один из экземпляров из определенного семейства стратегий тестирования. Данное семейство стратегий тестирования поддерживает следующий общий подход.

- Разработать множество тестов на основе проектирования тестов "сверху вниз", используя анализ рисков, требования, цели тестов и т.д.
- Укрепить это множество тестов за счет проектирования "снизу вверх" на основе реального использования системы, сценариев использования системы, данных пользователей и т.д.
- Использовать исследовательское тестирование для заполнения очевидных пробелов в покрытии тестами.
- Применить какой-либо вариант структурного анализа (динамический или статический, например оценку сложности кода) для определения скрытых дефектов в тестовом покрытии.

Рис. 6.6. Цель, стратегия и тактика группы тестирования

Стратегии тестирования, наряду с этими направлениями, были изучены еще в ранних работах по тестированию, в том числе в книге "*Искусство тестирования программ*" Майерса (Myers) и "*Complete Guide to Software Testing*" Гетцеля (Hetzel). Они остаются востребованными и сегодня, хотя некоторые варианты не содержат ряда заключительных этапов (т.е. выполняющихся после нисходящего анализа).

Однако есть противники этого подхода. Канер (Kaner), Бах (Bach) и Петтикорд (Pettichord) в "*Lessons Learned in Software Testing*" защищают подход, основанный на исследовательском тестировании, управляемом рисками. Некоторые сторонники так называемых "*быстрых методологий*", например экстремального программирования, говорят о необходимости "тестировать все, что может сломаться" под управлением программистов. Поскольку эти утверждения определяют направления, а не принципы, они указывают на необходимость тестирования на основе известных технологических слабостей. Я рассматриваю такой подход как стратегию тестирования на базе архитектуры.

Другая совокупность решений относительно стратегии относится к тому, как и какие тесты группа тестирования должна автоматизировать. Дастин (Dustin), Рашка (Raschka) и Пол (Paul) в книге "*Автоматизированное тестирование программного обеспечения*" продвигают формализованный подход к принятию этих стратегических решений. Они называют его методологией жизненного цикла автоматизированного тестирования. Менее формализованный подход принятия тех же решений можно найти у Фьюстера (Fewster) и Грэма (Graham) в "*Software Test Automation*".

Стратегия тестирования проекта "Суматра"

- Разработать автоматизированные и ручные тесты для покрытия всех рисков качества, которые определены как нуждающиеся в полном или сбалансированном тестировании, с особым вниманием к факторам поведения системы, наблюдаемого со стороны некоторого пользовательского интерфейса или доступного программного интерфейса приложения (API).
- Добавить тестовые данные или условия в рамках существующих или новых тестовых сценариев для покрытия критических вариантов использования системы, пользовательских данных или известных дефектов в нашем продукте или продуктах конкурентов.
- Использовать исследовательское тестирование в областях, которые ранее не были покрыты тестами, и которые, с точки зрения интуиции или результатов тестирования, представляются областями повышенного риска наличия дефектов. Обновить старые или определить новые тестовые сценарии для покрытия обнаруженных дефектов.
- Провести тестирование на конфигурациях (в тестовой среде), представляющих собой комбинацию серверов, сетей и клиентских машин, похожих на эксплуатационные конфигурации.
- Повторить несколько раз все тестовые сценарии для комплексного и системного тестирования, включая функциональные тесты для SpeedyWriter 3.0, в ходе каждой фазы для поиска регрессионной зависимости.
- Насколько позволяют время и ресурсы или предписывает представление о не выявленных рисках качества, использовать методы структурного покрытия для определения не подвергавшихся тестированию областей и затем дописать тесты для покрытия критических пробелов в тестовом покрытии.

Рис. 6.7. Высокоуровневая стратегия тестирования проекта "Суматра"

Еще одна часть стратегии — это планируемые к использованию методы тестирования.

- *Статические.* Тестирование посредством изучения элементов системы, например спецификаций требований, архитектуры и исходного кода. Два известных примера: инспекция кода и рецензирование требований. Эта тема подробно обсуждается в "*The Psychology of Computer Programming*" Уайнберга (Weinberg), "*Software Project Survival Guide*" Макконнела (McConnell), "*Software Requirements*" Вигерса (Wiegers) и во многих других работах.
- *Структурные (иначе, методы белого или стеклянного ящика).* Тестирование посредством изучения способа реализации системы. Три общепринятых метода: тестирование потоков данных, тестирование путей и тестирование транзакций. Большое число структурных методов тестирования приводит Бейзер (Beizer) в "*Software Testing Techniques*".
- *Поведенческие (иначе, методы черного ящика).* Тестирование за счет изучения свойств системы и поведения, которое она должна демонстрировать. Классические примеры — это эквивалентные разбиения и тестирование областей (domain testing). Эти темы подробно разбираются в работах Бейзера (Beizer) "*Black-Box Testing*" и "*Software*

System Testing and Quality Assurance", а также у Канера в работе "Тестирование программного обеспечения".

- *Опытная эксплуатация.* Тестирование на основе того, что пользователи реально делают с системой. Бета-тестирование, изучение удобства использования и, в некоторых случаях, приемо-сдаточные испытания — общепринятые методы. Канер (Kaner) и др. рассматривают опытную эксплуатацию в "*Testing Computer Software*", а Нильсен (Nielsen) подробно исследует удобство использования в "*Usability Engineering*".

Эти методы не отличаются резко друг от друга, как вкус сладкого, кислого, соленого и горького, а, скорее, непрерывно переходят друг в друга, аналогично холодному, теплому и жаркому. Например, тесты на основе архитектуры, в частности тесты производительности, являются в некоторой степени поведенческими, поскольку нам известно, насколько быстро система должна реагировать на определенные ситуации, а также немного структурными, так как тестовые условия учитывают известные ограничения в технологиях. Из-за их непрерывной природы я предпочитаю рассуждать об этих категориях в терминах спектра степеней точности определения тестов[1].

Вообще говоря, верно, но не обязательно, что первоочередное тестирование в проекте обычно статическое, затем следует структурное, потом поведенческое, и наконец живое. До некоторой степени это случайность, когда можно начинать определенные виды тестирования, хотя также верно, что каждый вид тестирования обычно позволяет найти дефекты определенных типов, которые труднее обнаружить впоследствии при применении более грубо определенных тестов.

Последующие тесты часто позволяют обнаружить ошибки, которые с помощью ранее применяемых тестов найти было нельзя. Два примера. Ошибки в условиях, такие как использование сравнения "больше" вместо "больше либо равно", наилучшим образом обнаруживаются в ходе модульного тестирования, проблемы сквозной работы (end-to-

Степень точности тестирования (Test Granularity)

Точность нацеливания теста. Точные (fine-grained) тесты позволяют тестировщику проверить низкоуровневые компоненты, особенно реализации; статические и структурные тесты являются точными. Грубые (coarse-grained) тесты предоставляют тестировщику информацию об общей работе системы и требованиях; поведенческие тесты и тесты для опытной эксплуатации являются грубыми.

[1] Енг Нгуен (Hung Nguyen) в книге "*Testing Applications on the Web*" говорит о тестировании методами "серого ящика", под которыми он понимает тестирование поведения системы с использованием определенных знаний о ключевых технических рисках и архитектурных решениях для более быстрого и качественного поиска дефектов.

end performance) приложения могут быть обнаружены только при системном тестировании.

На рис. 6.8 показана V-модель, рассмотренная в главе 1, снабженная степенями точности тестирования, обычно сопровождающими каждый уровень. Когда проект следует каскадной модели, все четыре вида тестирования могут применяться на протяжении большей части проекта.

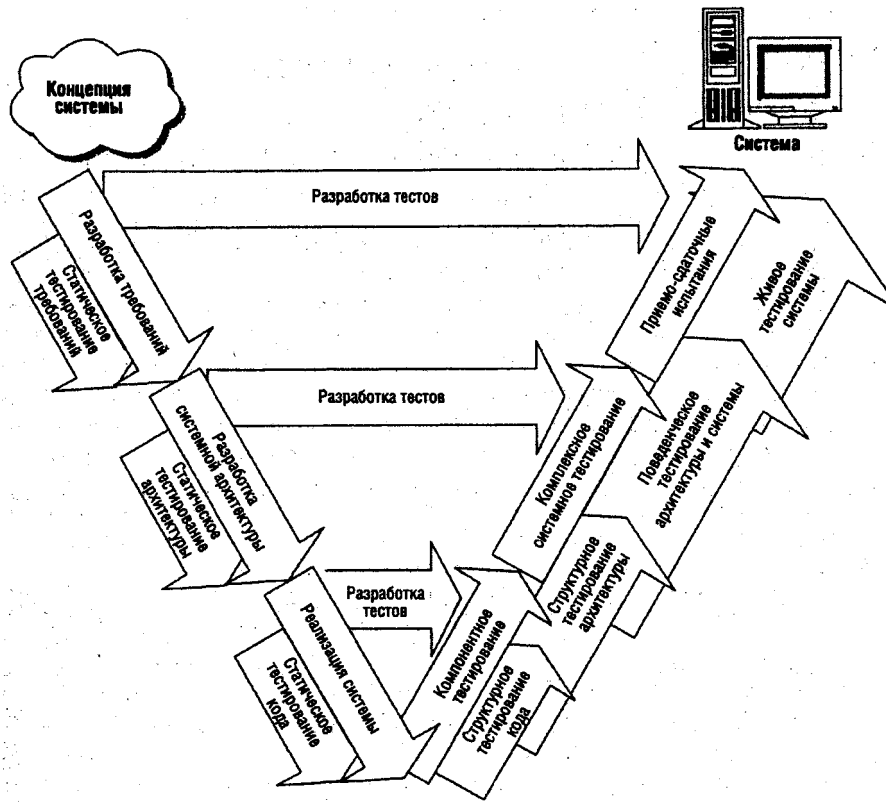


Рис. 6.8. V-модель, дополненная детализацией точности тестирования

Позвольте завершить этот подраздел двумя замечаниями. Во-первых, поскольку для изучения этой темы доступно много хороших книг, выбор верной стратегии для проекта тестирования является больше искусством, чем наукой. Планируйте со временем улучшить свои навыки. Если вы впервые готовите стратегию тестирования, поищите специалистов, консультантов и коллег, обладающих опытом. Они помогут вам.

Основанием для этого совета является следующее наблюдение. Некоторые опытные тестировщики и тест-менеджеры успешно проводят тес-

тирование, не составляя письменный план тестирования. В некоторых случаях эти специалисты не могут даже объяснить, почему они выполняют тот или иной тест. Тем не менее эти практики могут отменно себя проявить при работе с проектами, в которых не особенно внимательно относятся к подготовке документации. Сильная стратегия тестирования сама по себе может компенсировать большое число других недоработок, в том числе и в области планирования тестирования.

Особый случай: стратегии управления рисками регрессионной зависимости

Вернемся к предпоследнему пункту на рис. 6.6. Джамал предлагает повторить ранее выполненные тесты, для того чтобы обнаружить регрессионную зависимость, или регрессию. Вообще говоря, риск регрессионной зависимости существует всегда, если в систему вносятся изменения хотя бы в одной строчке кода[1]. Этот риск возникает не только при изменении кода, но и в случае изменения аппаратной части, встроенных программ, конфигурации, баз данных, сосуществующего программного обеспечения, сетей и т.п. В одном проекте по разработке Интернет-приложения основной источник риска регрессионной зависимости состоял в неконтролируемых изменениях, которые производил без уведомления производитель встроенных программ для интегрированного модема.

Риск регрессионной зависимости имеет несколько особенностей. Один из способов рассмотрения регрессионной зависимости состоит в том, чтобы работать с ней после ее обнаружения.

- Программист или инженер производит изменения в системе, приводящие к регрессионной зависимости, которая не обнаруживается при тестировании, пока не будут внесены в систему другие изменения. Чем больше таких изменений *внесено* в систему, тем труднее изолировать причины регрессионной зависимости до вызвавшего проблему изменения, особенно если последующие изменения маскируют регрессионную зависимость или оказывают влияние на ее поведение.
- Программист или инженер производит изменения в системе, приводящие к регрессионной зависимости, которая не обнаруживается при тестировании ни при каких условиях, но затем ее находят пользователи или заказчик. Как уже отмечалось, чем больше изменений внесено в систему до обнаружения пользователем регрессионной зависимости, тем труднее изолировать ее причину.

Первый вид регрессионной зависимости приводит к нежелательным трудностям и задержкам в процессе отладки, что также ведет к задержкам в тестировании. Второй вид регрессионной зависимости отрицательно

---

[1] Исключительный пример: это опечатка ("6" вместо "D")) в программе для телефона SS-7, которая привела к выходу из строя большого числа аппаратов (больше семи миллионов) в различных районах Соединенных Штатов.

Регрессия, регрессионная зависимость (Regression)

Система обладает свойством регрессии, если в результате изменения новая версия системы S_{n+1} содержит ошибку в ранее корректно работавшей функции, т.е. этой ошибки не было в предыдущих версиях, начиная с S_1 и заканчивая S_n.

Регрессионное тестирование (Regression Testing)

Тестирование, предназначенное для поиска дефектов, вызывающих регрессионную зависимость.

влияет на впечатление пользователя от системы. Если какой-либо вид регрессионной зависимости возникает в областях, которые не подвергались тестированию из-за намеренных компромиссов между рисками качества, сроками, бюджетом и свойствами системы, то такие регрессионные зависимости, как это ни печально, находятся вне пределов контроля группы тестирования.

Однако если регрессионная зависимость проявляется в областях, проверяемых при тестировании, это происходит в результате пропусков в регрессионном тестировании. Для второго вида риска регрессионной зависимости: если считать, что она имеет место в свойстве, протестированном до вызвавшего ее изменения, а не после, то ошибка регрессии, обнаруженная заказчиком или пользователем, также избежала тестирования.

(В качестве отступления вспомним обсуждение экономики тестирования в главе 4. Первый вид регрессионной зависимости увеличивает затраты на внутренние несоответствия, что сокращает объем возврата вложений в тестирование. Второй вид регрессионной зависимости представляет затраты, понесенные от внешних несоответствий. Эти несоответствия относятся и к группе тестирования, пропустившей регрессионную зависимость, и к программисту, который внес ухудшающее работу системы изменение.)

Границы регрессионного тестирования (Regression Test Gap)

Это разность между тестовым покрытием всего комплекта тестов и тестовым покрытием, которое достигается с помощью реально выполненных тестов для данного изменения или данной версии. Для версии, предоставляемой заказчику, границы регрессионного тестирования могут рассматриваться как область, которая в определенной степени не подвергалась проверке в ходе тестирования данной версии с применением всего комплекта тестов.

Другой способ рассмотрения регрессионной зависимости охватывает весь жизненный цикл системы: от первоначальной концепции до списывания системы. Предположим, что мы работаем над версией системы X.Y, а ранее мы выпустили версии 1.0, 1.1,..., 2.0, 2.1,..., X.0, X.1,..., X.(Y-1). Могут иметь место два вида регрессионной зависимости:

- Программист или инженер производит изменения в системе, приводящие к регрессионной зависимости возможностей или свойств системы, которые являются новыми для версии X.Y.
- Программист или инженер производит изменения в системе, приводящие к регрессионной зависимости возможностей или свойств системы, которые появились в одной из версий, предшествующих X.Y.

Первый риск является менее проблемным. Поскольку мы говорим о новом свойстве, оно, вероятно, представляет какую-то потребность некоторого подмножества пользователей, но, по всей видимости, не все пользователи в ней нуждаются, что означает уменьшение бизнес-риска. Более того, поскольку это свойство или возможность системы заказчик еще никогда не видел в продукте, бизнес-риск дополнительно снижается за счет того, что для этой возможности не было выработано никаких устоявшихся последовательностей действий, которые должны быть пересмотрены. (Хотя ранее объявленные возможности и свойства, недоступные пользователю, могли бы появиться в рамках *запланированных* последовательностей действий.) Наконец, поскольку разработка и тестирование версий сопровождения обычно сводится к работе над новыми свойствами и возможностями, технический риск таких регрессионных зависимостей и пропусков в регрессионном тестировании будет ниже.

И наоборот, бизнес-риски и технические риски будут выше в следующих трех случаях.

1. Вы часто располагаете ограниченной информацией о том, какое подмножество пользователей применяет определенное свойство или возможность системы.
2. Любое свойство или возможность системы может легко стать неотъемлемой частью последовательностей действий пользователей, причем в значительной мере независимо от того, какое подмножество пользователей действительно применяют это свойство или возможность системы.
3. Поскольку и программисты, и тестировщики фокусируются на каких-то других вопросах, можно с большой вероятностью пропустить регрессионную зависимость.

Пропуск при тестировании (Test Escape)

Любая ошибка, которая могла быть обнаружена при тестировании, но была пропущена. Термин часто применяют к ошибкам, обнаруженным на фазах, следующих за тестированием, хотя они должны были быть найдены на предыдущей фазе.

В таблице 6.2 собраны факторы, связанные с моментом обнаружения регрессии и с жизненным циклом системы, и соответствующие риски.

Таблица 6.2. Различные сценарии проявления регрессионной зависимости и вероятное развитие событий

	Обнаружены при тестировании	Обнаружены пользователем
Новое свойство или новая возможность	Увеличение затрат на достижение желаемого качества и задержка выпуска версии системы.	Неспособность поставить версию с заявленным составом свойств. Затраты на исправление ошибок эксплуатации. Возможный ущерб для имиджа компании.
Существующая возможность или существующее свойство	Значительные затраты на предсказание и обнаружение регрессионных зависимостей. Возрастающие затраты и задержки выпуска продукта с желаемым качеством. Возможные трудности при исправлении ошибок из-за текучести кадров.	Нарушение работы частей системы, ряд из которых является неотъемлемой частью последовательности действий пользователей. Затраты на исправление дефекта эксплуатации, обычно в виде дорогого аварийного патча. Серьезный ущерб имиджу компании. Юридическая ответственность.

Неспособность подготовить и выполнить эффективную стратегию тестирования может завести работу компании в порочный круг в рамках четырех клеток таблицы. Ранее я работал в компании, где некоторое время мы добавляли очень много новых свойств в каждую версию сопровождения при недостаточном времени на тестирование и отладку. В этот период, до постановки под контроль процесса выпуска версий сопровождения, президент компании получил письмо от одного из заказчиков, в котором была такая строчка: "Я никогда не имел дел с компанией, которая так мало заботится о качестве". Не самая приятная ситуация, особенно для тестировщика. Так это или нет, но часть обвинений в пропусках, допущенных при регрессионном тестировании, покончит с доверием группе тестирования.

Самая известная стратегия снижения риска регрессионной зависимости состоит в повторении тестов. Существуют разнообразные подходы к выбору нужных тестов и частоты, с которой эти тесты необходимо выполнять повторно. Поскольку тема выходит за рамки этой книги, я приведу несколько ссылок на другие источники.

Поскольку автоматизированное тестирование целиком состоит из подготовки и настройки повторяемых тестов, любая из книг на эту тему должна описывать способы выбора тестов для автоматизации, а также предлагать решение проблемы оптимизации возможностей управления риском регрессионной зависимости с помощью автоматизированных тестов. Эту информацию можно найти у Фьюстера и Грэма в "*Software Test Automation*" и у Дастина и др. в "*Автоматизированном тестировании программного обеспечения*".

Стратегии повторения тестов при ручном тестировании, а также некоторые ограничения автоматизированного регрессионного тестирования рассматриваются в главе 3 моей книги "*Managing the Testing Process*".

Если вы предпочитаете стратегии частичного, а не полного регрессионного тестирования при ручном выполнении тестов, можно воспользоваться методами анализа рисков качества, рассмотренными в главе 2 настоящей книги. Для отбора повторяемых тестов можно также использовать тренды дефектов, статус тестовых сценариев и общие результаты тестирования. Эти темы будут рассмотрены позднее.

Мне известны еще два варианта, позволяющих снизить эти риски без проведения полного или частичного перетестирования. Первый вариант — это аналитический метод. У одного из моих клиентов полное множество тестовых сценариев для комплексного и системного тестирования требовало 20 человеко-лет для однократного выполнения, однако он не располагал ни временем, ни бюджетом, позволяющим повторить и 5% всех тестов. (Эти 5% покрывали лишь основные функции, но не невероятные сложные последовательности действий.) Попытка покрыть все эти различные сценарии была бы бесполезной.

Он использовал измерения покрытия кода для того, чтобы не остались непроверенными большие разделы основной части исходного кода. Ему удалось добиться в среднем 75%-го покрытия всей функциональности при четырехнедельном цикле регрессионного тестирования, что, как показал опыт, существенно снижало риск регрессионной зависимости до приемлемого уровня. Существуют и другие аналитические подходы, включающие в себя анализ сделанных изменений и их влияния на систему, но я слышал от специалистов, которые пытались применять такие методы, что они крайне непрактичны, их следует применять к тривиальным изменениям.

Вторым направлением является эмпирический подход — широкое применение методов опытной эксплуатации. Большая программа бета-тестирования, например, при правильном выборе пользователей, дает хорошее пересечение различных сценариев использования системы и типов данных, которые имеют существенное значение для пользователей. (Уровень качества системы в этом случае должен подходить в первую очередь для бета-тестирования, иначе вы рискуете охладить желание настоящих и потенциальных пользователей участвовать в бета-тестировании.) Например, проблемы регрессионной зависимости, связанные с встроенными программами модема, о которых говорилось выше, были обнаружены в ходе бета-тестирования.

Опытная эксплуатация не исключает вероятности того, что часть ошибок будет пропущена. С одной стороны, в отличие от стратегии повторения тестов с известным покрытием, вы не знаете наверняка, где могут быть эти ошибки. С другой стороны, если эти ошибки не являются существенными для ваших пользователей или заказчика, вероятно, не стоит тратить много времени на их поиск. Конечно, применяя подход к тестированию на основе рисков, описанный в этой книге, я бы не стал разрабатывать тестовые сценарии для покрытия незаметных, несущественных рисков, а значение рисков в большой степени определяется тем, насколько они беспокоят пользователей.

В заключение позвольте заметить, что некоторые продвигают объектно-ориентированное программирование как средство ограничения риска регрессионной зависимости и, следовательно, потребности в регрес-

сионном тестировании. Однако в этом направлении была проделана определенная работа, которая показывает, что даже инкапсуляция и наследование при использовании компонентов (методов, объектов и т.п.) иначе, чем ранее применяемыми методами, не устраняют риска регрессионной зависимости. Поскольку невозможно с полной определенностью узнать, что изменение в одной части программы не повлияет на какую-либо другую ее часть (хотя и сильно удаленную в терминах экземпляров объекта или потоков данных), риск регрессионной зависимости по-прежнему существует[1]. По тем же причинам интеграция коммерческих программ или любых программ, разработанных третьей стороной, всегда страдает от тех же рисков регрессионной зависимости.

Определение ресурсов среды для проведения тестирования

Наряду с выполнением полного набора тестов вопрос о том, сколько раз выполнять каждый тест, оказывает существенное влияние на потребности в ресурсах тестирования. Позвольте пока оставить за рамками рассмотрения вопрос о персонале, исследуемый более подробно в главе 8. Сейчас обратимся к вопросу ресурсов тестовой среды, которые необходимы для выполнения тестов.

В элементарном случае выполнения тестов на автономной платформе, скажем, для программы офисной автоматизации на компьютере Macintosh, нужно располагать количеством систем, равным числу одновременно выполняемых тестовых сценариев. Допустим, у вас есть 120 часов в неделю на тестирование. Эмпирическое правило 30 часов в неделю на тестировщика дает минимальную потребность в четырех компьютерах Macintosh в тестовой лаборатории. Однако я предпочитаю иметь по два пользовательских терминала любого вида на тестировщика, поскольку требуется проводить локализацию проблем, зависящих от конфигурации или системы.

Предположим, что число четыре – это предельное число доступных рабочих станций. Если мы хотим, чтобы у каждого тестировщика было по два компьютера Macintosh, мы можем перейти на двухсменную работу, как это сделал Джамал в главном примере книги. При тестировании систем на редкой аппаратной базе, например на конструкторских прототипах или огромных корпоративных универсальных машинах (mainframe), для полной загрузки дорогого оборудования двухсменная работа часто просто необходима.

Если посмотреть на таблицу "Отслеживание выполнения тестов SpeedyWriter 3.1" ("SW 3.1 Test Tracking")), а именно на планы выполнения тестовых сценариев для первой и второй смены (Test Case Summary Plan (1Shift) и Test Case Summary Plan (2Shift)), можно увидеть, что используются довольно простые расчеты на основе продолжительности рабочего времени и тестирования, чтобы определить число серверных кластеров, необходимых для тестирования[2]. Тесты, выполняемые сто-

---

[1] Более подробно см. статью Дивейн Перри (Dewayne Perry) и Гейл Кайзер (Gail Keizer) "*Object-Oriented Programs and Testing*" на citeseer.nj.nec.com/perry90objectoriented.html.

[2] Эти и другие документы можно найти в разделе "Publications" на www.rexblack-consulting.com.

Искажение тестирования (Test Artifacts)

Поведение, которое возникает из-за искусственной природы тестовой среды или из-за того, что процесс тестирования отклоняется от потенциальных действий системы в среде эксплуатации; неверное поведение или некорректные результаты, выдаваемые системой тестов.

ронними компаниями, конечно, не берутся в расчет. Тесты, требующие использования серверного кластера целиком, не учитываются, поскольку, как минимум, один серверный кластер выделен для выполнения некоторых других множеств тестов. Два теста, которые не требуют использования кластера целиком, могут разделять его ресурсы, но тест, не требующий ресурсов кластера целиком, должен выполняться на какомлибо сервере (поскольку каждый тест нуждается в сервере), либо используя его целиком, либо разделяя его с другими. (Если вы все еще не поняли идею, предположите, что вы моете руки. Вы можете мыть левую руку больше, чем правую, но без правой руки вы не вымоете левую.) Формулы в этих таблицах не имеют 100%-ной точности, но предоставляют осторожную, а не чрезмерную оценку потребностей в аппаратном обеспечении.

В нашем примере оборудование для среды тестирования SpeedyWriter довольно простое, однако некоторые тестовые конфигурации могут включать в себя десятки серверов, соединенных с сотнями рабочих станций в различных местах с разнообразными видами соединений, как сетевых, так и коммутируемых. Для рассмотрения таких конфигураций обратитесь к главам 6 и 7 моей книги "*Managing the Testing Process, Second Edition*", где приводится база данных для планирования и управления сложными тестовыми конфигурациями.

При тестировании сетевых систем, систем клиент-сервер, систем на основе браузеров, систем для универсальных машин и им подобных может, а в некоторых случаях и должно иметь место разделение определенных аппаратных ресурсов для одновременного выполнения тестов. Например, в одном из проектов мы применяли генераторы загрузки для семейства серверов, чтобы построить модель пользовательской базы в 50 000 записей, одновременно проводя ручное тестирование с применением этих устройств, чтобы убедиться, что мнение пользователя о системе, особенно о ее производительности и надежности, не ухудшится.

Тем не менее нужно все делать осторожно. Например, обычно тестирование производительности требует использования выделенной изолированной тестовой среды, чтобы не произошло незаметного искажения результатов тестирования. Тестирование обработки ошибок и восстановления после сбоев, нагрузочное тестирование и тестирование перегрузок могут породить проблемы среды, которые будут проявляться при выполнении в системе любого другого теста. Даже если такие проблемы можно предвидеть, влияние медленной реакции и зависаний системы на произ-

водительность тестировщика и, следовательно, на ход тестирования должно быть сопоставлено со стоимостью дополнительного оборудования, с использованием нескольких рабочих смен, с выполнением наиболее тяжелых тестов в ночное время и др.

Другая сложность возникает, когда версии сопровождения, аварийные патчи и другие линии продуктов должны проверяться силами той же группы тестирования в той же тестовой лаборатории. В этом случае либо вы вынуждены иметь достаточно оборудования для раздельной поддержки всех необходимых конфигураций, *либо* нужно убедить руководство расставить приоритеты, если два или более проекта конкурируют за одни и те же ресурсы оборудования. План — это подходящее место для описания компромисса.

Универсальные машины, серверы, сети хранения данных и другие подобные объекты очень дороги, и обычно требуются разумные способы их использования. Однако для персональных рабочих станций, по моему глубокому убеждению, аргументы экономии обычно не выдерживают никакой критики.

Например, тестировщики рассказывали об использовании программ виртуальных машин для того, чтобы запускать для тестирования две операционные системы на одной рабочей станции одновременно. Ошибки, обнаруженные в такой конфигурации, вызывают сомнение. (Такие ошибки будут не только сомнительны, но и абсолютно ненадежны, если речь идет о производительности, надежности и других областях, в которых основным фактором является использование ресурсов.) Как узнать, что некоторое сложное взаимодействие между операционными системами, программой виртуальных машин, системой, предназначенной для тестирования, и другими сосуществующими программами, драйверами и данными, явилось причиной сбоя? По этой причине ошибки, которые были обнаружены в таких искусственных тестовых конфигурациях, вероятно, найдут недоброжелательный отклик со стороны программистов, которым поручат их исправление.

В качестве более приемлемого варианта рассмотрим мультизагрузочный диск или несколько дисков, которые можно выбрать во время загрузки компьютера. Конечно, нужна перезагрузка, но, по крайней мере, отсутствуют параллельно выполняемые задачи, которые могут использовать память и процессор компьютера. Однако, поскольку рабочая станция в среднем стоит дешевле, чем человеко-неделя инженера-тестировщика, а потеря недели обычно дороже, чем персональный компьютер, сколько вы готовы терпеть неэффективное использование рабочего времени, если избегаете приобретения нового оборудования или введения второй смены?

Управление системами тестов и конфигурациями тестовой среды

Какие бы ресурсы вы ни развернули в тестовой среде, вам потребуется некоторым образом управлять этой средой. Доступные вам политика и инструменты зависят в немалой степени от технологий, с которыми приходится работать. Для платформ с дисками IDE или SCSI часть политики

управления конфигурацией среды может включать в себя создание и хранение эталонных образов дисков, т.е. их побитовых копий для известных подходящих конфигураций, которые могут быть при необходимости восстановлены в тестовой среде. Для более крупных семейств серверов и универсальных машин существуют коммерческие средства управления конфигурациями, которые могут помочь вам в этом.

Ключевым моментом здесь является знание о том, как создать конфигурации, похожие на конфигурации пользователей, и о том, что эти конфигурации можно восстановить с минимальными усилиями. При тестировании возникают всякие неожиданности, иногда приводящие к разрушению конфигурации тестовой среды. Никому не нравится заниматься переустановкой недокументированной конфигурации вручную в середине тестового цикла, особенно с учетом закона Мэрфи, который гласит, что это событие случится в ходе последнего тестового цикла за три дня до срока сдачи версии или поставки системы!

Неважно, насколько легко восстановить конфигурацию, вы будете некомфортно себя чувствовать в случае, если она не восстанавливается с первого раза. Не всегда удается сразу распознать испорченную тестовую конфигурацию, и это означает, что тестировщик может выпустить массу ложных отчетов о дефектах, пока не поймет, что происходит. Также необходимо убедиться, что тестирование само не нарушает тестовую среду. В главе 10 будут рассмотрены процедуры настройки и демонтажа тестовой среды, которые являются частью всей мозаики.

Также нужно убедиться в том, что программисты и другие специалисты не портят тестовую среду. По этой причине политика управления тестовой средой должна включать в себя эффективный контроль прав и паролей доступа. Обычно в моей политике два железных правила. Первое: никто не должен вносить никаких изменений в тестовую среду без утверждения их тест-менеджером или, по крайней мере, старшим инженером-тестировщиком. (В этот список обязательно входят группы сетевой поддержки и администрирования. Внесение плановых обновлений и операции сопровождения тестовой среды должны быть отложены на время тестового цикла, а то и на весь период выполнения тестов из-за риска получения бессмысленных результатов тестирования.) Второе: никто не имеет права заниматься отладкой или разрешать ее проведение в тестовой среде, за исключением времени, официально определенного группой управления проектом, когда прекращено любое тестирование, на которое может повлиять отладка, и после завершения отладки среда восстанавливается до известного состояния.

Сотрудники, не входящие в группу тестирования, в том числе системные администраторы, персонал сетевой поддержки и т.п., часто берут под свой контроль управление сложной тестовой средой. Это может породить проблемы, влияющие на ход работ по тестированию. Вам может надоесть ждать, когда человек придет и внесет исправления в среду, чтобы можно было возобновить тестирование, но у него до вас еще было два-три звонка, над ответами на которые он работает. Запустите процесс эскалации проблемы для ее разрешения, чтобы не оказаться крайним в случае, когда целый день не начинается тестирование, поскольку представители поддержки не появляются. Как уже отмечалось в главе 5, особенно

осторожным надо быть в том случае, если вы полагаетесь на получение поддержки от незаинтересованных лиц проекта.

Необходимо письменно зафиксировать соглашения по поводу поддержки, а также процессы разрешения и эскалации проблем. Чем вероятнее, по вашему мнению, проблемы, тем более формализованными должны быть соглашения. Более того, процесс эскалации проблем должен поднимать проблему на уровень общего для тестировщиков и поддержки руководителя, либо, по крайней мере, на уровень некоторой группы управления, которая может форсировать сотрудничество. Кроме того, сотрудники низшего звена группы тестирования должны чувствовать себя абсолютно уверенно при эскалации проблем. В моей практике были случаи, когда чувство тщетности обращений наверх приводило непосредственных исполнителей группы тестирования к решению самостоятельно попытаться решить проблемы или найти обходной путь. Это скрытые снижение производительности группы и замедление хода работ по тестированию. Должно быть всем очевидно, что такие неконтролируемые изменения сфер ответственности недопустимы и являются неприемлемыми.

Я понимаю, что это звучит цинично, но такое случается. Не вызывайте злость, демонстрируя недоверие при обсуждении соглашений о поддержке, но и не становитесь мальчиком для битья в проекте, игнорируя эту проблему в надежде на солидарность трудящихся.

Наконец, убедитесь, что у вас есть план управления документами, которые будут появляться в ходе тестирования. Самый простой способ добиться этого — воспользоваться репозиторием для исходного кода или системой управления конфигурацией, применяемой в проекте для исходного кода системы. (В каждом проекте в настоящее время используется одна из таких систем, не так ли?) Как только появляется какой-либо документ, скрипт, отчет, график, таблица и т.п., после утверждения необходимо поместить его в этот репозиторий. Присвойте этому объекту версию и увеличивайте ее на единицу при каждом обновлении.

Наряду с отслеживанием объектов выберите способ, чтобы следить за тем, какой из объектов применялся для тестирования каких версий системы. Например, если имеется набор тестовых сценариев, версия 6.1.2 которого помещена в репозиторий, нужно иметь возможность открыть некоторый документ с перечнем перекрестных ссылок, позволяющий узнать, что вы использовали эту версию тестовых сценариев для тестирования версий системы, начиная с 3.1.007 и заканчивая 3.1.021.

Может показаться, что этот способ систематизированной каталогизации и хранения компонентов тестирования требует больших затрат времени, особенно если вы не знакомы с принципами управления конфигурациями. Однако построение репозитория продуктов тестирования многократного применения, тестовых данных, документов тестирования и других объектов очень полезно для будущего. Как раз потому, что ваша система — не одноразовое множество исходного кода, тесты являются ценными объектами для будущих проверок качества системы[1].

---

[1] Превосходное ведение управления конфигурацией, включая необходимость хранения артефактов тестирования, можно найти в статье Тима Кассе (Tim Kasse) и Патрисии Макуэйд (Patricia McQuaid) "Software Configuration Management for Project Leaders", которая опубликована в сентябрьском 2000 г. номере журнала "*Software Quality Professional*".

Риски подпроекта тестирования

Мы не только должны использовать стратегии тестирования на основе управления рисками для предоставления наилучших информационных продуктов и услуг по управлению рисками качества заинтересованным лицам и внутренним заказчикам, мы также должны иметь стратегию управления рисками для самого подпроекта тестирования. Легче всего начать размышления на тему рисков подпроекта тестирования с вопроса самому себе, какие элементы должны удачно совпасть для успеха подпроекта тестирования. Никакие два проекта не будут одинаковыми, поэтому необходимо опираться на отличительные черты, проявляющиеся в ходе всего подпроекта тестирования. Однако риски обычно растут в одном или нескольких направлениях, аналогичных направлениям проектов разработки.

- *Свойства.* Какие элементы, факторы и действия способствуют получению надлежащей совокупности услуг управления рисками качества и информационных продуктов?
- *Сроки.* Какие элементы, факторы и действия способствуют получению этих услуг и продуктов своевременно и в указанные сроки?
- *Бюджет.* Какие элементы, факторы и действия способствуют получению услуг и продуктов в рамках общего бюджета проекта путем, ведущим к положительному сальдо окупаемости вложений в тестирование?
- *Качество.* Какие элементы, факторы и действия способствуют получению услуг и продуктов группы тестирования, в целом удовлетворяющих заинтересованных лиц и внутренних заказчиков и лишь изредка (если есть в принципе) неудовлетворяющих их?

Насколько высок риск, связанный с этими элементами, факторами и действиями, настолько успех проекта находится в зависимости от этого риска.

Часто можно снизить риски, касающиеся только свойств и бюджета, за счет предварительного обсуждения. Каковы корректные рамки тестирования, и сколько за это готовы заплатить?

Проблема свойств и бюджета проекта тестирования возникает в том случае, когда руководство диктует: "Вот вам небольшая недостаточная сумма средств, теперь идите и сделайте так, чтобы все проблемы качества исчезли". Как уже обсуждалось в главе 1, если вы ясно представляете контекст, представляете риски качества, которыми нужно управлять, и способствуете пониманию другими людьми возможных компромиссов, можно добиться ликвидации этих нереальных ожиданий. Данные риски становятся вопросом компетенции руководства проекта, которое управляет бюджетом и хочет знать, что деньги эффективно расходуются в рамках проекта тестирования.

Риски качества для проектов тестирования по большей части связаны со способностью информировать заинтересованных лиц о результатах тестирования (см. главу 15.) Этот вопрос необходимо принимать в расчет при планировании. В основном примере книги Джамал рассылает на утверждение руководству формы отчетов и графиков. Если невозможно ут-

вердить предлагаемые отчеты или кажется, что люди не вникают в их смысл, возникает значительный риск в понимании качества проекта тестирования. Снова отметим, что ключевым моментом является корректное ограничение ожиданий заинтересованных лиц.

Риски сроков возникают из-за запутанных взаимосвязей подпроекта тестирования и проекта разработки. Простейший пример: риск сроков для проекта тестирования возникает по причине поздней поставки главной системы. В одном из проектов сервер стоимостью в четверть миллиона долларов застрял на таможне на две недели, поскольку никто не мог понять, как заплатить пошлину, что, естественно, отсрочило на две недели все тесты, запланированные к выполнению.

Особенно неприятны риски, связанные со сроками. Можно выпросить больше денег, можно не реализовывать свойства, можно расслабиться по поводу требований к качеству, но если день потерян, он потерян навсегда. Притом, что тестирование страдает от сдвига сроков, соответствующая дата окончания фазы тестирования часто не может быть сдвинута. Это может привести к болезненным компромиссам на уровне проекта, либо к поставке недостаточно протестированного и отлаженного продукта, т.е. с плохим и, вероятно, даже не полностью проверенным качеством, либо к упущению хорошего момента на рынке для продаж. Тщательное выявление рисков и управление ими, сопровождающее весь процесс тестирования, способствует предотвращению этих проблем. Еще помогает совместная работа с группой управления проектом над получением реалистичного графика проекта в целом[1].

Свистать всех наверх

Применяя процессы и политики, рассмотренные в этой главе, я полагаю, что могу создать краткие и полезные документы, которые, в частности, помогают понять и обсуждать смысл тестирования в проекте. Хороший план тестирования — это богатая смесь очищенных данных планирования (полученных из анализа рисков, графика, бюджета) с подробными процессами, хорошо продуманными политиками, разумными стратегиями и взаимными договоренностями с заинтересованными лицами. Из этого плана в проекте вырастает тестирование.

Но я не пишу хорошие планы тестирования самостоятельно. Мне нужна помощь других сотрудников проектной группы. Перед тем как двигаться вперед, мы должны убедиться, что наш план высокого качества и поддерживается заинтересованными лицами. В следующей главе продолжается изучение планирования тестирования. Мы рассмотрим опыт Джамала по привлечению всех заинтересованных лиц к обсуждению плана тестирования, а затем попытаемся понять, как разработать наиболее качественный план.

---

[1] В главе 2 я упоминал книгу Стива Макконела "*Software Project Survival Guide*", с. 93—101, в которой обсуждается управления рисками проекта. Вот еще один случай, где эти методы могут быть полезными.

ГЛАВА 7

От предложения к утверждению: поиск поддержки качественных планов тестирования у заинтересованных лиц

Написать план тестирования полезно, но план — это лишь документ, набор слов. Эти слова оживают только тогда, когда рабочая группа проекта начинает действовать согласно плану. Таким образом, документированный план — это предложение, которое имеет смысл лишь тогда, когда люди приходят к решению выполнять его. Различные ключевые фигуры и внутри, и вне группы тестирования должны согласиться делать то, что от них требуется. Другие ключевые лица, заинтересованные в процессе тестирования и качестве системы, должны договориться, что тестирование, описанное в плане, будет соответствовать их потребностям. Иными словами, будет ли тестирование предоставлять заинтересованным лицам информацию и услуги, которые способствовали бы управлению рисками качества системы? Как только договоренности достигнуты, вы получаете больше чем документ. Вы получаете общедоступную карту маршрута к ключевой части будущего проекта и поддержку специалистов.

Необходимо выполнить серьезную работу на пути от первой черновой версии плана тестирования к общедоступной карте маршрута и к формированию сообщества сторонников, чтобы заручиться поддержкой. В этой главе мы рассмотрим план тестирования, разработанный в предыдущей главе, и пройдем оставшиеся этапы подготовки плана. Затем мы снова взглянем на весь процесс планирования тестирования, обратив внимание прежде всего на то, чем отличаются грамотное планирование тестирования и качественный план тестирования, и на некоторые существующие проблемы и пути внедрения усовершенствований.

Джамал выполняет подачу

Утром 19 сентября Джамал пришел в офис и попал под гром и молнии, вызванные "странной пользовательской средой", о которой накануне говорил Кейт Ли. Менеджер пользовательской поддержки Косимо Неграев и менеджер продукта JetCalc Джайеш Депрендра уже ждали его. Они не в самых деликатных выражениях говорили о том, что в предыдущей версии JetCalc Лин Цу пропустила ошибку. Первой мыслью Джамала было, что ошибка зависит от конфигурации среды, поэтому все утро он сначала успокаивал их, затем пытался вместе с Лин Цу воспроизвести ошибку в тестовой среде и наконец продемонстрировал Джайешу и Косимо, что, хотя, по всей вероятности, можно воспроизвести ошибку вне особой конфигурации среды заказчика, она является довольно необычной.

"Простите, друзья, что вам пришлось через это пройти, — сказал Джамал, — но, честно говоря, я не думаю, что можно найти все проблемы, связанные с различными конфигурациями, в нашей тестовой среде с теми ресурсами, которыми мы располагаем. Джайеш, может быть, мы подумаем вместе над организацией тщательного бета-тестирования, которое позволит проверить большее число конфигураций среды и данных, которых у нас нет?"

"Хорошо, мы должны что-нибудь придумать! — воскликнул Джайеш. — Я не хочу прочитать в очередном обзоре продуктов фразу "Самая ужасная электронная таблица на рынке", если только речь не идет о наших конкурентах".

После этого Джамал погрузился в чтение электронной почты. Надо сказать, что отзывы на план тестирования потекли тонкой струйкой. Большинство из них были уточнениями формулировок, но Кейт Эрнандес сделала серьезное замечание. "Знаешь, может, я что-то упустила,– писала она. — Но где в тестовой среде серверы для управления иерархическим хранилищем?"

"Боже мой, — подумал Джамал, — мы пропустили это на семинаре FMEA. Позвонив Кейт, он подтвердил, что в настоящий момент устройства для управления иерархическим хранилищем в тестовой лаборатории отсутствуют. "А откуда их взять? — спросила Кейт. — Это первая версия, поддерживающая работу с иерархическими хранилищами".

"Хорошо, жди новых заявок от твоего покорного слуги, — сказал Джамал, — поскольку нам придется еще кое-что прикупить. Нам нужен хотя бы один сервер".

Джамал написал Кейт: "Отличное замечание! Работаю над устранением ошибки".

Затем Джамал отправил сообщения нескольким сотрудникам группы разработки, которые работали над поддержкой управления иерархическими хранилищами. Все они ответили, что убеждены в незначительности технического риска, поскольку системы управления иерархическими хранилищами используют проверенную технологию, в которой не будет ничего нового, но согласились с тем, что, по крайней мере, один сервер должен быть в тестовой комбинации. Далее он зашел в Интернет и выяснил цены на программы управления иерархическими хранилищами и устройства хранения. Позвонив Кейт, Джамал рассказал про обсуждение с

программистами и спросил, нет ли у нее каких-либо пожеланий по поводу конфигурации управления иерархическими хранилищами. Она ответила, что ей известно, какие программы и устройства используют заказчики, но ей не нравится мысль о расходовании денег на системы, позволяющие воспроизвести все такие конфигурации сред. Она предложила Джамалу выбрать приемлемую по деньгам конфигурацию, которая включала бы различные свойства, необходимые для разрабатываемых программ.

"Вот о чем я тебя попрошу, — сказал Джамал. — Позвони, пожалуйста, Джайешу и спроси, подходит ли для него этот план с учетом всего, что произошло сегодня утром".

"Зачем? Что случилось с JetCalc?"

Джамал пересказал Кейт всю историю, добавив в конце: "Итак, по существу риск состоит в том, что чем меньше конфигураций мы проверим, тем более вероятно проявление подобных проблем при эксплуатации".

"Понимаю, но если мы говорим о сумме в $7500 и более на серверы с полной конфигурацией, нельзя построить много конфигураций, не пробив серьезную брешь в бюджете. Как насчет того, чтобы взять только две: одну на базе NT, а вторую на базе UNIX?"

"Меня это устраивает", — ответил Джамал. Он подготовил и разослал следующее письмо всем участникам семинара FMEA и участникам обсуждения плана тестирования. В список адресатов он включил также Джайеша, чтобы дать ему возможность высказать мнение по поводу проблем, которые может вызвать минимальное число тестовых конфигураций.

Привет всем!

Проводя рецензирование плана тестирования, Кейт Эрнандес заметила, что в ходе анализа рисков мы не определили ни одного сервера управления иерархическими хранилищами в тестовой среде. Обсудив с разработчиками и изучив ситуацию по Интернету, я пришел к выводу, что технический риск минимален, хотя едва ли я являюсь специалистом в этой области.

Поскольку даже серверы начального уровня для управления иерархическими хранилищами стоят около $7500 с учетом конфигурирования, Кейт и я предлагаем закупить две конфигурации: одну на базе NT, а вторую на базе UNIX.

Если у кого-то есть мнения по этому вопросу и аргументы за и против нашего предложения, прошу сообщить об этом.

С уважением,

Джамал

Началось обсуждение. Люди писали электронные сообщения и заходили в офис Джамала. Обсуждение прекратилось с письмом Джона Албертсона, вице-президента по поддержке разработок и непосредственного начальника Джамала.

Всем!

После полезного обсуждения, в которое внесли вклад все участники, прошу принять к сведению, что Гарольд, Макс, Раджеш и я обсудили вопрос о серверах для управления иерархическими хранилищами для тестовой среды

проекта "Суматра". Хотя Джайеш и другие верно указали на необходимость разнотипных тестовых конфигураций, верно и то, что мы не можем протестировать все. Мы должны здесь уравновесить конкурирующие приоритеты. Мы думаем, что предложение Кейт и Джамала о тестировании на двух конфигурациях позволяет удачно уравновесить бюджет и качество. Когда мы обсуждали бюджет на тестирование, Джамал включил значительную сумму на непредвиденные обстоятельства. Закупка двух дополнительных серверов потребует израсходовать около 25% этой суммы. В качестве компенсации группа Джамала закроет при тестировании две популярные конфигурации серверов для управления иерархическими хранилищами. Это, по мнению старших менеджеров, является разумным балансом. Итак, мы даем указание Джамалу действовать в соответствии с этим решением.

Повторюсь, что мы благодарим всех за участие в обсуждении. На этом прошу считать обсуждение вопроса закрытым и направлять любые дополнительные комментарии вашим непосредственным начальникам.

Дж.А.

Джамал не смог сдержать улыбки, прочитав это письмо: "Класс! Было бы здорово, если бы письмо было в моем архиве, когда утром я столкнулся с Джайешем. Это помогло бы мне избежать роли мальчика для битья по вопросам качества". В пятницу 20 сентября он обновил план тестирования и бюджет на основе предложенных изменений и разослал новые варианты.

Привет всем!

Пожалуйста, рассмотрите готовую к рецензированию вторую версию плана тестирования, которая прикреплена к сообщению. Мы обсудим этот вариант во вторник на совещании. С нетерпением жду встречи с вами.

С уважением,

Джамал

№ этапа	Этап	Выполнено?
4.	Разослать план для закрытого (автономного) рецензирования, обычно сначала сотрудникам группы тестирования, а затем на более широкое обсуждение среди заинтересованных лиц и участников проекта. Собрать отзывы и пересмотреть план, повторив этапы 1–3 по необходимости. Изучить любые изменения в имеющемся графике и бюджете (превышающие оговоренные резервы времени и планы на случай непредвиденных обстоятельств), полученные в результате процесса планирования, и получить поддержку этих изменений у руководства.	☑

Совещание, посвященное обсуждению плана тестирования, не было богато на события. Джамал тщательно разобрал план, страницу за страницей, представив его основные фрагменты. Участники Джон, Гарольд, Дженни, Кейт Эрнандес, Ясухиро, Косимо, Речел, Макс, Джейм, Бобби, Лин Цу, Эмма и два старших программиста ели сэндвичи и задавали по ходу обсуждения вопросы. Несколько человек внесли полезные предло-

жения, которые привели к небольшим изменениям плана. В частности, Дженни предложила, чтобы примечания к версии (release notes) описывали известные ограничения и ошибки, помимо самого содержимого версии.

Завершая совещание, Джамал сказал: "Итак, с учетом изменений, о которых мы договорились сегодня, всех ли устраивает план? Нет ли чего такого, что бы не имело смысла?" Он осмотрел комнату и увидел только согласно кивающих людей.

"Неразумных требований к персоналу? Необдуманных предложений? Нелепых предположений?"

В ответ только смех и согласное кивание.

Джон сказал: "Джамал, я думаю, план отличный. Если мы сможем работать в соответствии с этим документом, тестирование пойдет хорошо. Макс, Гарольд, вы согласны?"

"Безусловно," – сказал Гарольд. Макс согласно кивнул.

"Хорошо, – сказал Джон. – Совещание закончено".

№ этапа	Этап	Выполнено?
5.	Разослать план для открытого рецензирования. Провести совещание по итогам рецензирования с участием всех заинтересованных лиц. Сделать все необходимые заключительные уточнения и удостовериться, что с учетом всех согласованных на совещании изменений план для подпроекта тестирования зафиксирован на бумаге и утвержден.	☑

Джамал, довольный тем, что у него есть надежный план с надежной поддержкой заинтересованных лиц, вернулся в свой кабинет. Он внес незначительные изменения, которые обсуждались на совещании, а затем поместил план в репозиторий документов проекта.

№ этапа	Этап	Выполнено?
6.	Пересмотреть оценку графика и бюджета на основе новой информации, полученной в ходе процесса планирования, включая использование ресурсов. Если в результате происходит сдвиг сроков или увеличение бюджета за рамки резервов или планов на случай непредвиденных обстоятельств, эскалируйте проблемы для решения на соответствующий уровень руководства. Обсуждение новых бюджета и графика может повлечь повторение этапов планирования или переработку оценки затрат.	☑
7.	Поместите план в библиотеку проекта или в систему управления конфигурацией. Ведите контроль изменений документа.	☑

Построение грамотного процесса планирования тестирования

В документах и шаблонах, которые являются хорошим дополнением к этой книге, можно посмотреть на план тестирования для проекта "Сумат-

ра»[1]. Вероятно, вы согласитесь со мной, что это хороший план тестирования. Но качественный процесс планирования позволяет группе тестирования сделать больше, чем просто подготовить хороший документ.

Установка четких критериев для фаз

В главе 1 я объявил о своем нейтралитете или, по крайней мере, открытости по вопросу жизненных циклов программных продуктов. Я уверен, что важным соображением для тестировщиков является согласование процесса тестирования с выбранными жизненными циклами и методологиями. Один из путей достижения этого — выбор соответствующих критериев начала, продолжения и завершения фаз тестирования, которые мы должны проводить. Если взглянуть на критерии, которые появились в документах проекта "Суматра", можно обнаружить, что они выясняют состояние качества и определяют рубежи для процесса в проекте. Другими словами, они устанавливают стандарты, с помощью которых можно ответить "да" или "нет" на три вопроса:

- Может ли группа проекта предоставить группе тестирования все необходимое для начала, продолжения или завершения данной фазы тестирования (готовность проекта)?
- Имеется ли у тестировщиков все необходимое, и завершили ли они все приготовления для начала, продолжения или завершения данной фазы тестирования (готовность подпроекта тестирования)?
- Является ли качество системы достаточным для начала, продолжения или завершения данной фазы тестирования (готовность качества системы)?

Ответы на первые два вопроса обычно можно дать со всей определенностью. Позвольте проиллюстрировать это с помощью следующих примеров.

Предположим, что один из критериев начала системного тестирования состоит в том, что все свойства, запланированные для версии, реализованы. Предположим также, что один из критериев продолжения комплексного тестирования состоит том, что все тестовые версии собираются из компонентов исходного кода под контролем средств управления конфигурациями. Пусть еще один критерий начала комплексного тестирования состоит в том, что установлена система управления дефектами. Пусть критерием завершения системного тестирования является то, что все запланированные тесты выполнены. Все эти критерии могут быть либо выполнены, либо не выполнены. Это примеры, которые не применимы ко всем моделям жизненного цикла, но применимы к некоторым из них. Квалифицированные и опытные специалисты-тестировщики, понимающие динамику контекста жизненного цикла проекта, могут внимательно отобрать подходящие и однозначные критерии готовности проекта в целом и подпроекта тестирования в частности.

---

[1] Этот и другие сопроводительные документы можно найти в разделе "Публикации" (Publications) на www.rexblackconsulting.com.

Даже при тщательных размышлениях и четком понимании выбранной модели жизненного цикла выработка точных критериев готовности качества системы остается ненадежной. Как узнать, что система достаточно хороша, чтобы можно было начать, продолжить или объявить завершенной фазу тестирования? Это трудные решения, требующие размышлений и проницательности. Тщательно документированные критерии могут способствовать этим размышлениям. В случае начала или продолжения тестирования я обычно описываю готовность качества (для начала и продолжения, а не завершения тестирования) с точки зрения возможности эффективного и качественного выполнения тестов.

Легко сказать, что система не готова к тестированию в экстремальных условиях. Я видел системы, которые зависали каждые 15 минут, и требовались длительная перезагрузка и переконфигурирование. Тестирование двигалось со скоростью улитки, если двигалось вообще, и позволяло узнать о качестве системы очень немного. Я тестировал системы, в которых из-за большого числа серьезных ошибок не работали многие функции, в результате большинство тестовых сценариев невозможно было довести до завершения. Дальнейшее тестирование не давало никакой полезной информации, поскольку система блокировала любые попытки движения. Я работал в проектах, в которых инженеры сборок не могли собрать систему в течение нескольких дней. Группа сборки версий могла предоставить систему, которой (и исправлениям ошибок в ней) было уже несколько недель. Мы не могли получить никакой значимой информации о текущем состоянии системы с помощью тестирования такой старой версии.

Помимо этих серьезных препятствий, становится менее ясным вопрос о качестве. Если мы можем тестировать, но лишь наполовину, означает ли это, что мы слишком агрессивно планировали или что система столь плоха, что нужно приостановить тестирование? Верно ли, что число ошибок, которые нуждаются в исправлении (о которых написаны отчеты и принято решение об исправлении), настолько велико, что дальнейшее тестирование лишь увеличит непонимание состояния системы и уменьшит возможности группы тестирования определять, какие ошибки нуждаются в исправлении? В нашем примере я выбрал число 50, так как, судя по моему опыту, более 50 открытых ошибок нельзя обсудить в ходе часового совещания по определению очередности исправления дефектов. А с любого совещания продолжительностью более часа, участники обычно начинают уходить. Тем не менее, различия в контексте проекта могут повлиять на это число. Например, если на совещаниях дефекты обсуждаются подробно, то, вероятно, подходящим числом является 20.

Еще сложнее дать ответ на вопрос, касающийся завершения определенной фазы тестирования. Как мы можем узнать, что обнаружили достаточно дефектов в ходе тестирования (и, надеюсь, исправили их) и что объем проведенного тестирования достаточен? Я встречал людей, которые пытались численно решить этот вопрос. К примеру, я видел такие критерии завершения системного тестирования:

1. Ни одной ошибки серьезности 1, менее 25 ошибок серьезности 2 и менее 50 ошибок серьезности 3.

2. Не более 10% тестовых сценариев, не завершенных успешно.
3. 90%-ное покрытие ветвей и условий в ходе всей фазы системного тестирования[1].

Считаю эти критерии хорошими в теории, но слабыми на практике. Можно вспомнить множество несущественных аргументов программистов и менеджеров по поводу классификации определенных отчетов о дефектах как ошибок серьезности 1 или 2. Комбинируя тестовые сценарии и манипулируя их числом, обычно нетрудно сделать количество успешно и неуспешно завершенных тестовых сценариев таким, каким нужно. А поскольку тесты в основном проходят большинство ветвей решений, означает ли это, что мы проверили интересные условия, изучили важные потоки данных и сравнили полученные результаты с ожидаемыми?

По этой причине я предпочитаю использовать виды критериев, показанные в планах тестирования Суматры. Применяя мой подход, тест-менеджер или ведущий тестировщик собирает информацию и измерения тестирования, которые дают возможность готовить значимые отчеты о качестве проверяемой системы, о ходе проекта тестирования, о реакции группы разработки на обнаруженные проблемы и о мнении о тестировании в целом. Эту информацию следует представить в удобной форме и передать группе управления проектом (см. главу 15.) Затем группа управления проектом должна уравновесить риски качества, о которых говорится в предоставленных отчетах, с рисками бюджета, сроков и, вероятно, свойств системы.

Уравновешивание рисков применимо не только к критериям, относящимся к качеству. Группы управления проектами могут отказываться и действительно отказываются от критериев готовности проекта и подпроекта тестирования иногда по вполне объяснимым причинам, связанным с бизнесом. В некоторых случаях, однако, уравновешивание рисков не является верным, и необходимо использовать критерии качества, измеряемые численно. Примером такого случая является поставка продуктов, чувствительных к безопасности, а также продуктов, критерии тестирования которых фиксируются в контракте. В этих случаях тест-менеджер должен принять соответствующие критерии на основе имеющихся ограничений.

Достижение консенсуса, согласование ожиданий, получение одобрения

В большинстве моих проектов наиболее важной частью процесса планирования тестирования было не создание документа, а достижение консенсуса, согласование ожиданий и получение одобрения, связанные с документом. Мой подход к планированию предполагает участие всех заинтересованных лиц. Они получают возможность заблаговременно узнать, что именно я собираюсь тестировать, а что не буду тестировать. О какой помощи я намереваюсь их попросить. Как группа тестирования

---

[1] Обсуждение различных методов структурного покрытия, а также критериев начала и завершения тестирования есть у Бориса Бейзера в "*Software Testing Techniques*".

будет взаимодействовать с их группой. Как, по моему мнению, мы будем действовать в случае непредвиденных ситуаций. Более того, они не только узнают о том, что я предлагаю, они также могут участвовать в обсуждении этих предложений.

Процесс, который я применяю для подготовки планов тестирования, в настоящий момент достигает кульминации при подходе, основанном на участии всех заинтересованных сторон и поиске консенсуса. Данный подход я использую в ходе всей части "Этап 1. Планирование" процесса тестирования, описываемого в главе 1. Джамал выстраивал консенсус по поводу контекста тестирования в главе 1, по поводу критических рисков качества – в главе 2, по поводу определения перечня задач, оценок затрат времени и ресурсов – в главах 3 – 5, и, наконец, в главах 6 и 7 он ищет консенсус по поводу плана. Другими словами, документирование плана – это заключительный этап в достижении того, что каждое заинтересованное лицо в подпроекте тестирования знает, что вы намерены делать, и что все ключевые игроки знают свои роли и сферы ответственности.

Однажды за чашкой кофе я услышал от коллеги следующее остроумное высказывание: "Я – менеджер. Моя задача – не допустить интересных событий". Можно сказать, что целью планирования является следующее: предотвратить неприятные сюрпризы, вызывающие панические настроения. Такие сюрпризы случаются, если мы не смогли предвидеть сложные взаимосвязи и допущения, касающиеся того, что мы собирались сделать. Такие сюрпризы случаются, когда мы просим других о поддержке, которую они не готовы оказать. Такие сюрпризы случаются, когда мы докладываем о необоснованных, необъяснимых результатах тестирования или делаем это несвоевременно. В плане тестирования мы сначала указываем на неизвестные или скрытые детали и согласовываем предложения о том, как мы будем противостоять им. Квалифицированный тест-менеджер или инженер-тестировщик при создании плана представляет аудиторию, что люди хотят знать, что они уже знают, и как взаимодействовать с ними с помощью документа.

Другой аспект, связанный с аудиторией, касается того, где дислоцируются потенциальные предметы дискуссии. План тестирования должен четко показывать возможные области расхождения во мнениях. Если это возможно, предложите решение, которое, по вашему мнению, устроило бы все стороны. Если вы не уверены, что у вас есть такое решение, или ожидаете, что участники обсуждения ввяжутся в полемику, можно использовать процесс планирования тестирования для содействия обсуждению. В главном примере Джамал привлекает всю группу проекта к выработке подходящего уровня покрытия конфигураций. Если бы Джамал просто написал, что надо протестировать только два сервера для управления иерархическими хранилищами, тогда он рассматривался бы как автор этого решения и отвечал бы за его последствия. Вместо этого он спровоцировал дискуссию о выравнивании рисков на уровне управления проектом и затем "позволил" своим менеджерам принять решение.

Джамал организовал и обсуждение, и решение вопроса с помощью электронной почты. Я тоже проводил такие обсуждения в рамках чернового варианта плана тестирования. Например, Джамал мог бы включить

[TBD (to be defined – будет определено позже): нам нужно решить, сколько серверов для управления иерархическими хранилищами включить в тестовую среду. Мы с Кейт Эрнандес предлагаем два: один на базе NT, второй на базе UNIX, – поскольку это уравновешивает влияние дополнительного оборудования на бюджет с потребностью протестировать различные варианты конфигураций управления иерархическими хранилищами. Джайеш, Джон, Гарольд и Макс, прошу Вашего совета по этому вопросу.]

следующий абзац в план тестирования, непосредственно в раздел, где определяется тестовая среда.

Такой подход полезен, если есть время подготовить итоговый план на основе нескольких черновиков. Этот процесс имеет обыкновение затягиваться, и вам приходится действовать в качестве центра, сопрягающего все последующие обсуждения. Например, устраивать обсуждение один на один, хотя, возможно, более эффективно собрать все открытые проблемы и разослать их всем за один раз, а не распространять десятки электронных сообщений всем участникам процесса, задавая каждому касающиеся его вопросы[1].

Один из аспектов, касающихся подхода к выработке критериев, как было показано ранее, связан с тем, что группа управления проектом должна подтвердить свою поддержку предлагаемого подхода. Фиксируя критерии, мы договариваемся с группой, что все мы будем совместно готовить решение о том, что движение по фазам проекта будет осуществляться с учетом того, что группа тестирования может предоставить информацию, необходимую для оценки рисков качества. Другими словами, тестировщики согласны предоставить определенный перечень услуг, а проектная группа совместно принимает на себя ответственность за принятие конкретных решений.

На уровне коллег существует другая форма соглашения с ответственными за основные процессы. Например, в проекте "Суматра" Ясухиро и

---

[1] Пример плана тестирования с отметками TBD (будет определено позже) приведен в шаблонах к книге "*Managing the Testing Process, Second Edition*", на web-сайте www.rexblackconsulting.com. Рецензент этой книги, тест-менеджер компании Macromedia Ребекка Совэда (Rebecca Sowada), упомянула один из методов, на который стоит обратить внимание. Она пишет: "Мы делаем что-то похожее [с использованием TBD], за исключением того, что наши планы тестирования готовятся в формате HTML и публикуются на интранет-сайте проекта. Поскольку у нас ведется контроль изменений любого документа на сайте, каждый имеет доступ к редактированию этого документа. Я просто создаю в плане раздел "Комментарии", где сотрудники могут задавать вопросы по ходу рецензирования плана. Позднее довольно просто привести план в порядок и установить ссылки на комментарии, относящиеся к конкретному уточнению. Или просто добавить ответ на комментарий непосредственно в разделе "Комментарии". Справедливости ради надо сказать, что я работаю с небольшой, дружной группой людей, которым я доверяю редактировать мой план тестирования, так что данный подход зарекомендовал себя хорошо. Мы по-прежнему проводим совещание, посвященное обсуждению плана, но я считаю, что данный подход более управляем, чем дебаты с помощью электронной почты. Кроме того, это позволяет избежать обвинений, что я кого-то забыла включить в список рассылки".

Джамал договорились о том, как будут поступать на тестирование тестовые версии. Кейт Ли и Джамал договорились о том, как будет происходить поддержка тестовой среды. Дженни и Джамал договорились о передаче отчетов о дефектах. Договоренности и соглашения достигаются в ходе процесса планирования и документально фиксируются в плане тестирования.

Формализация договоренностей и соглашений происходит различными путями. В некоторых случаях может хватить просто разговора и документирования его результатов на бумаге. В других случаях это достигается только в результате участия в совещании всех заинтересованных лиц и ключевых участников проекта.

В некоторых проектах более формализованный процесс приводит к подписанию документов заинтересованными лицами и ключевыми участниками, это означает, что они ознакомлены с планом, понимают его смысл и будут участвовать в его осуществлении. Решение о том, какой путь применять, целиком зависит от контекста. Выбирайте подход, наиболее подходящий для компании и проекта.

Проведение планирования в разумное время

Когда наступает подходящее время для планирования? Чтобы ответить на этот вопрос, позвольте вернуться к процессу тестирования в целом. Этап планирования этого процесса включает в себя создание контекста, но план также определяет, как будет происходить создание оставшейся части контекста. Процесс планирования происходит тогда, когда я обрисовываю в общих чертах необходимые мне тестовые сценарии, данные и инструменты, а также способы разработки и применения этих объектов. Процесс планирования происходит также тогда, когда я разрабатываю необходимые варианты тестовой среды и способы конфигурирования и поддержки этих вариантов. Некоторые из объектов и вариантов среды требуют существенных затрат. На получение или создание других нужно много времени. Третьи требуют, чтобы для их разработки или использования были наняты дополнительные сотрудники. Четвертые оказывают влияние на внешние по отношению к проекту группы людей из-за дороговизны, длительности или выхода за пределы их нынешних возможностей. О таких работах проектная группа должна быть предупреждена как можно раньше, что означает раннее осуществление планирования.

Однако здесь "работает" еще одна сила: кривая обучения (learning curve). Чем раньше я начинаю планирование, тем меньше я знаю о том, как будут происходить события, какие стратегии будут наиболее эффективными для тестирования данной системы, от каких видов дефектов будет страдать система и какую архитектуру и какие функции системы мы на самом деле реализуем. Это означает, что я должен планировать поздно.

В статье, размещенной на web-сайте www.stickyminds.com, посвященном тестированию и качеству программных продуктов, Джеймс Бах (James Bach), обсуждая планирование, проводит аналогию между планированием и игрой в американский футбол. Ли Копленд (Lee Copeland) на-

писал статью, в которой он расширил эту аналогию. В футболе тренеры и защитники выкрикивают варианты игры, часто взятые из книжечек с вариантами, но они также должны думать самостоятельно, на основе развивающихся событий[1].

Однако, продолжая аналогию, игра в американский футбол больше, чем обычная беготня 22 человек по полю с надутым кожаным эллипсоидом. Например, огромные усилия необходимы для поддержания травяного покрова на полях. Требуются годы, чтобы построить стадион. Даже для бейсбола, кроме ромба, находящегося на открытом пространстве, требуется запланировать приобретение баз, одной-двух бит, нескольких мячей, перчаток, рулетки и т.п. Для соревнований нужны две команды, и если мы собираемся посмотреть хорошую игру, участники команд должны обладать соответствующими умениями и навыками. Если нам нужны зрители, придется обзавестись трибунами.

Проект тестирования, как и любой проект, тоже имеет общие черты. Некоторые активности проекта требуют принятия по ходу самостоятельных решений, другие — координации действий, времени для получения или подготовки, финансовых затрат (требуется время для получения денег) и привлечения людей (которых необходимо найти, собрать вместе и обучить). Планирование должно учесть все эти долгосрочные задачи и завершиться достаточно рано, чтобы можно было своевременно произвести необходимые для качественного тестирования действия.

Разумная гибкость и творчество

Необходимость раннего планирования не отрицает замечаний Баха и Копленда. Качественное планирование и качественный процесс планирования должны быть гибкими и оставлять место для творчества. Это, в частности, предоставляет квалифицированным инженерам-тестировщикам стратегическое направление верхнего уровня детализации, что затем позволяет им проектировать и разрабатывать нужные тесты с заранее определенными параметрами. В нашем примере Джамал сообщит инженерам-тестировщикам Лин Цу, Эмме и тому, кто займет вакантную должность, когда, по его мнению, они должны применять ручное, а когда автоматизированное тестирование, когда они должны использовать структурный подход, а когда подход на основе поведения системы, и так далее. Однако существует большая степень самостоятельности, допускающая творчество при проектировании тестов.

Процесс выполнения тестов, предложенный Джамалом, более ограничен. Младшие тестировщики прослушают учебный курс, на котором им расскажут о процедурах выполнения различных видов тестов, о том, как сообщать о результатах тестирования и как писать отчеты о дефек-

---

[1] Статья Джеймса Баха называется "Исследовательское тестирование и миф планирования» ("Exploratory Testing and the Planning Myth»)), а статья Ли Копленда называется "Исследовательское планирование» ("Exploratory Planning»)). Оба автора обсуждают планирование именно в контексте исследовательского тестирования, но те же принципы применимы в случае, если используются скрипты для ручного тестирования или автоматизированное тестирование. Ли Копленд также рассматривает исследовательское тестирование и планирование тестирования в "*Mastering Software Test Design*".

тах. Поскольку младшие тестировщики менее квалифицированны, чем инженеры-тестировщики, Джамал ограничивает их деятельность. Если, однако, вы работаете с группой, полностью укомплектованной квалифицированными инженерами-тестировщиками, вы можете оставить значительно больше места для трактовок и импровизации.

Я тоже планирую с учетом гибкости. Хорошие планы должны гнуться, но не ломаться под давлением проекта. Предположим, что вы планируете тестирование при недостаточных разделяемых аппаратных ресурсах, например с применением корпоративной универсальной машины (mainframe). Если запланировать только один цикл тестирования для обнаружения дефектов, которые подлежат обязательному исправлению, у вас не будет плана для перетестирования очередной тестовой версии с исправлениями дефектов. Аналогично, если запланировать назначения в проекты очень плотно, перемещение людей из проекта в проект без резерва времени между проектами, то, что произойдет, если потребуется кого-нибудь задержать на денек в проекте?

Способ рассуждений "а что, если" является частью создания гибкого плана. Встраивание резервов на эти "а что, если", по крайней мере, которые я могу предположить, и есть способ добавления гибкости в план. В главном примере книги Джамал перечисляет риски, непредвиденные обстоятельства и способы их смягчения на трех страницах. Это может быть слишком для вашего проекта, но Джамал планирует предприятие, стоимость которого превышает три четверти миллиона долларов, а затраты ресурсов больше трех с половиной человеко-года. В этих условиях должна быть проявлена достаточная осторожность.

Я применяю резервирование (времени и финансов) для обеспечения гибкости для тех "а что, если", по которым я не могу сделать предположений. В нашем примере Джамал определил 10%-ный резерв в бюджете и запланировал 50%-ное увеличение объема выполнения тестов. Он также запланировал исследовательское тестирование, поскольку ему известно, что Лин Цу, Эмма и новый инженер-тестировщик не в состоянии продумать все тесты заранее. Кроме того, Джамал зарезервировал время и явным образом запланировал изменение существующих тестов и создание новых в ходе проведения тестирования.

Хороший план должен способствовать гибкости его применения. Например, мы говорили ранее о необходимости определения в плане четких критериев начала, продолжения и завершения различных фаз тестирования. Однако было также отмечено, что эти критерии определяют в основном методы и моменты времени для проверки рисков качества по отношению к проекту. Рабочая группа проекта должна соотнести эти риски с бюджетом, сроками и свойствами системы и вполне может принять решение временно отказаться от каких-то критериев. Как вы поступите, когда это произойдет? Вы, по всей видимости, не сможете работать столь же производительно и эффективно, как раньше. Если не предусмотреть заранее таких ситуаций, вам придется принимать решение на ходу. Хотя это также может сработать, степень гибкости не будет такой, как если бы эта ситуация была заранее предусмотрена.

Например, в одном из проектов я запланировал провести нагрузочное тестирование для интеграции некоторых подсистем с помощью ав-

томатизированных инструментов. Однако инструменты не были *вполне* готовы, когда наступило время для выполнения нагрузочных тестов. Группа управления проектом временно приостановила критерий начала тестирования, определенный как «готовность инструментов тестирования». Мне пришлось импровизировать, что я и делал, тестируя параллельные потоки данных относительно интерфейсов с привлечением большой группы (примерно 50 человек) младших тестировщиков. Поскольку я не планировал такого мероприятия, нам пришлось обратиться в относительно дорогое агентство по найму временных сотрудников, причем некоторые из них не имели достаточной квалификации. Дополнительная проблема состояла в том, что практически не было времени на обучение, поэтому они сделали много ошибок, которые потом легли тяжким бременем на инженеров-тестировщиков. Еще одно неудобство из-за отсутствия у младших тестировщиков опыта тестирования выразилось в необходимости написания подробных тестовых скриптов, которые с учетом ограничений во времени потребовали от инженеров-тестировщиков потратить все выходные на подготовку скриптов при работе по 12 часов в день. Поскольку эти тесты писались в пожарном порядке и в неудобное время, мы сделали массу промахов, включая отчеты о псевдоошибках и пропуск ошибок. Я бы мог смягчить эти проблемы, если бы учел их при планировании.

Конечно, существуют разумные пределы гибкости и творчества. Если попытаться учесть в плане все возможные проблемы, можно обнаружить то, о чем как-то писал Томас Джефферсон (Thomas Jefferson): "Сколько боли нам причинили несчастья, которых никогда не было!"[1] Можете ли вы позволить себе полное дублирующее множество конфигураций тестовой среды в другом месте, соединенное с другой магистралью Интернета, лишь на случай попадания метеорита или бомбы в тестовую лабораторию? В некоторых случаях это, может, и нужно, но мне никогда не приходилось работать в проекте, требующем такой степени гибкости.

Творчество, естественно, тоже имеет свои пределы. Исследовательское тестирование, конечно, может и играет важную роль при тестировании многих проектов. Однако следует определить ограничения для тестировщиков таким образом, чтобы знать, что они тестируют и что они тестируют то, что нужно. Эти ограничения могут существенно меняться в зависимости от навыков и опыта группы тестирования, от необходимости позднее в точности повторять тесты, от того, насколько подробно должны быть документированы тесты и результаты тестирования и т.д.

Подготовка надлежащей документации

В большинстве проектов, в которых я участвовал, план тестирования не являлся сам по себе завершающим документом, а выступал в качестве средства проведения успешного проекта тестирования. Документированный план, который готовится в ходе процесса планирования, — это суще-

[1] Из книги Джефферсона "A Decalogue of Canons for Observation in Practical Life", которую можно найти среди его избранных работ на сайте Библиотеки Конгресса США www.loc.gov.

ственная часть обсуждения проекта тестирования с другими людьми и достижения документированных соглашений в сложном предприятии. Иногда план тестирования является самостоятельным продуктом. При работе с независимыми тестовыми лабораториями план тестирования всегда является документом, поставляемым заказчику.

Независимо от того, на что делается упор в первую очередь: на процесс планирование или на документ под названием "план тестирования", – некоторые специалисты-тестировщики готовят итоговый документ в соответствии со стандартом документирования. Например, в вашей компании может быть принят стандарт документов Института инженеров по электротехнике и электронике (Institute of Electrical and Electronic Engineers, IEEE). В этом случае план тестирования должен соответствовать стандарту IEEE 829. Определенные методологии разработки и инструменты, которые автоматизируют подготовку документов, могут оказывать влияние и даже определять, как документируется план тестирования. В ряде отраслей закон предписывает определенные стандарты документирования, в том числе и для планов тестирования. Примером является, в частности, разработка программного обеспечения для атомной промышленности и медицины.

Цели плана и процесса планирования также оказывают влияние на степень детализации и объем документов. Планы тестирования, написанные в соответствии с определенным стандартом, должны включать в себя все разделы, заданные в стандартном шаблоне. Планы тестирования, которые готовятся для сложных случаев (сложность предметной области, техническая, проектная сложность или сложность тестирования), обычно требуют большей детализации и более подробного описания, чем планы тестирования для простых случаев. Планы тестирования, подготовленные для использования группами людей, состоящими целиком из опытных специалистов, будут, вероятно, слишком непонятными для группы тестирования, составленной из людей с недостаточным опытом тестирования или с полным его отсутствием.

В нашем примере ситуация, в которой оказался Джамал, подобна второму случаю. Частично он имеет дело с определенным уровнем детализации, необходимым для пояснений в процессе обучения и курирования младших сотрудников. Он также готовит точный, подробный план тестирования.

Даже если нужно писать точные и детальные планы, я остаюсь приверженцем краткости. Я не хочу тратить время, силы и бумагу на описание аспектов проекта, которые очевидны или уже согласованы. Джамал не должен объяснять понятие Интернет-протокола (IP) в своем плане, поскольку он предполагает, что все, работающие над проектом, знают, что это такое. Он не должен объяснять, что тестировщики будут использовать эти IP-адреса для доступа к Суматре через web-сервер, поскольку полагает, что все читатели плана тестирования знают, как это делается. (Возможно, он должен обучить этому некоторых неопытных младших тестировщиков.) Хороший инженер-тестировщик или тест-менеджер при написании плана тестирования представляет, для кого он пишет документ и как с его помощью взаимодействовать с этими людьми.

Я полагаю, что изобразительные средства являются хорошими заменителями текста в плане тестирования. Описание контекста, в котором работает группа тестирования, и архитектуры тестовой среды — два очевидных примера, которые можно увидеть в плане тестирования для проекта "Суматра". Также с помощью образов можно описать сложные процессы. Даже при условии, что на рисование требуется больше времени и из-за размера картинок увеличивается общий объем документа, я считаю, что картинки помогают визуально представить, как работает система, и делают план тестирования более понятным.

В некоторых проектах необходимо готовить несколько планов тестирования. Фазы тестирования, которые значительно отделены друг от друга по времени, могут потребовать отдельных планов, поскольку, вероятно, не удастся получить всю необходимую информацию сразу для написания единого плана. В ряде случаев существенно отличается аудитория читателей планов для различных видов тестирования, что также ведет к написанию нескольких планов. Например, я работал в проекте по созданию Интернет-приложения, в котором были подготовлены отдельные планы для тестирования аппаратной и программной частей. Если нужно готовить несколько планов тестирования, существенно сократить работу помогает подготовка общей части, например описания процесса подготовки отчетов о дефектах, которая включается в один из планов, а в остальных планах только приводится соответствующая ссылка.

Краткость планов тестирования не означает, что работа по их подготовке тривиальна. Напротив, нет ничего более тривиального, чем взять шаблон и заполнить его тем, что приходит в голову в соответствии с заголовками в нем, а затем отдать 100-страничный документ на рецензирование. Такие документы не способствуют эффективному и качественному тестированию, они являются вычурным орнаментом или безнадежным жертвоприношением богам программирования. Краткость означает использование одного правильного слова вместо сотен почти правильных слов. Говорят, что Густав Флобер, автор "Мадам Бовари", тратил часы, мучительно подбирая каждое слово. Конечно, не следует тратить столько времени, но большое значение имеют ясность выражений и хороший стиль письма. Марк Твен писал: "Различие между верным словом и почти верным словом — это различие между молнией и светлячком".

Предоставление возможности для обнаружения ошибок

В применяемом мною процессе планирования документ проходит через многократные рецензии, проводимые ключевыми участниками проекта и основными заинтересованными лицами. Процессы рецензирования служат двум целям, первой из которых является достижение согласия и принятия обязательств. Вторая цель — это обычная цель любых обзоров документов: предоставить возможность для обнаружения ошибок. И автономное рецензирование, и обсуждение с глазу на глаз позволяют выявить пропуски и неудачные допущения до того, как они приведут к значительным проблемам в ходе тестирования.

В главном примере книги Кейт Эрнандес обнаружила, что системы управления иерархическими хранилищами не включены в перечень тестовой среды в первой версии плана тестирования. Это такая ошибка, что если бы Джамал не вынес план на обсуждение, то ее могли бы не обнаружить до начала тестирования, а позже она бы наверняка отвлекла силы группы Кейта Ли на ее решение, а также потребовала бы усилий со стороны группы тестирования на проверку конфигурации и, вероятно, вызвала бы задержку в проведении тестирования. Кроме того, если бы группа тестирования обнаружила ошибку в программе, связанной с серверами для управления иерархическими хранилищами, в конце тестирования, задержка в локализации ошибки могла бы оказаться пагубной для сроков и бюджета проекта.

Также в примере была проиллюстрирована необходимость достижения баланса при исправлении ошибок. Как и в случае с анализом рисков качества, при планировании тестирования существует тенденция расползания рамок тестирования, когда планы подвергаются рецензированию. Разные люди заботятся о разных рисках, как, например, Джайеш в нашем примере. Ничего не стоит внешнему заинтересованному лицу заявить: "Нужно включить мой любимый тест (или тестовую среду, или длину цикла, или ...) в тестирование". Но если вы управляете тестированием, вы должны оказывать влияние на свой проект. Необходимо уравновешивать даже законные требования ключевых участников проекта или заинтересованных лиц исправить ошибку (для снижения риска качества) с другими рисками, возрастающими в рамках подпроекта тестирования.

Тестировщики должны уметь управлять процессами рецензирования независимо от того, предоставляют ли они свои материалы для рецензирования или изучают чей-либо еще документ. Журнал "*Professional Tester*" опубликовал серию статей о критических умениях и навыках тестировщика. Их авторы считают рецензирование одним из таких умений и классифицируют его как умение работать с людьми[1]. Более подробно тема умений и навыков тестировщика будет рассмотрена в следующих двух главах, посвященных поиску и найму персонала. В то же время для процесса планирования тестирования ключевым является то, как тестировщики применяют рецензирование, инспекции, просмотры или другие статические методы тестирования для поиска ошибок в написанных ими планах тестирования.

Использование синергетики в планах проекта, разработки, сборки и интеграции

Последнее указание на грамотный процесс заключается в том, что план тестирования и процесс планирования тестирования соответствуют планам разработки, сборки, интеграции и соответствующим процессам планирования. Под этим я подразумеваю, что процесс должен удовлетворять двум соображениям.

---

[1] См. "People Skills, Including the Review as a Test Technique", *Professional Tester*, Volume 2, Issue 2 (June 2001).

План разработки (Development Plan)

План, определяющий, каким образом будут создаваться компоненты, составляющие систему, включая последовательность разработки компонентов.

План сборок (Build Plan, Release, Engineering Plan)

План, определяющий, каким образом будут строиться сборки для тестирования в реальных условиях.

План интеграции (Integration Plan)

План, определяющий, каким образом различные компоненты будут собираться в некоторую логическую многоэтапную последовательность до тех пор, пока не будет получена система целиком.

Первое соображение связано с объединением процессов планирования, происходящим в компании как помимо вас, так и с вашим участием. В идеальном проекте производятся следующие действия:

- Менеджер проекта или группа управления проектом создает план проекта.
- Менеджер или менеджеры разработки создают планы разработки и интеграции компонентов.
- Менеджер подготовки сборок или конфигурационного управления готовит план сборок.

В нашем примере Джамал должен знать, когда инкременты будут готовы для комплексного и системного тестирования. Он получает эту информацию из плана Дженни. Информация о том, что тестировщику необходимо учесть в плане тестирования, становится доступной в ходе процессов планирования, используемых другими людьми, и, таким образом, тестировщик должен встраивать свой процесс планирования в упомянутые процессы планирования.

Необходимость согласования процессов и планов приводит ко второму соображению. В нашем примере Джамал планирует тестировать полные системы в том виде, в каком они должны поставляться заказчикам, в ходе системного тестирования. Что будет, если план сборок, подготовленный Ясухиро, определяет разработку пользовательской версии в середине системного тестирования? Тестирование, будучи в конце многих процессов, в том числе и описанного выше, не может игнорировать реальность.

Многие проекты не предусматривают разработку всех четырех планов. Я выполнял обязанности тест-менеджера в проекте, который включал в себя разработку шести подсистем, и менеджеры групп программистов каждой подсистемы писали планы разработки, а я должен был готовить план интеграции и план комплексного тестирования. Я был вы-

Планы-полуфабрикаты

При гладко работающих процессах тестирования становится возможным применение заранее приготовленных планов тестирования, особенно при разработке небольших версий сопровождения и патчей. Когда я работал тест-менеджером версий сопровождения для одного приложения, у нас был стандартный комплект тестов, который наращивался на основе состава версии. Мы выполняли этот комплект в каждой версии сопровождения. Планирование и осуществление плана происходило одновременно.

Такой подход может работать даже в случае, когда объем тестирования сильно различается от версии к версии. В статье, размещенной на моем сайте, старший менеджер системы Билл Линд, отвечающий за поддержку качества в компании Freightliner, описывает шестиуровневый процесс тестирования, применяемый аналогично меню. (См. "Freightliner Test Level Matrix" на www.rexblackconsulting.com.) Проектная группа может затребовать тестирование по уровню 0, по уровню 6 или по какому-либо промежуточному уровню на основе определенных доводов, связанных с проектом. Некоторые из этих уровней требуют подготовки специального плана тестирования, другим достаточно стандартного плана. Уровни могут комбинироваться с добавлением к компонентам низшего уровня компонентов более высокого уровня, если это необходимо. Эти методы могут повернуть планирование тестирования от повторяющейся рутинной работы к единовременному вложению в совершенствование процессов, которые можно будет многократно усиливать вновь и вновь.

нужден добавлять важные детали в раздел подготовки тестовых версий плана тестирования, чтобы компенсировать отсутствие плана сборок. В этих случаях существует определенная вариативность рекомендаций и решений, и тестировщик, определяя процессы, которым должны следовать не подчиняющиеся ему сотрудники, не должен перестараться. Мой опыт показывает, что мало кто столь нежелателен в проекте, как тест-менеджер, который пытается устанавливать процессы для других групп, независимо от того, делает ли это он для блага проекта или для своего удобства.

В некоторых случаях планы *действительно* покрывают дополнительные части процесса разработки, которые имеют выход на тестирование, но не рассматривают важные вопросы процесса тестирования. Например, многие планы проектов не определяют процесс управления дефектами. С точки зрения процесса тестирования, это важное упущение. Сборка версий, поскольку она непосредственно связана и с программированием, и с тестированием в проекте, возможно, располагает планом, который ориентирован только на одну совокупность требований. С учетом того, что эти критические интерфейсы могут быть нечетко определены в одном из планов, я обычно определяю их в своем плане после того, как договорюсь с менеджерами о работе этих процессов. Изображение диаграммы контекста, аналогичной той, которая была подготовлена для проекта "Суматра", может способствовать улучшению понимания этих интерфейсов.

Решение проблем

В ходе процесса планирования и создания качественного плана тестирования возникает множество проблем.

Выход за пределы шаблона

В документах, поступающих читателю с этой книгой и имеющихся на моем web-сайте, приводится шаблон плана тестирования[1]. Я уже говорил о шаблоне плана тестирования, имеющемся в стандарте IEEE 829. Эти шаблоны полезны для того, чтобы удостовериться, что вы не забыли ничего важного. Однако легко попасть в ловушку, если использовать шаблон для плана тестирования так же, как форму для налоговой декларации или сдачи единого государственного экзамена. Шаблоны выступают в роли полезных напоминаний, но они не могут заменить размышлений – существенный элемент качественного планирования. Какой бы шаблон вы ни выбрали, убедитесь, что процесс не деградирует до "заполнения с наименьшими усилиями всех пустых мест в шаблоне", поскольку это редко дает в результате полезный план[2].

Планирование для аутсорсинга

В нашем примере Джамал намерен подрядить две независимые компании для проведения двух специальных видов тестирования в рамках определенных ограничений. Кстати говоря, в последнее время стало модно рассматривать аутсорсинг всего тестирования. Как это повлияет на план тестирования? Существует ряд проблем, усложняющих аутсорсинг тестирования. Чтобы успешно их решать, планы тестирования должны их учитывать.

Во-первых, тестирование при аутсорсинге может легко разойтись с реальными потребностями пользователей, выйти за пределы договоренностей в случае изменений, не завершиться в срок или превысить бюджет обычно без каких-либо уведомлений до тех пор, пока решение проблемы не станет невозможным. Разработка и тестирование при аутсорсинге, основанные на сказочных сроках, скупом бюджете и неоднозначных требованиях к продукту, с большой вероятностью обречены на провал. План тестирования должен предусматривать гарантии, позволяющие удерживать группу тестирования при аутсорсинге на верном пути в течение всего проекта.

Во-вторых, любые действия по управлению рисками, включая управление рисками качества, являются при оптимальном решении межфункциональными. В главе 2 были показаны различные методы управления рисками качества. Довольно необычно, если внешняя компания проводит анализ рисков качества с привлечением специалистов разного профиля. Как прави-

---

[1] См. раздел "Publications" на www.rexblackconsulting.com.

[2] Тест-менеджер Nolo.com Дебора Маккандлесс (Deborah McCandless) использовала следующую аналогию: «Я думаю, что [шаблоны] похожи на отменный наваристый бульон для супа. Он предоставляет мне отличную основу, из которой я готовлю замечательное, вкусное блюдо». С учетом моего расположения к кулинарным сравнениям, мне кажется, что данное сравнение безупречно.

ло, лучше, если в ходе фазы планирования этим процессом руководит тест-менеджер, менеджер разработки или менеджер проекта той компании, которая ведет разработку, особенно в части анализа рисков качества, а затем передает план субподрядчикам для проведения тестирования.

В-третьих, менеджеры высшего звена, наблюдая за процессом сверху, иногда бывают дезориентированы субподрядными организациями, предлагающими недорогие рабочие руки и сертифицированные процессы качества. Эти руководители затем принимают стратегические решения о значительных объемах аутсорсинга, упуская определенные критические детали, проявляющиеся на уровне тактики, когда линейные менеджеры пытаются выполнить свою часть работы.

С одной стороны, вложения в логистику и неизбежные задержки, необходимые для поддержки географически распределенных проектов любого вида, являются значительными и в некоторых случаях непредсказуемыми, пока не попытаться их осуществить. С другой стороны, наличие у выбранного партнера-субподрядчика собственных процессов вовсе не означает, что компания, *нанимающая* партнера-субподрядчика, будет обладать *этими* процессами. Когда две компании пытаются работать совместно, порядок встречается с хаосом, что выливается в серьезные проблемы. Это особенно верно в том случае, когда неупорядоченная компания рассчитывает, что ее партнеры по аутсорсингу смогут изящно отреагировать на ежедневные серьезные изменения в плане проекта, находясь на значительном удалении во времени и пространстве.

В-четвертых, тестирование требует большого разнообразия умений и навыков, как это будет показано в следующих двух главах. Некоторые из этих навыков относятся к тестированию, и специалистами с такими навыками обязана располагать любая уважающая себя компания по предоставлению услуг тестирования в рамках аутсорсинга. Другие относятся к знанию предметной области и базовой технологии реализации системы. Необходимо очень тщательно выбирать партнера-субподрядчика по тестированию, если группам тестирования требуются хорошие знания предметной области и технологии.

Таким образом, возможно и во многих случаях желательно передавать часть тестирования внешним компаниям в качестве дополнения к небольшой эффективной внутренней группе тестирования, которая в этом случае занимает лидирующее положение и обладает ключевыми умениями и навыками. Внешние группы могут использоваться в качестве дополнительных рабочих рук, когда наступает время выполнения тестов. (Это можно сделать также внутри компании, наняв более дешевых младших тестировщиков.) Внешние группы, возможно, также требуются в тех случаях, когда необходимо разобраться с реализацией тестирования в областях, где внутренней группе тестирования не хватает умений и навыков, особенно в тестировании производительности и безопасности. Наконец, для гарантии успеха проекта важен тщательный отбор партнеров по аутсорсингу тестирования[1].

---

[1] Тема отбора и использования ресурсов аутсорсинга тестирования рассмотрена в главе 10 моей книги "*Managing the Testing Process, Second Edition*". Для полноты информации я должен упомянуть, что моя консалтинговая компания осуществляет тестирование на основе аутсорсинга, а также предоставляет другие услуги по консалтингу и обучению, связанные с тестированием.

Если ключевые участники проекта не поддерживают план тестирования

В большинстве планов тестирования, которые я готовил, упоминаются ключевые действующие лица, которые не входят в группу тестирования. В главном примере книги используется весьма типичный состав участников проекта: сотрудники группы системной поддержки, программисты, инженеры сборки и менеджеры. Группа тестирования зависима от этих внешних участников, как правило, по двум причинам. Во-первых, эти люди могут обладать полномочиями, связанными с предоставлением продукта или услуги, которая необходима тестировщикам. Во-вторых, эти люди могут отвечать в рамках проекта за выполнение задач, предшествующих задачам тестирования. Другими словами, зависимости существуют либо по организационным причинам, либо по причинам, связанным с проектом.

В большей части случаев, как показывает мой опыт, более чем в 90% случаев, все ключевые участники, упомянутые в плане тестирования, либо сразу соглашаются с тем, что я прошу, либо мы приходим к некоторому компромиссу. Иногда тем не менее возникают проблемы. Когда мы подходим к ситуации, которая, по моему мнению, означает существование проблем, связанных с поддержкой, я применяю подход постепенной эскалации проблемы.

Я начинаю с рассмотрения возможности, что я очень даже мог заблуждаться, делая запрос. Почему я допустил, что данный ключевой участник должен выполнять обязанности, которые поручаются ему согласно моему плану? Возможно, я просто что-то перепутал. Я дважды проверяю мои предположения, прежде чем начать требовать от кого-либо что-то сделать.

Если подтверждается, что данный участник действительно должен выполнять определенные обязанности, я пытаюсь убедить его в этом. Я обращаюсь к нему один на один. Объясняю причины возникновения моих потребностей, а также почему сотрудники, занятые в проекте, не могут мне помочь. Затем я показываю значение любой задержки или приостановки его содействия проекту. Если я вижу позитивные выводы из обсуждения — человек однозначно подтверждает, что он окажет мне содействие и согласен с предлагаемыми обязанностями, — я четко документирую это в плане тестирования. Отправляя план тестирования на рецензирование, я включаю в список рассылки не только этого человека, но и его непосредственного начальника.

Важно проявлять гибкость в обсуждениях такого рода. Для достижения соглашения могут потребоваться поиски компромисса или разного рода взаимовыгодные предложения. Допустим, что Кейт Ли не мог согласиться настроить все системы в соответствии с обсуждаемым графиком. В этом случае Джамал мог предложить угостить обедом и ужином Индера и Петру, если бы они согласились потратить выходные, чтобы подготовить системы в срок. Или же можно было согласиться отложить подготовку некоторых систем на несколько дней, при условии, что ряд основных систем был бы уже готов к работе.

Можно придумать или использовать нетрадиционные методы переговоров. Однажды в Японии я участвовал в совещании рабочей группы проекта, которая работала с моим заказчиком по созданию новой системы для ноутбуков. Я представил план тестирования всей группе. Во время перерыва менеджер линии продуктов отозвал меня в сторонку и пригласил вечером на ужин. Во время ужина, выпив пива и сакэ с огромным количеством суши, он сообщил мне по секрету, что производственная группа не сможет предоставить все примеры разработки, о которых я просил в плане тестирования. Покончив с рыбой и пивом, мы набросали новый план примерного размещения прототипов разработки, который отвечал возможностям его группы и одновременно предоставлял моей группе тестирования элементы в достаточном количестве для выполнения наших обязательств, хотя и более медленно, чем планировалось первоначально.

Если я не могу получить необходимую поддержку такими способами, я выхожу на следующий уровень, т.е. я эскалирую проблему отсутствия соглашений вверх по управляющей цепочке в направлении к общим руководителям, будь то вице-президент или даже президент компании. Для успеха этого предприятия необходимо приложить все возможные политические и коммуникативные навыки. Если эскалация вопроса ведется неверно, это может привести к потере доверия и репутации. Предпочитаю решать такого рода вопросы в непосредственном диалоге один на один, а не с помощью массовых рассылок электронной почты и не на совещании.

Цель эскалации проблемы — убедить одного из менеджеров в необходимости поддержать вас. Однако опасайтесь пустых обещаний. Я встречал таких людей, которые под давлением соглашались что-то сделать, а потом прибегали к затягиванию с недобрым намерением "сравнять счет". Если вы полагаете, что сможете победить в дебатах за счет распоряжения руководства, которое фиксирует ваш запрос, а сотрудник, принуждаемый таким образом, действительно будет решать поставленную задачу, то этот уровень эскалации проблем сработает. В противном случае лучше перейти на следующий уровень решения проблем.

Последний уровень, по моему мнению, — это обращение к своему непосредственному руководителю с запросом ресурсов, необходимых для самостоятельного решения вопроса. Это тяжело, поскольку, вероятно, никто не будет рад этому запросу, и это несопоставимо с необходимостью управлять выполнением новой совокупности задач. Если задача чрезмерна для какой-либо другой группы, — а это почти наверняка так, поскольку на первом месте тот человек, которого вы уже просили об этом, — то потребуется продемонстрировать, что вы испробовали все возможные варианты заручиться его поддержкой. Также потребуется объяснить, почему существует зависимость, а это может оказаться в некоторых случаях трудной задачей.

Некоторые считают процесс эскалации проблем конфронтационным и наносящим ущерб сплоченности команды, поэтому они молча переносят дополнительные объемы работ, сваливающиеся на группу. Не считаю этот подход верным. Однажды одному заказчику я помогал готовить план

тестирования, выстраивать процесс тестирования и разрабатывать систему тестов, а затем передал план тест-менеджеру и ее группе. В плане тестирования я описал процесс подготовки версий, за который отвечала группа программирования. На деле же программисты оказались неспособны выполнить обязанности по подготовке тестовых версий. Если тест-менеджеру нужна была тестовая версия, ей приходилось разбираться с этим самостоятельно.

Вместо того чтобы работать согласно процессу, описанному мной, она поручила работу по сборке версий кому-то из своей группы. Это существенно снизило производительность группы тестирования и уменьшило ее возможности по выполнению всех запланированных тестов в каждом из тестовых циклов. В результате у руководства появились различные вопросы к ней и возникло впечатление неэффективности группы тестирования и неисполнительности.

На ее месте я бы вежливо, но твердо обсудил проблему с менеджером программистов и менеджером проекта (который был общим начальником менеджера программистов и тест-менеджера). Я бы выразил готовность взять на себя эту обязанность, если это необходимо, но при условии, что либо будут уменьшены рамки тестирования, либо будет увеличено число сотрудников группы тестирования. Возможно, в список также были бы включены обучение и консультации, если бы я принял на себя дополнительные обязанности.

Как уже отмечалось в главе 5, особой проблемой является наличие ключевых участников проекта тестирования, которые не являются заинтересованными лицами проекта. Если их интересы не являются сравнительно нейтральными, а действительно направлены против успеха проекта, то вы, конечно, выступаете против этого. Старайтесь избегать таких зависимостей. Если это невозможно, запаситесь временем для решения этих проблем с помощью процесса эскалации либо каким-либо другим способом на ваше усмотрение.

Некоторые проблемы поддержки связаны с отношением к делу, другие возникают из-за некомпетентности. В рассматриваемом примере Джамал тщательно определяет перечень сотрудников, чтобы выявить проблемы компетентности заранее. В ходе обсуждения поддержки Кейт, в сущности, назначил испытательный срок Индеру из-за ранее возникших проблем с его квалификацией.

Однажды я работал в проекте, где мне не удалось конкретно определить в плане тестирования, кто является сотрудниками группы поддержки. Менеджер сетевых операций выделил человека, который, хотя и в полной мере *желал* работать в качестве администратора UNIX, но знал об этом меньше, чем кто-либо в группе тестирования. Половину времени, когда его просили о поддержке, он демонстрировал неспособность диагностировать проблему. Вторую половину времени он, как правило, реализовывал неверное решение, что выливалось в бесполезную трату усилий группы тестирования при работе с неверно конфигурированными серверами. Только компетентные и опытные профессионалы должны заниматься поддержкой тестовой среды.

Независимо от того, обусловлены ли проблемы отношением к делу или компетентностью, они могут проявиться либо на фазе планирования, что предпочтительнее, либо позднее, во время его осуществления. Одной из целей планирования является распознавание и разрешение (или, по крайней мере, смягчение) этих проблем до начала выполнения тестов.

Мой опыт показывает, что успех разрешения проблем, связанных с внешними ключевыми действующими лицами зависит от двух факторов. Во-первых, я использую рычаги для укрепления взаимоотношений с другими менеджерами и коллегами. Это важно не только для получения поддержки, но и для понимания, насколько надежным является их отношение к поддержке. Что касается меня, я убежден, что доверие между профессионалами растет по мере того, как мы даем обещания друг другу и затем выполняем их. Если человек, не подводивший меня на протяжении 6–7 проектов, говорит, что он это сделает, я верю ему. В то же время человек, с которым я никогда не работал, должен иметь хорошие рекомендации.

Во-вторых, я не только стремлюсь занимать разумную позицию, но стараюсь также тщательно управлять и добиваться понимания разумности моей позиции. Если меня станут воспринимать как истерическую личность, преувеличивающую опасность ситуации или устраняющуюся от процесса, я нанесу непоправимый ущерб доверию и взаимоотношениям с моими коллегами. Специалист-тестировщик может быть эффективным только в том случае, когда другие люди прислушиваются к результатам его поисков и доверяют тому, что он сообщает им[1].

Подчинение закону и следование правилам

В качестве последней проблемы отметим, что в некоторых случаях план тестирования может случайно и весьма неприятно столкнуться с законом, если быть невнимательным. В одном из проектов мы собирались тестировать сквозной процесс установки и сопровождения версии для программы, написанной в США, установленной на аппаратной платформе в Тайване, а затем поставленной непосредственно заказчикам снова в США. Было только одно препятствие: программа содержала в браузере технологию шифрования, которая классифицировалась согласно американским законам как средство вооружения. Экспортировать это "средство вооружения" без разрешения Министерства торговли является серьезным преступлением, которое наказывалось в то время пятью годами заключения и штрафами, связанными с признанием виновным.

Существует множество законов, регулирующих перемещение технологий через границы либо внутри страны. Например, некоторые европейские страны регулируют технологии шифрования и запрещают частное

---

[1] По общей теме поддержки со стороны ключевых действующих лиц можно также посмотреть презентацию Ли Копленда (Lee Copeland) "When Helping Doesn't Help" на www.stickyminds.com.

владение такими программами. Перечень непристойных материалов может быть увеличен в некоторых странах или юрисдикциях. Настоящие законы США запрещают публикацию технологий, которые обходят определенные схемы предупреждения копирования, например, для DVD.

Существуют ситуации, когда тестировщик, пытаясь проверить продукт перед поставкой заказчикам, может непреднамеренно совершить серьезное преступление, способное привести к неприятным последствиям для его последующей жизни и даже к окончанию карьеры. Если вас одолевают сомнения, проконсультируйтесь с юрисконсультом компании, знакомым с содержанием продукта и законами, которые регулируют владение им, его передачу и использование.

В некоторых случаях тестирование является также предметом регулирования властей. В США Управление по санитарному надзору за качеством пищевых продуктов и медикаментов устанавливает порядок тестирования медицинского оборудования. Стандарты Министерства обороны могут применяться к контрактам с Вооруженными силами США. Во многих других странах есть аналогичные механизмы регулирования. Подход к тестированию должен быть признан имеющим силу, что часто требует подготовки документа о том, что тестирование проводится в соответствии с определенными стандартами. В некоторых случаях необходимо утвердить документы в регулирующих органах. Если это так, нужно помнить, что люди, утверждающие документы, не являются заинтересованными лицами проекта. Они стоят на страже общественных интересов, здоровья и безопасности и проводят законы в жизнь. Они в меньшей степени заботятся о ситуации с бюджетом и сроками проекта, в основном уделяя внимание выполнению своих обязанностей перед налогоплательщиками, законом и исполнительными органами правительства.

Внедрение усовершенствований

Что конкретно нужно сделать для улучшения процесса планирования, зависит от текущего состояния этого процесса. На верхнем уровне детализации может помочь следующий подход.

1. Понять цель процесса планирования. Зачем вам нужен план? Для того, чтобы написать документ? Является ли план документом для внутреннего или внешнего употребления? Является ли процесс планирования больше неформальным, чем формальным? Нужен ли вообще документированный план?
2. Понять контекст тестирования, плана и процесса планирования. Кто является пользователем документированного плана? Какие лица являются заинтересованными в процессе планирования? Кто является ключевыми действующими лицами процесса тестирования?
3. Если план должен быть документирован, примите решение, как создавать документ. Какой формат или шаблон является подходящим? Должны ли соблюдаться стандарты документирования?

Если нет, какую информацию необходимо собрать и зафиксировать?

4. Начните с простого. Не надо заботиться о планировании всех возможных непредвиденных ситуаций и событий. Работайте с заинтересованными лицами для создания хорошего плана, который позволит разрешать наиболее вероятные проблемы. Не пишите слишком много, а сосредоточьтесь на подготовке четких и ясных формулировок того, что вы собираетесь делать.
5. В процессе планирования, особенно на первых порах, ищите как можно больше вариантов консенсуса, по возможности при личных встречах. Тест-менеджер, который в одиночку готовит план тестирования, а затем рассылает его как директиву, вряд ли добьется результата. Специалист-тестировщик, работающий непосредственно с заинтересованными лицами и ключевыми участниками, подготавливая совместный и гармонично сочетающийся с контекстом процесс тестирования, с большой вероятностью разработает план качественного подпроекта тестирования.

Применяя этот подход для улучшения процесса планирования тестирования, можно обновлять методы и стиль документирования постепенно на протяжении нескольких проектов. Я разрабатывал свой подход к планированию тестирования и документированию планов тестирования в течение 15 лет работы в качестве тест-менеджера проектов. Вы можете опереться на знания, которыми я поделился с вами в последних двух главах, так что вам не потребуется 15 лет на обучение, однако нужно понимать, что процесс планирования должен улучшаться итеративно и постепенно. Обычно отличными возможностями для этого как на уровне проекта, так и в рамках самой группы тестирования, являются ранее выполненные проекты[1].

Окончание этой главы также означает конец первой части книги. Мы рассмотрели четыре процесса, которые наметили курс на успешное проведение тестирования. Выделение времени и внимание к части процесса тестирования, связанной с планированием, для запуска проекта тестирования в верном направлении, обычно приносит дивиденды позднее в ходе выполнения проекта тестирования. В моих предыдущих проектах многое из того, что потом шло верно – предотвращение катастроф, улучшение командной работы, эффективное и производительное тестирование, – начиналось на этапе планирования процесса тестирования. Как тест-менеджер я давно верю в старую поговорку о том, что неудачно подготовленный план – это план неудачного проекта.

От понимания того, что делать, мы переходим к непосредственному осуществлению планов. Следующий этап процесса тестирования застанет нас за подготовкой системы тестов и группы тестирования. Как и наличие хорошего плана и хороших процессов, наличие надлежащим образом подготовленных и верно выбранных людей и инструментов является

---

[1] Я благодарю Криса Денардиса (Chris DeNardis), одного из рецезентов, за упоминание этого факта в одном из писем.

критическим для успеха следующих этапов, когда мы будем выполнять тесты и улучшать систему, предназначенную для тестирования, и для самого тестирования.

№ этапа	Этап	Выполнено?
1.	Планирование: понимание места тестирования.	☑
1.A	Определить функциональный (система, проекты и процесс) и организационный контексты, в которых выполняется тестирование.	☑
1.B	Определить и расставить приоритеты рисков качества системы, достичь консенсуса заинтересованных сторон проекта относительно возможностей тестирования по смягчению этих рисков.	☑
1.C	Оценить временные, ресурсные и финансовые затраты, необходимые для выполнения тестирования и согласованные с пунктом 1.B, и получить поддержку оценки у руководства.	☑
1.D	Запланировать задачи, зависимости и состав участников, которые необходимы для смягчения рисков качества системы, и получить поддержку плана у заинтересованных сторон проекта.	☑

Часть II

Подготовка

Во второй части книги мы изучим процессы, относящиеся к формированию эффективной и производительной группы тестировщиков и эффективной и производительной системы тестов. Так же, как у нас должно быть две руки, чтобы хлопать в ладоши, у нас должны быть и группа тестирования, и система тестов. Начнем с людей, поскольку именно инженеры-тестировщики, младшие тестировщики и другие специалисты по тестированию будут разрабатывать систему тестов. Затем мы рассмотрим планирование, создание и сопровождение самой системы тестов. Как только мы разберемся в хитросплетениях создания группы тестирования и системы тестов, мы сможем перейти к третьей части книги, где обсуждается выполнение тестов.

№ этапа	Этап	Выполнено?
2.	Подготовка: собрать людей и подготовить тесты.	☐
2.A	Посредством найма и обучения создать команду специалистов-тестировщиков, обладающих соответствующими умениями, навыками, отношением к делу и мотивацией.	☐
2.D	Спроектировать, разработать, получить и проверить систему тестов, которую группа тестирования будет применять для проверки качества системы, предназначенной для тестирования.	☐

ГЛАВА 8

Поиск искусных тестировщиков: как и кого нанимать

Мы переходим от планирования к тем внутренним процессам, которые существенно влияют на нашу подготовку к выполнению тестов. Для того чтобы группа тестирования могла приступить к этим процессам, она должна быть сформирована из тщательно подобранных индивидуальностей. Отличные группы тестирования добиваются выдающихся достижений, а посредственные группы затевают распри с группой разработки и тормозят успех проекта. Выдающиеся группы тестирования образуют вокруг хороших менеджеров, а хорошие менеджеры способствуют развитию карьеры классных специалистов-тестировщиков в этих группах.

Наличие тщательно подобранных специалистов в группе имеет большое значение для тестирования. В самом деле, незрелые группы тестирования со слабо поставленными процессами иногда добиваются успеха только потому, что была подобрана подходящая группа тестировщиков. Если грамотно поставлены процессы, отбор подходящих сотрудников закладывает основу успешности группы тестирования. Только команда, составленная из квалифицированных специалистов, может добиться успеха в ключевых процессах тестирования, которые рассматриваются в этой книге.

Процесс создания команды состоит из двух основных активностей. Во-первых, мы должны нанять подходящих людей, имеющих навыки, знания и отношение к делу, которые дают им возможность выполнить работу. Во-вторых, с течением времени мы должны способствовать повышению квалификации команды. Повышение квалификации должно быть направлено как на более качественное выполнение тех обязанностей, с которыми уже справляется группа, так и на подготовку к решению новых задач.

Процесс создания команды существенно зависит от контекста и требует совместных усилий. В некоторых компаниях при приеме на работу тестировщики проходят те же начальные испытания, что и программисты. В других компаниях специальные этапы входного контроля для кандида-

Тестировщик, специалист-тестировщик (Tester, Test Professional)

Лицо, выполняющее задачи, относящиеся к тестированию. Человек, работающий в группе тестирования постоянно или временно в качестве сотрудника, контрактника или консультанта. Я использую этот термин как аналог понятия "разработчик" или "программист" в группе разработки.

та, стремящегося занять вакансию, определяют отделы по работе с персоналом. Обучение и другие программы повышения квалификации значительно отличаются в разных компаниях, и в некоторых компаниях стандартизованы. Не пытаясь охватить весь спектр проблем управления персоналом, я остановлюсь на тех его элементах, которые влияют на тестирование.

При найме на работу мы должны ответить на вопрос: "Способен ли этот человек помочь нам проверить качество программного продукта?" Этот вопрос отличается от вопросов: "Может ли этот человек писать код?" или "Понимает ли этот человек бизнес-задачи, которые решает система?", – хотя квалифицированный тестировщик часто должен и иметь технические знания, и разбираться в предметной области.

При обсуждении повышения квалификации мы должны ответить на вопрос: "В каком направлении развивается карьера каждого из сотрудников группы тестирования, какие навыки необходимо иметь для движения по этому пути и насколько они соответствуют потребностям группы тестирования?" Такой вопрос применим и к другим группам тестирования. Однако в каждом конкретном случае ответ будет уникальным, поскольку ключевые навыки тестирования уникальны для каждого проекта.

Процесс создания команды

Независимо от того, как определен процесс создания команды в вашей компании, можно, не боясь ошибиться, предположить, что желаемым результатом является достижение практических договоренностей с наиболее подходящей и компетентной кандидатурой на занятие любой вакансии в компании. Эта высокая и очевидная цель часто бывает подчинена тактическим требованиям, в частности обеспечению соответствия трудовому законодательству, достижению справедливости в оплате труда и движении по карьерной лестнице, а также соответствию местному рынку труда.

Поскольку процесс создания команды управляется людьми, он может также быть подвержен менее рациональным, но столь же решающим факторам. Например, люди в компании, которые могут влиять на кадровые решения или вовлечены в принятие этих решений, часто имеют предвзятое мнение и опираются на прошлый опыт. Другим примером являются распоряжения по различным кадровым вопросам, которые часто появляются по прихоти руководства или как реакция на некоторые внешние со-

бытия или слухи. В целом, выбор коллег по работе подвержен влиянию тех же неуловимых психологических факторов, которые влияют на выбор друзей, супруги(а) или клубов.

Находясь в центре этого водоворота совокупности факторов, процесс создания команды в большинстве средних и больших компаний определяется отделом по работе с персоналом. Этот процесс, возможно, похож на Процесс 6. В этой главе мы рассмотрим этап 1, а в следующей главе – этап 2.

Процесс 6. Процесс построения команды

№ этапа	Этап	Выполнено?
1.	Нанять подходящих сотрудников группы тестирования.	☐
1.A	Получить разрешение на поиск кандидатов.	☐
1.B	Определить параметры вакансии и подготовить рекламное объявление.	☐
1.C	Найти и подвергнуть испытаниям кандидатов на основе резюме, телефонных интервью; отсеять неквалифицированных и нежелательных кандидатов.	☐
1.D	Провести личные интервью с квалифицированными и желательными кандидатами.	☐
1.E	Если есть подходящие кандидаты, сделать предложение наиболее успешному из них, как правило, направив ему соответствующее письмо.	☐
1.F	Если наиболее успешный кандидат принимает предложение, принять нового сотрудника на работу. Если нет, то либо повторить этапы 1.E и 1.F для каждого следующего по успешности кандидата, пока успешный кандидат не примет предложение, либо, если ни один из желательных кандидатов не принял предложение, начать процесс заново с этапа 1.A. Уведомить отвергнутых кандидатов, что они должны искать другие варианты.	☐
2.	Стимулировать повышение квалификации и карьерный рост сотрудников группы.	☐
2.A	Совместно с новыми сотрудниками разработать пути развития их карьеры.	☐
2.B	На регулярной основе проверять пути развития карьеры всех сотрудников и отмечать прогресс каждого.	☐
2.C	Активно управлять процессом овладения сотрудниками навыками, которые требуются для достижения целей как сотрудника, так и всей группы тестирования.	☐
3.	Повторить этап 1, если нужно найти новых людей. Непрерывно повторять этап 2.	☐

Письмо с предложением о занятии вакантного места (Offer Letter)

Письмо из компании успешному кандидату с предложением занять вакантное место, включая детальное описание должностных обязанностей, указание фактического размера заработной платы и даты начала работы.

Новый сотрудник (New Hire)

Лицо, получившее письмо с предложением о занятии вакантного места и ответившее на него устно или письменно согласием занять предложенную вакансию.

Отдел по работе с персоналом (Human Resources Department)

Группа, отвечающая за 1) поиск сотрудников, контрактников и консультантов; 2) выработку политики и процедур управления персоналом; 3) планирование и управление изменениями в заработной плате и карьерном росте сотрудников.

Разновидности процесса создания команды

Только что представленный процесс является типичным для создания группы тестирования, однако существует много его разновидностей. Выбор варианта зависит не только от предпочтений компании и законодательства о труде, но и от размера компании, уровня сотрудника и типичного направления развития карьеры в компании.

Например, в некоторых компаниях сотрудникам низшего звена уделяют очень мало внимания, особенно если они изначально были наняты по контракту или временно. Когда я поступал на работу в большой университет программистом UNIX, процесс найма состоял из разговора с двумя профессорами о должностных обязанностях, опубликованных на доске объявлений в кампусе. Посовещавшись, они предложили мне работу. Ввод в курс дел состоял для меня в получасовой беседе с одним из дипломников этих профессоров о содержании проекта, в котором я должен был работать. Справедливости ради надо сказать, что позднее мой статус сотрудника университета рассматривался более серьезно. Я должен был пройти через формальный процесс, включая заполнение заявления о приеме, несмотря на то, что я уже давно работал там!

Сравните это с вооруженными силами США, которые тратят огромные усилия на процессы найма, обучения и карьерного роста солдат и офицеров. Почему такая разница? В университете было невозможно представить, что в один прекрасный день я стану деканом, ректором или членом совета университета. Все считали, что я потрачу год-два на кодирование на C и затем уйду, что и произошло.

В профессиональной армии сегодняшний рядовой может запросто стать сержантом, который будет отвечать за жизни и безопасность десятков людей, сегодняшний курсант военно-морского училища может однаж-

ды стать адмиралом или командиром атомной подводной лодки, оснащенной ракетами Трайдент, заняв вакансию, на которой человек может реально влиять на ход истории. Армия способствует исключительно внутреннему карьерному росту. Таким образом, отличные решения по найму на низшем уровне в сочетании с непрерывной тренировкой лидерских качеств и других умений и навыков ведут к превосходному мастерству высших чинов, которые стоят на страже мира и умеют при необходимости воевать.

Ничего столь значительного, как судьба мира, не руководит вашими решениями при формировании группы тестирования, но умение окружить себя талантами определяет успех тестирования в проекте, ощущение компетентности группы тестирования и карьерные перспективы для каждого тестировщика. По всей вероятности, ваши возможности по существу влиять на этот процесс, организованный согласно приведенному выше описанию либо иначе, ограничены. Тем не менее в рамках определенного процесса создания команды вам придется принимать решения и выполнять действия, имеющие ключевые последствия для группы тестирования. Как вы принимаете эти решения и что вы делаете — более важно, чем сам процесс[1].

Увеличение группы тестирования проекта "Суматра"

Даже при составлении плана тестирования Джамал Браун занимался поиском людей для группы тестирования. В соответствии с бюджетом, утвержденным его непосредственным начальником Джоном Албертсоном, Джамал мог нанять одного инженера-тестировщика на постоянную работу и четырех младших тестировщиков по контракту. Особенно срочно он хотел найти старшего инженера по автоматизированному тестированию, которому поручалось готовить и выполнять регрессионные тесты для новой функциональности Суматры, а также тестировать надежность и стабильность. Он знал, что этого человека будет трудно найти.

№ этапа	Этап	Выполнено?
1.A	Получить разрешение на поиск кандидатов.	☑

Для того чтобы провести анализ наличия умений и навыков группы тестирования на настоящий момент Джамал начал с создания ядра группы тестирования проекта "Суматра". В него вошли Эмма Мурхаус, уже работающий инженер по автоматизированному тестированию, и Лин Цу

[1] Дополнительные идеи по найму персонала я рекомендую поискать в учебнике Джоанны Ротман (Johanna Rothman) "Hiring Technical People: A Guide to Hiring the Right People for the Job", который есть на ее сайте www.jrothman.com. Еще одна неплохая книга — "*The Accidental Project Manager*" Патрисии Энсворт (Patricia Ensworth).

Ву, инженер по ручному тестированию. На основе проведенного анализа все трое пришли к выводу, что должностные обязанности описаны в общем виде, но их суть отражена (см. рис. 8.1). Джамал отправил перечень должностных обязанностей для кандидатов на вакансии в различные группы новостей Интернета и в раздел вакансий рекламного отдела местной газеты.

№ этапа	Этап	Выполнено?
1.B	Определить параметры вакансии и подготовить рекламное объявление.	☑

В ответ на рекламное объявление Джамал получил несколько резюме по электронной почте и факсу. Получая резюме, он сопоставлял навыки каждого из кандидатов с результатами анализа навыков группы тестирования (см. рис. 8.2). В определенной степени это были гипотезы, и Джамал мог только предполагать на основе количества времени, проведенного в работе над проектами, требовавшими этих навыков, какой уровень знаний имеет или использовал каждый из кандидатов. Однако это помогло ему отсеять кандидатов, которые со всей очевидностью не соответствовали минимальным требованиям.

Инженер автоматизированного тестирования

Обязанности

- Участвовать в процессе выбора инструментов автоматизации тестирования.
- Разрабатывать и выполнять автоматизированные тесты, используя выбранный инструмент.
- Интегрировать комплекты тестов в систему управления тестами и в заказные тестовые драйверы.

Образование

- Бакалавр компьютерных наук, вычислительной техники или электротехники, либо наличие опыта работы не менее двух лет, либо сертификаты тестировщика.

Опыт работы

- От пяти лет опыта работы по автоматизации тестирования.
- Требуется опыт тестирования UNIX-систем, особым плюсом является опыт разработки заказных систем автоматизированного тестирования, включая знание языков разработки скриптов ksh, csh и tcl.
- Желательно наличие опыта тестирования Интернет-приложений, включая разработку автоматизированных инструментов тестирования.

Требуемая квалификация

- Предпочтительно наличие подтвержденных карьерных устремлений, связанных с тестированием или контролем качества. Кандидат должен продемонстрировать знакомство с теорией и практикой тестирования и способность объяснить, как он применял эти знания на предыдущем месте работы.

Прочее

- Форма одежды деловая, свободная. Рабочий день ненормированный. Иногда нужно будет работать в выходные и в вечернее время.

Рис. 8.1. Описание должностных обязанностей инженера по автоматизированному тестированию

Группа тестирования компании Software Cafeteria
Таблица оценки умений и навыков

	Условные обозначения	0 = знания отсутствуют		1 = знания незначительны		R = требуется	
		2 = хорошие знания		3 = эксперт		D = желательно	
		TM = тест-менеджер		MTE = инженер по ручному тестированию		ATE = инженер по автоматизированному тестированию	
Навыки и квалификация	ATE, минимум	MTE, минимум	TM, Джамал Браун	MTE, Лин Цу Ву	ATE, Эмма Мурхаус	Минимум для группы	Среднее значение для группы
Общая квалификация							
Образование							
Степень бакалавра наук (BS) (или выше)	D	D	BS (компьютерные науки или выч. техника)	Доктор философии (компьютерные науки)	BS (Бизнес)		
Обучение тестированию или наличие сертификата	D	D	Контроль качества программного обеспечения	ISEB			
Прочее					Дипломированный бухгалтер		
Опыт работы (лет)							
Должности, связанные с тестированием	5R	5R	7	5	12		
ИТ (не тестирование)	D	D	3	2	4		
Предметная область (не ИТ)	D	D					
Не предметная область (не ИТ)					6		
Итого/Все/Прочее	5R	5R	10	7	22		
Профессиональные качества							
Владение устной речью	2R	2R	3	1	2	1	2.0
Навыки неформальной письменной речи	3R	3R	3	3	3	3	3.0
Навыки подготовки формализованных документов	D	D	3	0	1	0	1.3
Непрерывное обучение	R	R	Да	Да	Да	Да	Да
Участие в создании группы тестирования/взаимное обучение	2R	2R	3	2	2	2	2.3
Налаживание взаимоотношений с группами другой функциональной направленности	2R	2R	3	2	2	2	2.3
Чтение (запоминание, способность к рассуждениям и анализу)	2R	2R	3	2	2	2	2.3
Понимание тенденции развития отрасли и технологий (чтение профессиональных журналов)	1R	1R	3	1	1	1	1.7
Навыки тестирования							
Общие навыки							

Рис. 8.2. Анализ умений и навыков группы тестирования

Условные обозначения	0 = знания отсутствуют		1 = знания незначительны		R = требуется	
	2 = хорошие знания		3 = эксперт		D = желательно	
	ТМ = тест-менеджер		МТЕ = инженер по ручному тестированию		АТЕ = инженер по автоматизированному тестированию	

Навыки и квалификация	АТЕ, минимум	МТЕ, минимум	ТМ, Джамал Браун	МТЕ, Лин Цу Ву	АТЕ, Эмма Мурхаус	Минимум для группы	Среднее значение для группы
Стандарты тестирования	2R	2R	3	3	3	3	3.0
Жизненные циклы разработки программных продуктов	2R	2R	3	3	2	2	2.7
Процессы тестирования/разработки и их зрелость	1R	1R	2	2	1	1	1.7
Управление изменениями	1R	1R	2	2	1	1	1.7
Связь тестирования и бизнеса/жизненные циклы разработки ПО	1R	1R	3	3	1	1	2.3
<u>Планирование</u>							
Оценка затрат	D	D	3	1	1	1	1.7
Подготовка документации	D	D	3	1	1	1	1.7
Стоимость качества	D	D	3	2	1	1	2.0
Риски качества/анализ видов ошибок и их влияния	D	D	3	2	1	1	2.0
Анализ и управление рисками качества	D	D	3	2	1	1	2.0
<u>Проектирование и разработка тестов</u>							
Поведенческое тестирование (черный ящик)	2R	2R	2	3	2	2	2.3
Структурное тестирование (белый ящик)	D	1R	1	3	2	1	2.0
Статическое тестирование (требования, спецификации, документация)	D	2R	2	3	2	2	2.3
Тестирование надежности (статистика)	2R	D	1	1	2	1	1.3
Тестирование производительности	2R	D	1	2	3	1	2.0
Покрытие кода/потоков данных	2R	2R	2	3	3	2	2.7
Покрытие рисков качества/требований (возможность отслеживания)	1R	2R	3	2	1	1	2.0
<u>Автоматизация (разработка)</u>							
Выполнение тестов на готовых инструментах (Silk, Validor и т.п.)	3R	D	1	1	3	1	1.7
Управление тестами с помощью готовых инструментов	D	D	3	1	1	1	1.7
Разработка собственных инструментов	3R	D	3	1	3	1	2.3
<u>Конфигурация</u>							
Генераторы тестовых данных	1R	D	1	1	2	1	1.3
Версионный контроль	1R	1R	2	2	3	2	2.3
Управление конфигурациями	D	1R	1	2	1	1	1.3
Комплексное тестирование	D	1R	3	2	1	1	2.0

Рис. 8.2 (*продолжение*)

Условные обозначения	0 = знания отсутствуют		1 = знания незначительны		R = требуется	
	2 = хорошие знания		3 = эксперт		D = желательно	
	TM = тест-менеджер		MTE = инженер по ручному тестированию		ATE = инженер по автоматизированному тестированию	

Навыки и квалификация	ATE, минимум	MTE, минимум	TM, Джамал Браун	MTE, Лин Цу Ву	ATE, Эмма Мурхаус	Минимум для группы	Среднее значение для группы
Выполнение тестов							
Ручное (скрипты)	D	3R	3	3	1	1	2.3
Ручное исследовательское	D	3R	3	3	1	1	2.3
Автоматизированное	3R	D	1	1	3	1	1.7
Локализация дефектов	3R	3R	3	3	3	3	3.0
Подготовка отчетов о дефектах	3R	3R	3	3	3	3	3.0
Подготовка отчетов о ходе тестирования	2R	2R	3	2	2	2	2.3
Измерения тестирования (инструментальная панель)	1R	1R	3	2	3	2	2.7
Навыки тестирования (в среднем)			2.4	2.1	1.9	1.4	2.1
Знание предметной области							
Электронная обработка текстов							
Приложения Windows	1R	2R	3	3	3	3	3.0
Приложения UNIX	D	D	1	1	2	1	1.3
Приложения Macintosh	D	D	0	1	3	0	1.3
Графики и рисунки	1R	2R	2	2	1	1	1.7
Таблицы	D	1R	1	2	1	1	1.3
Математические и инженерные формулы	D	1R	1	3	0	0	1.3
Управление документооборотом							
Приложения Windows	D	D	1	2	2	1	1.7
Приложения UNIX	D	D	0	0	1	0	0.3
Приложения Macintosh	D	D	0	0	1	0	0.3
Прочее	D	D	0	0	1	0	0.3
Управление иерархическими хранилищами	D	D	1	2	3	1	2.0
Обмен документами							
Приложения Windows	D	D	1	2	3	1	2.0
Приложения UNIX	D	D	1	0	3	0	1.3
Приложения Macintosh	D	D	0	0	3	0	1.0
Печать							
Цветная	D	D	0	0	2	0	0.7
Лазерная	D	D	1	1	2	1	1.3
Струйная	D	D	1	1	2	1	1.3

Рис. 8.2 (*продолжение*)

Условные обозначения	0 = знания отсутствуют		1 = знания незначительны		R = требуется	
	2 = хорошие знания		3 = эксперт		D = желательно	
	TM = тест-менеджер		MTE = инженер по ручному тестированию		ATE = инженер по автоматизированному тестированию	

Навыки и квалификация	ATE, минимум	MTE, минимум	TM, Джамал Браун	MTE, Лин Цу Ву	ATE, Эмма Мурхаус	Минимум для группы	Среднее значение для группы
Печать/переплет	D	D	0	0	2	0	0.7
Публикация в Web							
HTML	D	D	1	3	3	1	2.3
XML	D	D	1	3	2	1	2.0
Прочее	D	D	1	1	2	1	1.3
Знание предметной области (в среднем)			0.8	1.3	2.0	0.7	1.4
Технические знания							
Программирование							
C/VB (3GL)	1R	D	2	2	3	2	2.3
Java/(C++) (Объектно-ориентированные)	1R	D	0	1	2	0	1.0
Скрипты shell (Tcl/Ksh)	2R	D	3	2	2	2	2.3
Сложность и измерения кода	1R	D	2	0	2	0	1.3
Операционные системы							
Windows	1R	1R	2	3	2	2	2.3
Linux	1R	1R	2	2	2	2	2.0
Solaris	1R	1R	2	1	2	1	1.7
Mac OS	D	D	0	2	3	0	1.7
Прочее	D	D	1	1	3	1	1.7
Сети/межсетевое взаимодействие							
Интернет-протоколы TCP/IP, FTP, RCP	1R	1R	2	1	1	1	1.3
Браузеры (Netscape, IE и т.п.)	1R	1R	2	3	1	1	2.0
Архитектура сетевых приложений (ярусная)	1R	1R	2	3	1	1	2.0
Аппаратное обеспечение сетей	1R	1R	2	3	1	1	2.0
Системы и серверы							
Web-серверы на Java	1R	1R	1	3	3	1	2.3
Серверы баз данных	1R	1R	2	3	3	2	2.7
Универсальные машины (mainframe)	1R	1R	1	2	3	1	2.0
Технические знания (в среднем)			1.6	2.0	2.1	1.1	1.9

Рис. 8.2 (*продолжение*)

№ этапа	Этап	Выполнено?
1.C	Найти и подвергнуть испытаниям кандидатов на основе резюме, телефонных интервью; отсеять неквалифицированных и нежелательных кандидатов.	☑

С кандидатами, которые соответствуют требованиям, Джамал, Эмма и Лин Цу провели телефонные интервью, чтобы уточнить оценку навыков каждого из кандидатов. Джамал обратил особое внимание на вилку должностного оклада, а также на путь развития карьеры, который может предложить успешному кандидату компания Software Cafeteria.

Произошел дополнительный отсев, в результате которого в следующий раунд процесса найма вышли три кандидата. Джамал запланировал личное интервью с каждым из них. Обсудив планы с менеджером по работе с персоналом, он решил, что кандидаты проведут половину рабочего дня, отведенного на интервью, следующим образом:

1. Рассказ менеджера по работе с персоналом Боба Френкеля о компании (0.5 часа).
2. Интервью один на один с Джамалом (1 час).
3. Групповое интервью за обедом с участием Джамала, Эммы и Лин Цу (1 час).
4. Выполнение контрольных заданий (1 час).
5. Встреча с менеджером разработки Дженни Кауфман и менеджером проекта Кейт Эрнандес (0.5 часа).
6. Краткое подведение итогов с Бобом Френкелем.

Боб Френкель позвонил каждому из кандидатов и определил дату и время встреч.

Два интервью прошли самым обычным образом. Кандидаты казались достаточно компетентными, но одного из них в ходе интервью постоянно отвлекали звонящий сотовый телефон и какие-то личные дела, связанные со списком покупок в продуктовом магазине. Его вопросы в основном были связаны с тем, чтобы от него не требовалось работать более 45 часов в неделю. Постоянно возникали замечания о "необходимости иметь возможность получать удовольствие от жизни, а не только работать". Джамал понимал, что эти соображения справедливы, особенно в мире высоких технологий и запуска новых продуктов, но даже после того как он сообщил о своей приверженности к ограничению переработок, кандидат продолжал намекать, что данная вакансия может потребовать большой загрузки и работы в выходные. Другая кандидатка, хотя и более деликатная, дважды упомянула, что она не в восторге от предложенной зарплаты, и спросила, есть ли какая-либо "возможность для переговоров" по этой цифре.

Выполнение контрольных заданий (Audition Interview)

Часть процесса интервьюирования при приеме на работу, которая посвящена показу умений и навыков, необходимых для занятия вакансии, например написанию тестового сценария или подготовке отчета об ошибке.

Однако третье интервью с кандидатом, резюме которого показано на рис. 8.3, прошло очень хорошо. Динеш, очевидно, обладал необходимыми навыками, а также продемонстрировал положительное отношение к вакансии. Динеш сказал Джамалу и Эмме, что с ростом его семьи ему пора заканчивать работать по контрактам, с короткими проектами, неопределенностью будущего и ненадежной медицинской страховкой. Он выразил желание устроиться на долгосрочную работу в стабильной компании, что смягчило беспокойство Джамала по поводу того, что Динеш часто менял работу, будучи контрактником.

В качестве контрольного задания Джамал и Эмма попросили Динеша выполнить один из автоматизированных тестов. Он с этим успешно справился, заметив не только ошибку в продукте, но и подозрительное поведение самих автоматизированных скриптов. Лин Цу проверила отчет об ошибке, написанный Динешем, и сочла его проникающим в суть, лаконичным и полным.

№ этапа	Этап	Выполнено?
1.D	Провести личные интервью с квалифицированными и желательными кандидатами.	☑

Поскольку обсуждение, которое провел Джамал с Эммой, Лин Цу и Бобом Френкелем по результатам интервью, было весьма позитивным, Джамал решил принять Динеша на работу. На основе перечня должностных обязанностей и результатов оценки навыков Боб и Джамал написали письмо с предложением о занятии вакансии. Для дополнительного введения Динеша в курс дел Джамал запланировал для Эммы один час в день на кураторство Динеша, начиная с момента его выхода на работу, а также попросил Боба подготовить для Динеша самую свежую копию справочника сотрудника. Он также приступил к планированию однодневного обучения для Динеша и новых младших тестировщиков в течение следующих трех-четырех недель[1].

№ этапа	Этап	Выполнено?
1.E	Если есть подходящие кандидаты, сделать предложение наиболее успешному из них, как правило, направив ему соответствующее письмо.	☑
1.F	Если наиболее успешный кандидат принимает предложение, принять нового сотрудника на работу. Если нет, повторить этапы 1.E и 1.F для каждого следующего по успешности кандидата, пока успешный кандидат не примет предложение, или, если ни один из желательных кандидатов не принял предложение, начать процесс заново с этапа 1.A. Уведомить отвергнутых кандидатов, что они должны выбирать другие варианты.	☑

[1] Крис Денардис (Chris DeNardis) в статье "Perspectives From a Test Manager" в журнале "*Software Testing and Quality Engineering*", Volume 2, Issue 5 (Sep/Oct 2000), доступной в настоящее время на www.stickyminds.com, рекомендует проводить обучение самому программному продукту, а также выпускать руководство тестировщика для всех вновь принятых на работу тестировщиков.

Динеш Кумар

Профессиональный опыт

2000 — наст. время Denial180 Буффало, штат Нью-Йорк

Старший инженер-тестировщик, консультант

- Разрабатывал средства тестирования для набора инструментальных средств (SDK) компании. Разрабатывал систему тестов для динамически генерируемых тестовых сценариев для web-приложения. Разрабатывал инструмент автоматизированного тестирования для графического интерфейса, управляемый данными, с использованием Segue SilkTest. Проводил настройку исходных нагрузочных тестов с помощью Segue SilkPerformer.

1999 — 2000 DefunctBank Провиденс, штат Род-Айленд

Старший инженер-тестировщик, консультант

- Оценивал и разрабатывал инструменты тестирования, а именно средства автоматизации тестирования, отслеживания проблем и SCM. Разрабатывал функциональные, нагрузочные тесты и тесты для проверки производительности для web-приложения финансового назначения с помощью Mercury Interactive LoadRunner.

1999 Rustbucket Motor Credit Детройт, штат Мичиган

Старший инженер-тестировщик, консультант

- Разрабатывал процессы тестирования и контроля качества для среды разработки клиент-серверных приложений. Оценивал инструменты автоматизации тестирования, давал рекомендации по приобретению и готовил определение инфраструктуры автоматизированного тестирования.

1998 — 1999 Star69 Калгари, провинция Альберта

Старший инженер-тестировщик, консультант

- Разрабатывал планы тестирования и процедуры автоматизированного и ручного тестирования для клиент-серверного приложения на основе телефонной системы частного пользования. Тестовая среда — QA Partner на NT 4.0.

1998 — 1999 BFSP Software Уимбердудл, штат Пенсильвания

Инженер по контролю качества

- Разрабатывал планы тестирования и процедуры для автоматизированного и ручного тестирования миграции баз данных, утилит обновления и конверсии. Тестовая среда — QA Partner на NT 4.0 и клиенты Windows 98.

1996 — 1998 DefenceGiant Сиэтл, штат Вашингтон

Инженер по контролю качества

- Разрабатывал планы тестирования и процедуры для автоматизированного и ручного тестирования систем регистрации и вспомогательных утилит военного назначения. Оценивал системы управления дефектами для принятия решения о выборе. Тестовая среда — Windows NT 4.0/3.51 и клиенты Windows 9x с серверами HPUX Oracle, использующими QA Partner.

Рис. 8.3. Резюме успешного кандидата Динеша Кумара (Dinesh Kumar)

Профессиональный опыт

1994 — 1996 BigTestCompany Орландо, штат Флорида

Инженер по контролю качества

- Разрабатывал планы тестирования и процедуры для автоматизированного и ручного тестирования для различных продуктов Intersolv PVCS, включая Version Manager и Tracker. Тестовая среда — QA Partner на Windows NT 4.0 и 3.51 и Windows 95 и 3.1.

1989 — 1994 DenseByte Альбукерк, штат Нью-Мексико

Инженер по контролю качества

- Разрабатывал, сопровождал и выполнял тестовые процедуры для Density, системы восстановления баз данных и генерации отчетов. Тестовая среда — AutoTester, скрипты UNIX shell, VMS DCL и C на UNIX, VMS, DOS и Windows.

1988 — 1989 DenseByte Альбукерк, штат Нью-Мексико

Программист-аналитик

- Разрабатывал программные продукты, используя утилиты распознавания образов, созданные в компании. Приложения включали в себя систему исследования баз данных. Вся разработка велась на персональных компьютерах с использованием языка Си.

1988 — 1988 Depends Альбукерк, штат Нью-Мексико

Программист-аналитик

- Разрабатывал дополнения и участвовал в системной поддержке приложения для демонстрации преимуществ страхования жизни для персональных компьютеров. Приложение было конвертировано с языка Фортран на С.

1986 — 1988 Big Massive Utility Альбукерк, штат Нью-Мексико

Программист-аналитик

- Разрабатывал программы управления межпроцессорным обменом на Фортране для VAX./VMS для системы учета расхода газа и электроэнергии SCADA. Приложения обмена включали в себя графическое представление, модуль удаленной передачи данных и поддержку пользовательского приложения для системных сервисов.

1986 — 1986 Scorpion Tech Альбукерк, штат Нью-Мексико

Программист-аналитик

- Участвовал в разработке фрагмента торговой системы под управлением процессора на базе SWTPC 68010.

1985 — 1986 Labirint, Inc. Альбукерк, штат Нью-Мексико

Программист-аналитик

- Оказывал помощь в проектировании и разработке языка четвертого поколения и системы разработки приложений для персональных компьютеров. Разрабатывал пользовательскую документацию.

1984 — 1985 TelComSwitch Альбукерк, штат Нью-Мексико

Программист-аналитик

- Разрабатывал и сопровождал программу пользовательского интерфейса для главного компьютера телекоммуникационной системы. Проектировал и разрабатывал программы для поддержки распределенной архитектуры ACD86 1.3. Координировал работу с техническими писателями для обеспечения полноты и точности пользовательской документации.

Рис. 8.3 (*продолжение*)

Профессиональный опыт

1981 — 1984 DatumSphere Альбукерк, штат Нью-Мексико

Специалист по сертификации программных продуктов

- Сертифицировал программы для телекоммуникационных систем, устанавливал требования и определял процедуры для тестирования на основе анализа спецификаций архитектуры и функциональности; проектировал и кодировал системы тестов.

1979 — 1981 Chaos Computing Финикс, штат Аризона

Программист-аналитик

- Разрабатывал и сопровождал программы для интерактивных графических геосистем. Проектировал и разрабатывал программы для доступа к графическим и неграфическим базам данных и переформатирования информации для выдачи отчетов.

1979 — 1979 Forko Альбукерк, штат Нью-Мексико

Программист-аналитик

- Разрабатывал и сопровождал программы для университетов, главным образом интерактивные программы для научных и деловых приложений. Консультировал преподавателей и студентов.

1978 — 1979 Brainiak Univ. Уимбердудл, штат Пенсильвания

Программист-аналитик

- Разрабатывал программу для сбора данных в реальном времени для исследований сердечной деятельности. Разрабатывал программы для анализа данных, поддерживающие программу сбора данных.

Статьи

Ноябрь, 1996 EuroConference Париж, Франция

- "Integrated Test Automation" в соавторстве с Бобом Джоунзом (Bob Jones), Дженной Брюкером (Jenna Brucker) и Эмамалини Чоудри (Hemamalini Chowdry).

Образование

1975 — 1977 Braniac Univ. Уимбердудл, штат Пенсильвания

Рис. 8.3 (продолжение)

Выявление ключевых навыков тестировщика и управление ими

Одним из важнейших инструментов, только что представленных в главном примере книги, является подготовленная Джамалом таблица ключевых умений и навыков группы тестирования. Он использовал этот документ для подготовки первоначального рекламного объявления о вакансии, для отсеивания резюме и для подготовки описания должностных обязанностей. Как бы вы подготовили такую же таблицу для ваших групп тестирования?

Первым делом нужно определить, какие умения и навыки являются критическими для группы тестирования. Мне нравится применять для этого нисходящий подход. Если внимательно посмотреть на рис. 8.2, можно заметить, что я выделил четыре основные категории навыков: общая квалификация, навыки тестирования, знание предметной области и технические знания[1].

Первая категория, общая квалификация, обычна для большинства компаний. Здесь описывается некоторый базовый уровень навыков и образования, необходимый для занятия должности. Для специалистов-тестировщиков, т.е. для поставщиков информационных продуктов и услуг, помимо прочего, имеет большое значение умение доносить свои мысли в письменной и устной форме как формально, так и в неформализованной среде. Также для статического тестирования в форме рецензирования, инспекций и просмотров необходимо умение тщательно читать документы с большим вниманием к деталям.

Однако больший интерес представляют области навыков тестирования, знания предметной области и технические знания. Считаю полезным изобразить навыки в трехмерном пространстве, как показано на рис. 8.4. Мы можем разместить различных сотрудников группы тестирования в этом пространстве на основе их специфических навыков, а также сделать обобщения по поводу того, как различные должности отображаются в этом пространстве.

Различные проекты и группы предъявляют различные требования к каждой из этих трех областей. В группе тестирования проекта "Суматра" знание предметной области является низшим уровнем компетентности группы тестирования. Это правильно? Ответ зависит от продукта. Текстовый процессор – это простое приложение, знакомое большинству пользователей компьютера. Хотя текстовые процессоры сейчас содержат массу модных "примочек", большинство пользователей текстовых процессоров применяют весьма узкое подмножество свойств. Даже если принять во внимание огромное число планов тестирования, статей

---

[1] Этот инструмент я придумал сам много лет назад. Другие тестировщики разработали похожие инструменты. Таблица ключевых навыков, представленная в этой главе, содержит многие идеи, которые используются в таблице оценки навыков, разработанной Енг Нгуеном (Hung Nguen). Я благодарю Енга и компанию LogiGear Corporation (www.LogiGear.com) за предоставленную версию. Джеймс Бах (James Bach), Берни Бергер (Bernie Berger) и Элизабет Хендриксон (Elisabeth Hendrickson) также предоставляют несколько идей, касающихся ключевых навыков, необходимых тестировщикам.

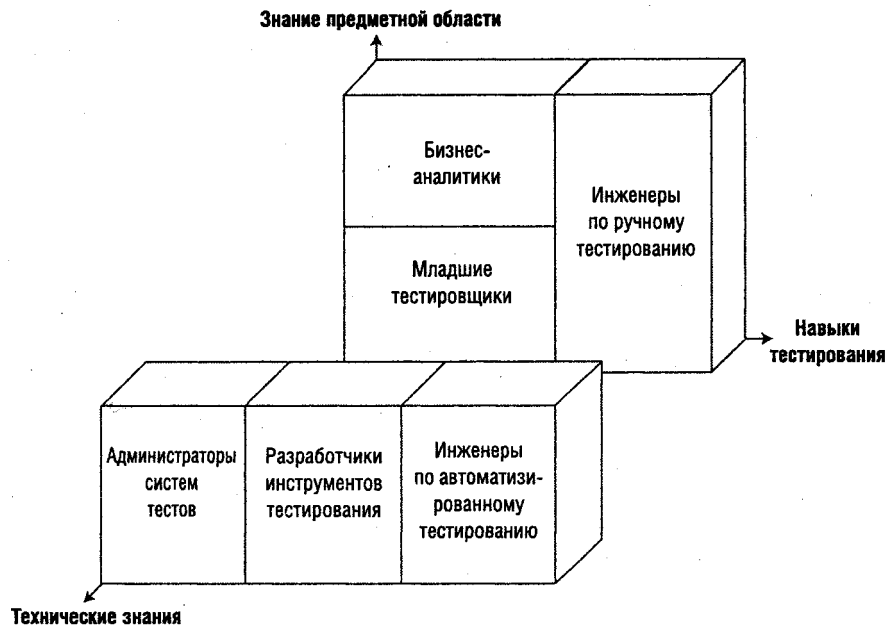


Рис. 8.4. Пространство навыков и должности тестировщиков

Умения и навыки тестирования (Testing Skills)

Квалификационные способности, относящиеся к выполнению задач тестирования, например к разработке качественных тестовых сценариев и написанию подробных отчетов о дефектах.

Навыки в предметной области приложения, экспертиза в предметной области (Application Domain Skills, Subject Matter Expertise)

Навыки или знания, относящиеся к какой-либо области или к решаемым задачам в программном или аппаратном обеспечении; другими словами, понимание того, как заказчики и пользователи будут применять создаваемую систему.

Технические умения и навыки (Technical Skills)

Способность применять определенные методы и инструменты, которые планируется использовать при создании системы, например, языки программирования или серверную и сетевую архитектуру.

и книг, написанных мной, подозреваю, что я использую примерно 20% возможностей моего текстового процессора.

Рассмотрим в качестве противоположного примера компанию одного из моих клиентов. Они пишут геологическое программное обеспечения

для поиска нефти. Это настолько сложное приложение, что даже их собственная группа тестирования, составленная полностью из геологов, геофизиков и других подобных специалистов, не может целиком понять и протестировать все типичные сценарии использования системы. В результате эта группа тестирования вынуждена приглашать на фазу системного тестирования некоторых наиболее активных пользователей из числа своих заказчиков в качестве гостевых тестировщиков.

Процесс найма персонала оказывает влияние на совокупность навыков группы тестирования и находится под влиянием этой совокупности. Группа тестирования Суматры начала работу, имея совокупность навыков, представленную на рис. 8.2. Предположим, что мы обобщили эту информацию для различных категорий навыков. Тогда совокупность навыков группы тестирования на верхнем уровне детализации будет выглядеть, как показано на рис. 8.5.

В небольших группах тестирования, как эта, важно иметь всесторонне подготовленную группу специалистов. Таким образом, частью процесса найма Динеша был учет того, как он заполняет пробелы в квалификации и увеличивает совокупные возможности группы тестирования. После того как Динеш приступил к работе, совокупные навыки группы тестирования изменились, как показано на рис. 8.6. Динеш, очевидно, — самый

	A	B	C	D	E	F
1		Software Cafeteria Test Team				
2		Skills Assessment Minima and Averages				
3		TM Jamal Brown	MTE Lin-Tsu Wu	ATE Emma Moorhouse	Team Minimum	Team Average
4	Average Testing Skills	2.4	2.1	1.9	1.4	2.1
5	Average Domain Knowledge	0.8	1.3	2.0	0.7	1.4
6	Average Technical Expertise	1.6	2.0	2.1	1.1	1.9

Рис. 8.5. Краткая сводка совокупных навыков группы тестирования

	A	B	C	D	E	F	G
1		Software Cafeteria Test Team					
2		Skills Assessment Minima and Averages					
3		TM Jamal Brown	MTE Lin-Tsu Wu	ATE Emma Moorhouse	ATE Dinesh Kumar	Team Minimum	Team Average
4	Average Testing Skills	2.4	2.1	1.9	2.6	1.4	2.2
5	Average Domain Knowledge	0.8	1.3	2.0	2.0	0.7	1.5
6	Average Technical Expertise	1.6	2.0	2.1	2.8	1.1	2.1

Рис. 8.6. Краткая сводка совокупных навыков группы тестирования после принятия на работу Динеша Кумара

сильный сотрудник группы (исключая навыки управления) и поднимает группу в среднем по техническим знаниям и навыкам тестирования. Использование взаимного обучения даст возможность Динешу поделиться своими умениями с остальной частью группы, поднимая средние значения еще выше. К этой теме я вернусь в следующей главе.

О младших тестировщиках

Джамал планировал взять на работу не только Динеша, ему еще нужно нанять четырех младших тестировщиков по контракту. Должность младшего тестировщика, вероятно, не знакома читателю, поэтому позвольте мне объяснить, что я имею в виду. Я ищу и нанимаю младших тестировщиков — наименее оплачиваемых и наименее квалифицированных сотрудников группы тестирования — для выполнения ручных и автоматизированных тестовых сценариев, подготовки отчетов о результатах тестирования и дефектах и прочих работ, поручаемых им инженерами-тестировщиками. Когда я разговариваю с людьми о младших тестировщиках, у них часто возникают вопросы и различные идеи. Что такое младший тестировщик и где его найти? Как заставить работать эти взаимоотношения и на работодателя, и на сотрудника? Не противоречит ли использование младших тестировщиков тому, что тестировщикам требуются навыки в технологиях и тестировании и знания предметной области?

Что касается навыков, необходимых для выполнения работы, определенно существует много задач тестирования, требующих навыков или знаний в используемой технологии разработки, предметной или проблемной области системы и, не в последнюю очередь, в тестировании. (Под навыками тестирования я понимаю все навыки, необходимые для ключевых процессов тестирования, обсуждаемых в этой книге, а также другие навыки тестирования, в том числе проектирование, документиро-

Инженер-тестировщик (Test Engineer)

Специально обученный член команды тестировщиков, имеющий практический навык и опыт работы во всех областях тестирования (планирование, проектирование, реализация и выполнение). Обязан обладать также необходимыми техническими навыками и знанием предметной области. (В некоторых странах специалисты, имеющие инженерную квалификацию, должны подтверждать ее профессиональной лицензией.)

Младший тестировщик (Test Technician, Test Specialist)

Член команды тестирования, менее подготовленный, обученный и опытный, чем инженер-тестировщик, в одной, двух или всех трех критических областях (тестирование, предметная область, технологии). Младшие тестировщики обычно используются для проведения ручного или автоматизированного тестирования, а также для выполнения других технических задач.

вание тестов и знание теории автоматизированного и ручного тестирования, а не только выполнение тестов.) Для выполнения этих задач нужны инженеры-тестировщики, люди, обладающие навыками, опытом и знаниями инженера-программиста, но в своей работе ориентированными на тестирование.

Однако также верно, что многие задачи тестирования, особенно ручное регрессионное тестирование с помощью скриптов, установка программ для тестирования и конфигурирование систем тестов, могут быть механически повторяющимися действиями. В некоторых случаях автоматизация не является практичным решением. Например, если группа тестирования выполняет регрессионное тестирование системы с нестандартным интерфейсом, инструменты автоматизации тестирования могут не поддерживать такой интерфейс, и нет времени на разработку средства автоматизированного тестирования внутри компании. Вместо использования дорогих, высококвалифицированных инженеров-тестировщиков я предпочитаю нанимать младших тестировщиков для выполнения обязанностей, связанных с тестированием вручную.

Признание, что некоторые задачи тестирования являются механическими и повторяющимися и находятся в рамках возможностей менее квалифицированных сотрудников, не противоречит тому, что тестирование само по себе предполагает уникальную квалификацию. Наличие младших тестировщиков в группе тестирования не должно нарушать их прав и лишать их уважения со стороны других сотрудников компании. Кроме того, тест-менеджер может назначать ответственными за выполнение определенных задач как инженеров-тестировщиков, так и младших тестировщиков на основе их навыков. Наличие младших тестировщиков должно способствовать укреплению духа коллектива, а не наносить ему ущерб.

Вряд ли кто-то представит, что шеф ресторана менее квалифицирован, раз он нанимает людей мыть посуду; или поскольку в Белом доме есть стажеры, быть президентом — это работа низкой квалификации. Мы сами пользуемся преимуществами, предоставляемыми различиями в умениях и навыках в повседневной жизни и бизнесе. На самом деле, я бы мог доказать, что невозможность задействовать младших тестировщиков и принуждение инженеров-тестировщиков выполнять работу, не требующую их уровня квалификации и опыта, действительно наносит ущерб имиджу тестирования как профессии, а также приводит к бессмысленному расходованию денег.

Младшие тестировщики приходят из разных мест и имеют за плечами разный опыт. Рекламное объявление, размещенное в местной газете и в Интернете, обычно приводит к обращениям большого количества квалифицированных кандидатов во многих регионах США. Я пониманию, что в некоторых местах США, а также в большей части Европы и Азии тест-менеджеры испытывают трудности в поисках квалифицированных младших тестировщиков и в их продуктивной загрузке после приема на работу из-за различий в рынках труда, профессиональных ожиданий, несоответствия доходов и т.п. Тем не менее, попробовав использовать младших тестировщиков, многие поражаются, как просто создать рабочее место для

таких людей в группе тестирования, насколько они производительны, и как каналы, с помощью которых никогда не найти инженеров-тестировщиков, оказываются полезными для поиска и найма младших тестировщиков.

Как показывает опыт проектов, в которых я участвовал, лучшие младшие тестировщики делятся на следующие три категории:

- Студенты или недавние выпускники, особенно технических вузов или программ сертификации.
- Специалисты, выбравшие новый, второй путь продолжения карьеры, особенно уволенные в запас военные, которые хотят найти себя в компьютерном бизнесе.
- Бывшие специалисты по технической поддержке, сетевых систем поддержки, а также персонал центров обработки вызовов, особенно имеющие опыт работы с применяемой технологией или предметной областью.

Однако нужно забрасывать сети достаточно широко. Можно позволить себе установить планку для младших тестировщиков довольно низко, поскольку мы собираемся их обучать тому, что они должны знать. Что действительно важно для младших тестировщиков, так это желание учиться, самостоятельность, способность к напряженной работе и правильное отношение к ней.

Нужно обязательно планировать обучение младших тестировщиков. Для младших тестировщиков, работающих по контракту, я использую назначение кураторов из числа инженеров-тестировщиков и однодневное обучение. Для постоянных сотрудников, выбравших карьеру в вашей компании, или в случае сложных приложений может потребоваться более серьезное обучение и наставничество. Начинающие младшие тестировщики нуждаются в обучении основам тестирования: управлению рисками качества с помощью тестирования; концепциям структурного и поведенческого тестирования, ручного и автоматизированного тестирования; выполнению скриптов для ручного тестирования; написанию хороших отчетов о дефектах. Им необходимо рассказать о других документах, с которыми они могут встретиться (например, о плане тестирования), о том, как тестирование встраивается в жизненный цикл разработки, об экономике производства программных продуктов и о проблемах, возникающих в ходе выполнения тестов.

Также необходимо организовать дополнительное наставничество и руководство младшими тестировщиками, особенно не имеющими серьезного опыта работы. Действительно, есть большая разница в подходе к младшему персоналу и опытным специалистам. У меня работали два младших тестировщика, которые быстро справились с серьезными техническими трудностями и пришли за новыми заданиями раньше, чем я ожидал. У меня также работал младший тестировщик, который часами просиживал без дела в тестовой лаборатории, ожидая появления новой версии, не заботясь о том, чтобы позвонить или эскалировать возникшую проблему. (Все три младших тестировщика были недавними выпускниками компьютерных факультетов ведущих университетов.) Трудно заранее точно оценить, как человек будет работать в качестве младшего тестиров-

щика, поэтому нужно внимательно наблюдать и при необходимости вносить коррективы в его работу.

Это может показаться довольно хлопотливым делом, но оно того стоит. Во-первых, вам удастся избежать недовольства опытных тестировщиков по поводу работы, которую они считают глупой, и тем самым повысить моральное состояние группы. Во-вторых, при разумном использовании младших тестировщиков компания может сэкономить большую сумму денег, что может сделать предусмотрительного тест-менеджера, нанимающего младших тестировщиков, героем, а также предоставит ему возможность работать с более внушительной по размеру группой и выполнять больший объем тестирования, чем это было бы в противном случае.

В-третьих, но не в последнюю очередь, тест-менеджера, получающего шанс-другой поработать с младшими тестировщиками, которые кажутся недостаточно квалифицированными, может ожидать приятный сюрприз. Были такие младшие тестировщики, которые меня разочаровали. Но большинство оправдали мои ожидания относительно того, что они должны делать. По крайней мере, могу вспомнить троих, которые, по моему мнению, были одними из лучших сотрудников. У вас всегда есть возможность дать человеку отдохнуть, перевести его на другую должность, на другую работу, позволить ему сменить направление развития карьеры. Я бы рекомендовал не зацикливаться на незначительных и абсолютно несущественных отличиях в одном или двух годах опыта работы, степени бакалавра или незаконченного образования бакалавра, а создать нижнюю ступеньку в вашей компании, чтобы принимать людей, которые смогут стать вашими лучшими сотрудниками.

Варианты состава группы тестирования: временное назначение, ротация, команда-питомник и тихая заводь

До настоящего времени мы рассматривали найм новых сотрудников и развитие карьеры с неявным предположением, что люди присоединяются к группе тестирования, работают в ней на протяжении некоторого периода своей карьеры, а затем занимают более высокие технические или руководящие должности в компании. Это, конечно, один путь развития карьеры, причем вполне обычный, но едва ли единственный. Мне известны несколько распространенных альтернатив, которые заслуживают упоминания.

Первый вариант — это временное назначение сотрудников, работающих где бы то ни было в компании, в группу тестирования. Эти люди могут усиливать группу тестирования в период выполнения тестов или работать в ней в течение всего проекта. Эту роль нередко играют пользователи и сотрудники группы технической поддержки разрабатываемой системы. Они вносят значительный вклад в знание предметной области в ходе тестирования. Если этих сотрудников можно включить в группу

тестирования на этапах планирования и подготовки, которые рассматриваются в первых двух частях книги, они также помогут убедиться, что тестирование отражает реальные потребности пользователей.

Такой подход бывает эффективным, особенно когда сотрудники пополняют существующую группу инженеров-тестировщиков. В этих случаях приток знаний предметной области часто заполняет вакуум в совокупности навыков группы тестирования, что в противном случае привело бы к серьезным упущениям. В качестве примера напомню ранее упомянутых разработчиков программ для поиска нефти. Тестировщики, приглашенные из крупнейших компаний-заказчиков, участвовали в системном тестировании. Они принесли свои наборы данных. И им предоставили возможность тестировать систему так, как они ее используют. Это позволило проверить основные сценарии использования системы и данных, которые в противном случае оказались бы неисследованными.

Тем не менее могут возникнуть проблемы, если группа тестирования состоит только из временных сотрудников. Такие группы независимо от того, кто включен в них — эксперты в проблемной области или технические специалисты, — обычно испытывают недостаток навыков тестирования. Однажды я наблюдал за проектом продолжительностью в несколько лет и с многомиллионным бюджетом, который частично провалился, поскольку сотрудникам, не умеющим тестировать, были поручены все ключевые работы по тестированию одной из подсистем.

Другой альтернативой является ротация, подход, который, по всей вероятности, не является столь традиционным, как он того заслуживает. Например, сотрудники могут переводиться из группы разработки в группу тестирования, из группы тестирования в группу технической поддержки, из группы технической поддержки в группу программистов и т.д. Другими словами, эти три группы можно рассматривать как техническую команду системы и изменять обязанности ее членов каждые несколько месяцев. В ходе ротации при помощи менеджеров, которые отслеживают состояние умений и навыков каждого сотрудника и всей руководимой ими группы, люди получают всестороннее развитие навыков. Специалисты, которые рассказывали мне о таком подходе к работе, единогласно выражали удовлетворение его результатами.

Использование ротации приводит к тому, что все менеджеры, через чьи группы проходят люди, соглашаются с таким подходом. Отдел управления персоналом, если он есть, также нуждается в поддержке таких возможностей для сотрудников. Менеджеры должны убедиться, что сотрудники, которых планируется перемещать на другие позиции, понимают цели и согласны с назначениями. Для того чтобы контролировать, что люди, перемещенные на новые должности, получили назначения, соответствующие их набору навыков (в противном случае ротация может отрицательно сказаться на производительности и эффективности работы одной или всех групп), необходимо тщательное управление структурой и навыками группы. Поскольку перемещение в группы тестирования и технической поддержки обычно рассматривается программистами как нежелательное, для реализации такого плана необходимо наличие опре-

деленных навыков управления. Новые сотрудники также должны заранее знать о планах ротации[1].

Я также слышал о пользе ротации от тест-менеджеров, работающих в компаниях, в которых группа тестирования является местом, где новые сотрудники, в частности, имеющие намерения заниматься программированием, проводят некоторый период времени, изучая продукт. Тест-менеджеры, руководящие такими группами, говорят, что этот подход полезен для компании в целом. Однако у меня есть три замечания.

Во-первых, подход способствует увеличению знания предметной области и технической экспертизы, что является ключевым фактором эффективного тестирования, но сводит к минимуму навыки, относящиеся именно к тестированию. Кроме того, как отмечалось в главе 1, довольно сложно убедить тестировщика в этой роли повышать навыки тестирования, поскольку рост этих навыков не соответствует его карьерным устремлениям. Во-вторых, непрерывная текучка в группе тестирования добавляет новые проблемы тест-менеджеру, который и без того достаточно занят. Мне кажется, что для того чтобы заставить этот подход работать, тест-менеджер и, естественно, вся компания должны совместно работать в поисках решения этих проблем.

Третья проблема, касающаяся использования группы тестирования в качестве тренировочного полигона возникает в том случае, когда вы понимаете, что не можете выдвигать на другие роли одного или нескольких сотрудников. Некоторые люди могут по каким-то причинам не подходить для перемещения в другие группы, и менеджеры этих групп откажутся принять их. Проблема не является спецификой именно этого подхода, но возникает всегда, когда группа тестирования становится тихой заводью или болотом для сотрудников, отвергнутых другими подразделениями компании. Это, естественно, наиболее проблемная ситуация для ведущего тестировщика или тест-менеджера при построении группы тестирования. Когда я сталкивался с ней, для меня неявным посылом был следующий: "Здесь можно работать с людьми, которые считаются нежелательными по разным причинам; нужно идти и проводить тестирование в существующих условиях". Некоторые из этих людей превратились в первоклассных тестировщиков, в то время как другие оказались для меня неисчерпаемыми источниками проблем.

---

[1] Роберт Сабурин (Robert Sabourin), главный консультант компании Amibug и один из рецензентов этой книги, также сообщил об огромном успехе хорошо подготовленной программы ротации. Он писал мне: "Я использовал много вариантов ротации. Одним из лучших себя зарекомендовал следующий. Сотрудники групп обучения пользователей, технической поддержки, инсталляции и даже разработки назначаются на один день каждый месяц все без исключения, даже менеджеры, в группу тестирования. В течение этого дня они работают над выполнением, проверкой и даже проектированием пользовательских сценариев и соответствующих тестовых процедур. Мы превратили это занятие в развлечение, которое приносит большую помощь тестированию, обогащает новыми знаниями все группы и повышает уважение к работе, выполняемой группой тестирования. Преподаватели, технические писатели, сотрудники поддержки получают возможность заранее "поиграть" с новыми продуктами и получают представление о том, что появится в конце разработки. Кроме того, все привлекаемые программисты повышают свои навыки в модульном тестировании и более напряженно работают над тестами, предназначенными для проверки состояния системы внутри группы программистов, и тестами для приемочного тестирования версий перед передачей сборок группе тестирования!"

Вы могли подумать, что включение сотрудников в свою группу будет оценено как средство добиться расположения других менеджеров, однако мой опыт показывает, что это не так. Однажды я работал с менеджером по маркетингу, который решил, что один из программистов испортил его любимое свойство системы. На самом деле из-за неспособности определить требования к этому свойству системы менеджер по маркетингу решил возложить вину за результат на программиста.

Программист был переведен в качестве наказания в группу тестирования, но всякий раз, когда приходило время важных проектов тестирования, этот маркетолог хотел немного поуправлять тестированием, настаивая на том, чтобы упомянутому экс-программисту не доверяли "работать ни с чем важным". Молчаливое согласие с этим назначением с надеждой избежать политических волнений позволило мне только сузить пространство для собственного маневра. Мои возможности стали ограниченными, и на все, что я поручал несчастному экс-программисту, накладывали отпечаток его предыдущие неудачи. Я рекомендую избегать таких ролей на задворках. Определение перечня умений, навыков, уровня образования и опыта, необходимых для инженера-тестировщика или младшего тестировщика в группе тестирования, выступает в качестве одного из элементов установления входного барьера на достаточной высоте, чтобы не допустить подобных неприятностей.

После создания группы тестирования, если это не временный коллектив сотрудников, созданный на один раз для тестирования продукта и не требующий непрерывной поддержки, необходимо строить команду вокруг этих людей. Группа должна обладать соответствующими навыками, опытом и знаниями для успешного проведения тестирования. Тестировщики должны иметь правильное отношение к работе, чтобы четко выполнять свои обязанности и без проблем взаимодействовать с остальной частью группы проекта. А для поддержания мотивации работа, которую выполняет каждый тестировщик, должна соответствовать его карьерным устремлениям.

Образование, дополнительное обучение, сертификация и превращение тестирования в профессию

Надлежащее образование тестировщики могут получить в разных местах. Во-первых, есть колледжи. В примере на рис. 8.1 степень бакалавра является одним из требований для инженеров-тестировщиков. Я также отмечал в качестве требования непрерывное обучение. Некоторые компании предоставляют помощь сотрудникам в получении степени магистра и даже доктора.

Но нужно ли тестировщикам формальное образование? Обычно я отдаю предпочтение инженерам-тестировщикам, окончившим колледж. Любые обобщения опасны, но, по моему мнению, четырехлетнее обучение или более высокая степень образования означает некоторое упорст-

во в комбинации со способностью осваивать сложный материал, и это позволяет людям быть полезными.

Некоторые говорили мне, что считают необходимым, чтобы все тестировщики окончили колледж, если это требуется для всех программистов и других технических сотрудников в компании. Если вы хотите создать группу тестирования как параллельную техническую команду, равноправную с группой разработки и другими техническими службами, отсутствие требования к уровню образования при условии, что другие группы имеют такое требование, будет тормозить достижение этой цели. В то же время, когда мы рассматриваем эту тему при изучении курса управления тестированием, один или несколько слушателей почти всегда приводят убийственные аргументы в защиту того, что уровень образования не имеет значения или что тестировщик, имеющий необходимый уровень образования, может не обладать ни настойчивостью, ни способностью к обучению.

Предположим, что вы решили по какой-либо причине, что тестировщики должны формально иметь образование, чтобы попасть в группу тестирования. Отлично, тогда каким должно быть образование? Если вы считаете, что наибольшее значение имеет знание технологий, то вам, возможно, захочется иметь тестировщиков с образованием в области компьютерных наук или разработки программного обеспечения. Если вы считаете, что ключевым является знание предметной области, то, возможно, вы будете искать тестировщиков с образованием в области администрирования бизнеса, бухгалтерского учета или других специфических проблем, которые помогает решать пользователям разрабатываемая система. Если вы думаете, что наибольшее значение имеют навыки тестирования, то, к сожалению, ваши возможности ограничены. Лишь немногие учебные планы специальностей разработки программного обеспечения и компьютерных наук начинают включать курсы тестирования (некоторые даже создают специализации, связанные с тестированием), так что число выпускников, имеющих такое образование, похоже, в ближайшем будущем останется небольшим.

Если требуется разностороннее образование в области компьютерных технологий, существуют другие возможности, помимо четырехлетних колледжей. Некоторые средние школы в Соединенных Штатах в настоящее время являются средними техническими учебными заведениями, включающими в программу обучения компьютерные курсы. Военные училища также проводят компьютерную подготовку для некоторых вновь поступивших на военную службу и кадетов. Общественные колледжи в США предоставляют двухлетнее обучение, есть также технические и торговые училища, которые могут дать серьезную подготовку в области компьютерных технологий.

В некоторых частях мира имеет большое значение четырехлетнее (и более) обучение в колледже по определенной специальности. Например, когда я обсуждал эту тему с моими коллегами тест-менеджерами из Индии, они сказали, что окончание колледжа является необходимым условием для приглашения на интервью, не говоря уже о работе. Аналогич-

но, в Японии формальное наличие образования в колледже рассматривается как существенная часть начала карьеры.

Независимо от того, имеет кандидат формальное образование или нет, обычно требуются специальная подготовка и непрерывное обучение. Существуют различные учебные курсы, семинары и сертификационные программы. Большое количество конференций по тестированию, включая консультирование и публично анонсированные обучающие мероприятия, проводятся в разных точках мира[1].

Курсы и статьи, представляемые на конференциях и публично анонсированных обучающих мероприятиях, проводятся ведущими практиками в области тестирования. До включения в программу конференции статьи проходят оценку группой экспертов. Преподаватели учебных курсов обычно являются авторами многочисленных публикаций, ведущими консультантами или практиками с десятилетиями опыта в тестировании.

Кроме профессиональных конференций, существует много способов пройти платное обучение. Обучение является во всем мире серьезным бизнесом. Находитесь ли вы в Бангоре, штат Мэн, или в Бангалоре, Индия, вы наверняка встретите приглашение пройти обучение и сдать экзамен, чтобы стать сертифицированным инженером Microsoft System. Некоторые компании, производящие инструменты тестирования, в настоящее время также предлагают обучение для получения сертификатов. Наконец, как коммерческие, так и некоммерческие организации предлагают пройти обучение и сдать экзамены на получение сертификатов по тестированию.

В книге "*После золотой лихорадки*" ("*After the Gold Rush*") Стив Макконнел (Steve McConnell) описывает общепринятую совокупность знаний, а также программы сертификации, соответствующие этой совокупности знаний, как один из общих признаков профессии. Существует большое разнообразие сертификационных программ, относящихся к навыкам тестирования. Если вы рассматриваете одну из сертификационных программ, советую вам задать следующие вопросы:

- В какой степени совокупность знаний или программа курсов связана с практическим тестированием, навыками и наилучшими способами выполнения работ?
- Кто разработал и поддерживает совокупность знаний или программу курсов?
- Каковы квалификация, опыт создателей совокупности знаний или курса и какова степень доверия к ним как к специалистам по тестированию?
- Требует ли прохождение обучения для сдачи сертификационного экзамена (экзаменов)? Если да, то каковы квалификация, опыт соз-

---

[1] Я лично представлял и лицензировал учебные курсы по тестированию и управлению им в США, Канаде, Европе, на Ближнем Востоке, в Австралии, Южной и Восточной Азии. Я участвовал и выступал на конференциях по тестированию в США, в Европе, на Ближнем Востоке, в Южной Азии и в Австралии.

дателей учебных курсов и степень доверия к ним как к специалистам по тестированию?

- Можно ли сдать сертификационный экзамен (экзамены) по Интернету?
- Существуют ли требования минимального опыта, необходимого для получения сертификата, наряду со сдачей одного или нескольких экзаменов?
- Соответствует ли сертификационная программа какому-либо промышленному стандарту, и применим ли этот стандарт к вашей деятельности?
- Сертификационная программа посвящена тестированию или контролю качества?
- Какова стоимость сертификационной программы в сравнении с другими программами, которые, возможно, предоставляют аналогичную совокупность знаний или программу курса?
- Необходима ли повторная сертификация или непрерывное обучение для сохранения сертификата? Если да, каковы затраты на это?
- В каких странах предлагается и признается данная сертификационная программа?
- Что думают об этой программе сертификации ваши коллеги и другие специалисты по тестированию, мнению которых вы доверяете и у которых нет других скрытых интересов?
- Не пытается ли программа сертификации проповедовать "единственно верный путь тестирования", чрезмерно инструктивный, контекстно-независимый подход?

Среди программ сертификации есть очень неплохие, которые позволяют соискателям получить дипломы специалистов-тестировщиков. Есть также и плохие сертификационные программы. Права на предложение сертификации обычно не предоставляются никакой ассоциации или организации, за исключением самой сертифицирующей организации. Другими словами, большинство программ сертификации могут быть описаны термином "*бумажная сертификация*" (fiat certification). Они базируются только на доверии, опыте и рекомендациях, которые имеются у людей, участвующих в преподавании программ. Даже хорошие программы весьма различаются между собой, поэтому советую вам проделать подготовительную работу перед выбором сертификационных программ.

Помимо этих программ, некоторые из компаний-разработчиков инструментов тестирования предлагают сертифицироваться для использования их инструментов и специальных методологий разработки, рекомендуемых ими. Однако такие программы с учетом их происхождения всегда, по крайней мере, воспринимаются, если не являются таковыми на самом деле, элементами достижения главной цели самих производителей инструментов — получения прибыли.

Такие компании, как Cisco и Microsoft, предлагают сертификацию в своих особых технологических наработках, также доступна сертификация в таких предметных областях, как финансы и юриспруденция. Хотя

эти программы могут быть полезны при изучении технологий и предметной области, я уверен, что тестирование программного обеспечения дефакто стало отдельной профессией со своими уникальными умениями и навыками в мире разработки систем. Компетентные тестировщики должны обладать этими навыками и демонстрировать мастерство владения ими. Вероятно, через какое-то время станет доступным получение образования, соответствующего современным требованиям в области тестирования, в колледжах, профессиональных училищах или с помощью программ сертификации.

Отношение к делу

Прием на работу сотрудников связан не только с вопросами их квалификации. Некоторые неотъемлемые черты тестировщика больше относятся к мировоззрению, нежели к навыкам. Конечно, всем известно, что означает, когда о ком-то говорят, что у него хорошее отношение к делу. Это общепризнанный элемент мировоззрения профессионала. Однако для тестировщика важны еще некоторые черты.

Первое, это то, что я бы назвал *профессиональным пессимизмом*. Это означает, что тестировщик подходит к решению задач тестирования с рабочим предположением, что в продукте есть ошибки и группа тестирования их обнаружит. Однако тестировщик должен придерживаться этой точки зрения ограниченно, имея в виду общее предположение о том, что качество — это только одна из общих категорий рисков, подвергающих опасности проект. Тестировщики должны защищать качество и мнение пользователей о системе. Но они не должны это делать, выступая в качестве соперников программистов, выдвигая претензии личного характера или в неконструктивной манере. Предпочтительнее, если мы будем это делать путем, объединяющим реалии бизнеса с системной разработкой и сопровождением.

Во-вторых, тестировщики должны привносить разумную любознательность в свою работу. Чтобы найти ошибку, тестировщики должны быть скрупулезными как при разработке, так и при выполнении тестовых сценариев. Отсутствие любознательности ведет к небрежности в тестовых сценариях, которые не проверяют наиболее укромные уголки и трещины в системе, являющиеся потенциальными источниками ошибок. Отсутствие любопытства приводит к появлению отчетов о дефектах, которые не связаны со сбоем, наблюдаемым с точки зрения пользователя. Однако важно соблюдать разумный баланс, поскольку приходится писать и выполнять большое число тестов в рамках ограниченных ресурсов и времени. Тестировщик должен всегда задаваться вопросом: "Сколько усилий потребуется для решения данной задачи?"

Третье, также относящееся к разумному балансу, — это способность концентрироваться. Под ней я подразумеваю понимание ключевых приоритетов и нацеливание тестирования на следование им. Это трудно сделать, поскольку приоритеты часто меняются. (Конечно, тест-менеджер должен обсуждать приоритеты и причины, по которым они меняются, с инженерами-тестировщиками и младшими тестировщиками.) Я знаю тес-

тировщиков, которые из-за неспособности концентрировать свое внимание испытывали трудности в выполнении поставленных перед ними задач на должном уровне качества и в надлежащие сроки. Хотя они обладали хорошими знаниями в тестировании, этот единственный пробел в их характере ограничил их потенциал.

В четвертых, тестировщик должен иметь желание выполнять обобщающую и вспомогательную работу, нацеленную на устойчивый, надежный процесс, а не на поиск славы или магического разрешения проблем посредством личного героизма. Даже в хороших компаниях, которые предоставляют широкие возможности для развития карьеры тестировщиков и способствуют формированию равных отношений с коллегами из других подразделений, тестирование иногда рассматривается как работа с множеством нежелательных свойств. Например, группа тестирования выходит на первый план непосредственно в конце проекта, когда становится очевидным чрезмерный оптимизм при подготовке планов проекта, проявляющийся в ошибочных сроках и перерасходе бюджета. Другим примером является то, что тестировщики должны сообщать плохие новости группе разработки. Не имеет значения, насколько преуспевает тестировщик в дипломатии и общении. Он время от времени сталкивается с сопротивлением и защитной реакцией, выступая в роли гонца с плохой новостью. Оба этих явления порождают дополнительные стрессы в жизни тестировщиков. Хорошие тестировщики заставляют себя работать в условиях недооценки и плохого понимания их роли со стороны других участников проекта.

В-пятых, тестировщики должны быть готовы к напряженной работе, поскольку главное внимание на них обращено в конце проекта, когда нарушаются сроки, испаряется бюджет, растет раздражительность и увеличивается давление с тем, чтобы бросить дополнительные усилия на проект, причем особенно возрастает давление на группу тестирования. В этом нет никакого рационального смысла. Например, в одном из проектов я работал с менеджером разработки, который делал оскорбительные замечания по поводу того, что группа тестирования не работает по 16 часов в день, как программисты. Я сказал контактному лицу заказчика, что, поскольку сотни ошибок стоят в очереди в системе управления дефектами, ожидая исправления, группа тестирования не является узким местом проекта. Тем не менее все закончилось введением ночной смены и выполнением тестов семь дней в неделю, чтобы устранить ощущение того, что группа тестирования неким таинственным образом не выполняет свою часть работы.

Шестое и последнее — тестировщики должны стремиться отстаивать и защищать качество разумно и в пределах бизнес-контекста, но делать это твердо и убедительно. Если тестировщик готовит отчет об ошибке, который не нравится разработчику, и сталкивается с противостоянием "несчастного" программиста, ему не нужно склонять голову, класть руки в карманы и смиренно бормотать: "Хорошо-хорошо, я думаю, что отзову этот отчет об ошибке".

Вместо этого тестировщик должен выпрямить спину, выслушать аргументы программиста и затем сказать что-то вроде этого: "Да, но если бы я был заказчиком и увидел такое поведение системы, у меня не было бы

поводов для восторга. Возможно, я создаю проблему на ровном месте, поэтому мы можем поручить принять решение по этой ошибке комитету по контролю о внесении изменений (или комитету по управлению дефектами) на следующем совещании". Твердый, но гибкий характер является требованием к хорошему тестировщику.

По ту сторону очевидных вопросов в ходе интервью

Одним из аспектов уникальных навыков тестировщиков (что нередко игнорируется) является то, что интервью с ними нельзя проводить так же, как мы проводим интервью с разработчиками. О собеседованиях с программистами при приеме на работу написано много литературы. Amazon.com дает обзор списка рекомендованных книг об интервью с кандидатами на должность разработчика программного обеспечения.

Тем не менее навыки программирования — несущественный, а в некоторых случаях даже ненужный элемент квалификации специалиста-тестировщика. Я бы порекомендовал несколько вопросов, которые выходят за рамки стандартных вопросов типа: "Напишите мне программу, которая копирует файл..." или "Опишите проблемы в своем развитии карьеры и как вы их решали".

- Ожидали ли вы или нет в своем последнем проекте, что тесты пройдут без ошибок? Почему? Будет ли, по вашему мнению, обстоять иначе в текущем проекте? Почему?
- Приведите примеры тестовых сценариев из недавно выполненных вами, про которые вы с уверенностью могли сказать, что с их помощью будут обнаружены ошибки. Как вы об этом узнали? Как вы проверяли ожидаемые результаты?
- Когда вы в последний раз писали отчет об ошибке, какой способ и какую тональность вы использовали при обсуждении ошибки с сотрудниками группы разработки? Удовлетворил ли вас результат обсуждения? Расскажите о моменте, когда вы сообщили об ошибке и почувствовали, что ваши манера и тон способствуют эффективному обсуждению.
- Расскажите о случае, когда вы были убеждены, что обнаруженная вами проблема окажет негативное влияние на мнение пользователей о системе, но с вами не все были согласны. Удалось ли вам развеять сомнения и отстоять отчет об ошибке, или это не входило в ваши обязанности?
- Расскажите о ваших проектах, тестирование в которых доставляло удовольствие и было интересным. Какие его части были приятными и интересными? Какие части тестирования не были приятными или интересными? Расскажите о проектах тестирования с вашим участием, в которых не было ничего интересного. Почему, по вашему мнению, так произошло?

- Расскажите о случаях, когда вам удавалось не соглашаться, не становясь неприятным для собеседников. Что вы можете сказать о случаях, когда добиться этого не удавалось?
- Работая здесь, нужно будет отслеживать множество тестовых сценариев, тестовых данных, вариантов тестовой среды, тестовых версий, отчетов о дефектах (добавьте все, что имеет отношение к вашей ситуации). Состояние каждого элемента может меняться ежедневно. Расскажите, как вы будете управлять таким большим множеством непрерывно меняющихся объектов.
- Иногда эта должность может повлечь за собой 60-часовую рабочую неделю/занятость в некоторые выходные/вечерние смены (что применимо к вашей компании). Вы можете работать по такому графику? Приведите пример из прошлого опыта, когда вы работали таким образом. Что происходило, когда вы переутомлялись? Как вы боролись с этой проблемой?
- Расскажите о последнем случае, когда вы боролись за разумное выделение времени, необходимое на написание тестового сценария. Что вы можете сказать по поводу разумного выделения времени на локализацию дефекта? Если вам поручено выполнять шесть тестов в день, а второй позволил обнаружить ошибку, как вы определите, стоит ли рискнуть завершить четыре оставшихся теста, чтобы исследовать найденную ошибку?
- Расскажите, какие виды информации ваши менеджеры предоставляли вам и как вы использовали полученную информацию для сохранения правильного направления движения. Были ли случаи, когда информация не доходила и вы не понимали смысл?
- Как вы планируете применить то, чему научились в ходе (некоторых ранее выполненных) проектов, для тестирования нашего продукта? Какое поведение и какие действия из тех, что вы выполняли в предыдущих проектах, вы не будете повторять сейчас?
- Почему вам нравится тестирование? Каковы ваши карьерные планы? Предполагаете ли вы продолжить карьеру в качестве тестировщика? Чем вы планируете заниматься через пять лет? Какое место вы отводите тестированию как специализации в ближайшие пять лет?

Можно задавать и много других вопросов, но главное — понять основную идею. Быть хорошим тестировщиком — это больше, чем просто иметь набор навыков и подходящее образование. Быть хорошим тестировщиком, значит обладать также соответствующим отношением к делу, темпераментом и мировоззрением. Вероятно, нужно соответствующим образом организовать процесс интервью[1].

---

[1] Дополнительные идеи по поиску, интервьюированию и приему сотрудников на работу можно найти в статье Сема Канера (Cem Kaner) "Recruiting Software Testers" по адресу www.sdmagazine.com и в справочнике Джоанны Ротман (Johanna Rothman) "Hiring Technical People", доступном на www.jrothman.com.

Коротко о приеме на работу Дебора Маккандлесс (Deborah McCandless)

Планирование до начала поиска сотрудника — это наиболее критичный этап в процессе подбора кадров. Определите потребности вашей группы в самом начале процесса подбора сотрудников, и вы сэкономите время и избежите проблем с персоналом в будущем, даже в случае единственного проекта тестирования.

Начните с изучения подробного описания должностных обязанностей или перечня общих навыков и знаний сотрудника группы контроля качества, соответствующих требуемому уровню квалификации. Затем проведите инвентаризацию знаний, умений и навыков группы в текущем состоянии. Это просто сделать, если проставить отметки справа от элементов перечня общих навыков и знаний сотрудника группы контроля качества или перечня должностных обязанностей. Используйте эту информацию для определения пробелов и слабостей в группе в целом или для данного проекта. Затем подготовьте объявление о вакансии, вопросы интервью и тесты в соответствии с потребностями группы тестирования, проекта и компании в целом.

Полезный прием при подготовке тестов — дать их на рецензирование нескольким людям и получить отзывы. Это позволит уточнить множество вещей, в том числе:

- Являются ли инструкции по выполнению тестов ясными и полными?
- Позволяют ли результаты тестов узнать то, что вы хотите?
- Не являются ли тесты слишком простыми или слишком трудными?
- Сколько времени в среднем нужно, чтоб выполнить тесты?

Изучите и проранжируйте резюме. (Хороший способ сэкономить время и деньги – завести папку для хранения резюме тех людей, которые не подходят на данную позицию, но являются первоклассными специалистами. Позвоните или напишите им электронные письма, когда у вас будут подходящие вакансии.) Не забудьте проверить те навыки и знания, которые подходят для разных специальностей. Проведите телефонные интервью с основными кандидатами. Это позволит понять, насколько они правдивы, когда говорят о своих умениях и навыках, и насколько подходят для занятия вакансии. После телефонных интервью снова проранжируйте кандидатов и пригласите основных из них на интервью. Если вы планируете проводить для них испытания, предупредите их об этом.

Проводите интервью в обстановке, максимально удобной для того, чтобы получить больше открытых и честных ответов. Нужно задать кандидатам все обычные вопросы (об их сильных сторонах, чему они хотели бы научиться, о прошлом опыте и т.д.). Свободная форма вопросов даст возможность кандидатам сказать все, что нужно. Поощряйте их задавать вопросы: люди, которые задают вопросы, обычно хорошо проявляют себя в тестировании. Подготовьте вопросы, которые помогут оценить их способности или навыки в расстановке приоритетов, решении проблем,

управлении временем, оценить организованность, аналитические способности, умение работать в команде, документировать, общаться, доводить дело до конца, давать представления на концептуальном уровне, следовать указаниям, ориентироваться в деталях, оценить гибкость, производительность в условиях стресса и определить наличие целей и мотивации. Подготовьте специальные вопросы для каждого этапа интервью: телефонного интервью, группового интервью или интервью с коллегами, интервью с менеджером.

От увеличения группы к росту квалификации

До сих пор я в основном рассматривал процесс подбора кадров, а также включение новых тестировщиков в группу тестирования. Конечно, это важная часть увеличения группы тестирования. Работа нуждается в исполнителях, и мы должны найти нужных людей для задач, решать которые придется в ближайшем будущем.

Тем не менее большинство специалистов не хотят просто ходить на работу и делать каждый раз одно и то же, ничему не обучаясь и применяя только существующие навыки. Профессионалы хотят двигаться по пути развития карьеры, что означает рост квалификации. Кроме того, совокупность навыков, необходимых для группы тестирования, находится в постоянной динамике. Расширение перечня задач группы тестирования, новые технологии и новые продукты требуют новых навыков группы тестирования. В следующей главе мы посмотрим, как можно управлять этой частью процесса создания группы, стимулируя рост квалификации и развитие карьеры сотрудников группы. После этого мы вернемся к процессу создания группы тестирования, чтобы определить параметры качественного процесса, найти решение неизбежных проблем и внедрить усовершенствования процесса.

ГЛАВА 9

Подготовка превосходной группы тестирования: навыки, отношение к делу и пути развития карьеры

В предыдущей главе я описал первую часть процесса создания группы тестирования, рассмотрев вопросы, связанные с подбором кадров. Очевидно, что для создания сильной команды тестирования очень важен подбор подходящего коллектива сотрудников. Тем не менее существует также долгосрочный аспект построения группы, связанный с обучением. Чтобы получить грамотный, законченный процесс, мы должны уделить внимание этому аспекту. Позвольте посвятить несколько страниц второй части процесса построения группы тестирования, а затем мы вновь рассмотрим весь процесс для завершения темы.

Джамал и Лин Цу обсуждают вопросы повышения квалификации

Сентябрь был трудным месяцем для Джамала. Он не только готовил планы для проекта "Суматра", но также руководил тестированием версии сопровождения для двух других продуктов пакета OfficeArrow и искал пять новых тестировщиков. Кроме того, все нынешние сотрудники его группы проходили ежеквартальную аттестацию роста квалификации. Джамал отнесся серьезно к этой обязанности, поскольку он знал, что долгосрочная жизнеспособность его группы тестирования, а также лояльность его сотрудников существенно зависят от возможностей развития карьеры каждого из них. Таким образом, он провел в офисе всю вторую половину дня субботы, 21 сентября, за изучением таблицы контроля навыков и в размышлениях о том, что нужно сделать. Он отправил результаты

оценки всем членам группы тестирования, сопроводив их следующим письмом.

Привет всем!

К письму прикреплена таблица оценки навыков. Таблица с названием "Управление навыками (S02)" содержит краткий перечень навыков на настоящий момент.

Изучив эту таблицу, я обнаружил, что наиболее слабым местом является для нас знание предметной области, особенно текстовых процессоров, работающих в средах, отличных от Windows. Я думаю, что в ближайшие месяцы это может стать проблемой, особенно для ручного тестирования.

Также, хотя в целом мы сильны в тестировании, есть ряд недостатков в области автоматизированного тестирования, на которые я бы хотел обратить внимание - это тестирование надежности и комплексное тестирование.

Прошу вас потратить немного времени в ближайшие два дня на размышления о разделах, в которых вы хотели бы улучшить свои навыки, имея в виду упомянутые выше требования, предъявляемые к группе тестирования. Я назначу встречи с каждым из вас на следующей неделе, чтобы подготовить планы повышения квалификации.

С уважением,

Джамал

Джамал решил начать с Эммы и Лин Цу, поскольку ему хотелось, чтобы они сразу же приступили к освоению новых навыков, которые им потребуются в проекте "Суматра". Во вторник, 24 сентября, Лин Цу и Джамал провели получасовую встречу в его кабинете. После пары минут светской беседы Джамал перешел к делу.

"Итак, Лин Цу, что ты думаешь о повышении квалификации в следующем квартале?"

"Ты знаешь, что я хочу приобрести больше опыта за пределами Windows, но технологии не являются для нас слабым местом. Тем не менее, что ты думаешь об этом? Если судить по твоему письму, мне кажется, что я бы смогла поработать над улучшением знаний в предметной области, а именно в обработке текстов и управлении документооборотом, включая платформы Mac и UNIX."

"Было бы отлично. Я рассчитывал услышать от тебя что-то подобное, — сказал ободряюще Джамал. — Есть ли какие-то идеи, как это сделать?"

"Да, тут через несколько недель будет конференция, посвященная управлению документооборотом, — ответила Лин Цу. — Предлагаются два учебных курса в ходе конференции, что также кажется полезным". Она передала Джамалу объявление о конференции.

Джамал изучил его, одобрительно мыча. Наконец, он поднял глаза и сказал: "Да, было бы отлично. Знаешь, вообще-то все мы можем посетить только две конференции или пройти два вида обучения в течение финансового года. Ты уверена, что это один из двух возможных случаев, которые ты хотела бы использовать?"

"Да, по двум причинам, – ответила она. – Первая – это нужное обучение в нужное время. Вторая – конференция будет в Сан-Диего, где живут мой брат и невестка..."

"О!" – воскликнул Джамал.

"Я бы хотела уехать в ночь на пятницу, взяв день в счет отпуска, и провести там целую неделю".

"Хм, – произнес Джамал. – Сможем ли мы дать тебе один день в счет отпуска непосредственно перед конференцией? У нас много дел в проекте "Суматра". Боюсь, что десять дней подряд приведут к отсрочке каких-то дел, и это может потом негативно сказаться на работе".

"Но я сэкономлю компании кучу денег на командировке. Сюда попадает уикенд, и надо учесть тот факт, что я остановлюсь у моего брата, а не в отеле".

Джамал сел и на минуту задумался. "У ее плана есть преимущества, – признался он сам себе, – и, вероятно, нечестно мелочиться из-за одного дня в этих обстоятельствах. Конечно, получится больше, чем один день, поскольку она наверняка уйдет рано в четверг, чтобы успеть на самолет. Но для компании хорошо, если удастся сэкономить деньги".

"Хорошо, – сказал Джамал. – Давай заключим сделку. Ты уедешь в четверг вечером, а я не буду считать пятницу днем отпуска, назовем ее днем отгула, если ты вернешься в пятницу вечером, а в субботу выйдешь на работу, чтобы наверстать упущенное. Я буду тоже работать с тобой".

"Согласна", – улыбнулась Лин Цу.

"Итак, как насчет способов применения всех тех знаний, которые ты собираешься усвоить?" – спросил Джамал.

Джамал и Лин Цу продолжили обсуждение способов применения полученных навыков в тестировании. Лин Цу предложила несколько специальных тестов, которые она запланировала разработать и выполнить. Джамал заметил, что Динеш Кумар, новый инженер-тестировщик, который приступит к работе через две недели, хорошо знает области, о которых она говорила.

"Если ты не против, – сказал Джамал. – Я попрошу Динеша проверить твои тесты. Таким способом он сможет передать тебе часть знаний".

"Это было бы отлично. С помощью этого плана, – резюмировала Лин Цу, – я должна достичь уровня между двойкой и тройкой для каждого из четырех навыков в области текстовых процессоров и управления документооборотом для систем Mac и UNIX".

"Отлично. Давай определим цель: повысить уровень владения навыками на единицу в этих областях в таблице управления навыками. Мы оба можем быть приятно удивлены, если ты добьешься большего. Согласна?"

"Здорово!"

"Это позволит тебе опередить график, когда мы будем проводить в январе ежегодную аттестацию. Также, Лин Цу, начинай думать о ежегодной аттестации, чтобы можно было определить для тебя более высокие цели в следующем году. Я в самом деле доволен твоим ростом в профессиональном плане, поэтому я бы хотел нацелить тебя на повышение на следующий год".

№ этапа	Этап	Выполнено?
2.	Стимулировать повышение квалификации и карьерный рост сотрудников группы.	☑
2.A	Совместно с новыми сотрудниками разработать пути развития их карьеры.	☑
2.B	На регулярной основе проверять пути развития карьеры всех сотрудников и отмечать прогресс каждого.	☑
2.C	Активно управлять процессом овладения сотрудниками навыками, которые требуются для достижения целей как сотрудника, так и всей группы тестирования.	☑

Оценка навыков как инструмент управления карьерным ростом группы тестирования

В предыдущей главе было показано, как таблица оценки навыков становится полезным инструментом поиска новых сотрудников для любой группы тестирования, большой или маленькой, специальной или однородной, нацеленной на реализацию конкретного проекта или на наличие определенной совокупности навыков. Можно использовать эту таблицу и для поиска существующих слабых мест в группе тестирования и для приема новых сотрудников с целью ликвидации этих мест. Имея резюме кандидатов, можно также применять таблицу навыков для их отбора, а затем на ее основе нацеливать вопросы интервью на исследование определенных разделов знаний, которые вы пытаетесь добавить к тем, которыми уже владеет группа тестирования.

Как вы уже видели из разговора Джамала и Лин Цу, таблица оценки навыков также является мощным средством для управления развитием карьеры и совокупности навыков группы тестирования. В перечне должностных обязанностей, представленном на рис. 9.3, одним из требований является "участие сотрудника в ежеквартальной аттестации ключевых навыков группы тестирования совместно с тест-менеджером, ежеквартальное определение целей развития навыков в трех, как минимум, разделах путем внутреннего (взаимного) и внешнего (семинары/учебные курсы/конференции) обучения и их успешное достижение".

Тест-менеджер вместе с каждым сотрудником должны регулярно обсуждать и выбирать разделы, в которых сотрудник может повысить квалификацию. Как правило, тест-менеджер имеет список разделов, в которых он хочет улучшить навыки группы в целом, как это было у Джамала, а сотрудники знают, в каких разделах они хотят повысить квалификацию, чтобы продвигаться по выбранному пути развития карьеры. С помощью целевого поручения определенных задач, наставничества и внешнего обучения квалификация сотрудников группы тестирования может непрерывно расти и развиваться.

Это не только на пользу группе тестирования. Чтобы способствовать карьерному росту сотрудников группы тестирования, тест-менеджер должен оказывать им помощь в приобретении навыков и опыта, которые

Таблица ключевых навыков: открытая регистрация или конфиденциальная информация о сотруднике

Когда я пользуюсь таблицей ключевых навыков в качестве средства подбора кадров и управления повышением квалификации сотрудников, я делаю это открыто. Другими словами, каждый может просмотреть уровни развития навыков других сотрудников группы и планы их развития. Я пытаюсь способствовать такой атмосфере в группе, в которой люди честно и открыто говорят о том, что касается их сильных сторон и в чем им требуется помощь. В этом случае они знают, к кому обращаться с конкретным вопросом.

Это также способствует объективной самооценке. Как показывает мой опыт, тестировщик может обманывать меня, говоря о наличии того или иного навыка, поскольку у меня может не быть возможности проверить каждое утверждение подробно, но другие сотрудники группы это заметят. Иногда я говорю об этом сотрудникам, когда рассказываю о процессе самостоятельной оценки навыков, в особенности, если я чувствую, что они могут поддаться искушению сообщить о себе более выгодную, но неточную информацию.

Если вы решили сделать эту информацию открытой, уверен, что нужно учесть два момента:

1. Не используйте этот процесс и полученную информацию в качестве части процесса ежегодной аттестации или повышения зарплаты. Что касается групп, которыми я руковожу, единственным требованием является активное участие сотрудников в процессе повышения квалификации. Я не увязываю оценку повышения квалификации ни с одной целью ежегодной аттестации. Это означает, что информация об управлении повышением в компании не является частью ежегодной аттестации кого бы то ни было и, следовательно, не является конфиденциальной. Если по какой-то причине вы приняли решение использовать данные непосредственно, например, в качестве одной из целей ежегодной аттестации "Увеличить на пять рейтинг навыков в каждом из трех основных разделов", то оценки навыков должны быть конфиденциальными личными данными[1].
2. Не давайте доступ к этой информации вашему руководству и вашим коллегам-менеджерам, если вам кажется, что они воспользуются ею в качестве инструмента для продвижения по карьерной лестнице, увольнения или перевода лучших сотрудников из группы тестирования. Я не распространяю эту информацию за пределы группы тестирования, если стандартом компании не является обратное, и в этом случае информация обо всех группах доступна всем желающим.

Таков мой подход, и некоторые говорили, что он весьма радикальный. Если вы решили внедрить его в свою практику, рекомендую посоветоваться с кем-нибудь из сотрудников отдела управления персоналом по вопросу о том, какие политики конфиденциальности информации допускаются в компании, какую информацию вы имеете право собирать и как необходимо обращаться с полученной информацией.

---

[1] Некоторые компании проводят исследование навыков сотрудников только в качестве части ежегодной аттестации. Я думаю, что это слишком редко, поскольку совершенствование навыков сотрудников в целях совершенствования возможностей группы должно осуществляться постоянно. Кроме того, существует риск увязывания процесса совершенствования навыков, проводимого таким образом, с оценкой производительности, поскольку может оказаться довольно трудным делом отделить оценку и премии сотрудника от численного повышения его рейтинга навыков. Любая привязка, явная или на уровне восприятия, может привести к искусственному увеличению уровня навыков, достигнутого сотрудниками, что снижает ценность процесса повышения квалификации.

необходимы для решения проблем, связанных с их новым назначением. Это может быть новая роль внутри группы тестирования, например, в связи с повышением с должности младшего тестировщика на должность инженера-тестировщика, это может быть новая должность где-то еще в компании, например, в связи с переходом с должности ведущего инженера-тестировщика на должность менеджера сетевой системы поддержки пользователей. Таким образом, тест-менеджер и все сотрудники группы тестирования должны совместно работать для синхронизации процесса совершенствования навыков с направлением развития карьеры каждого сотрудника.

Построение успешного процесса создания группы тестирования

Мы рассмотрели весь процесс создания группы тестирования, поэтому предлагаю обсудить, что дает группе тестирования и принимаемому кандидату хороший процесс создания группы тестирования.

Применение философии "выигрывают обе стороны"

При поиске новых сотрудников вы стремитесь отобрать лучших людей из пула кандидатов, тех, кто имеет наилучшую квалификацию, чтобы помочь проверить качество продукта и способствовать достижению более масштабной цели — управлению рисками качества продукта. Если эти люди выбирают карьеру в вашей компании, вы, вероятно, будете способствовать их продвижению по карьерному пути, что ставит перед тестировщиками более серьезные цели, способствует непрерывному росту производительности и эффективности группы тестирования. Если вы сами выступаете в роли кандидата, вам хочется найти наиболее подходящую работу для данного этапа развития карьеры и знать, что данная работа будет способствовать продвижению по выбранному карьерному пути.

Смысл философии "выигрывают обе стороны" при подборе кадров состоит в том, что обе стороны, работодатели и работники, приобретают от взаимоотношений каждого работника с работодателем. Эти приобретения могут быть как долгосрочными, так и краткосрочными. К сожалению, некоторые менеджеры по подбору персонала, особенно в сложные времена для экономики, подходят к поиску сотрудников и работают в соответствии с правилами игры с нулевой суммой, (например, шахматы или покер), игры, в которой есть победитель и побежденный. В играх с нулевой суммой одна сторона может выиграть только в случае поражения других сторон. Процветающее предприятие, созданное квалифицированными менеджерами, не играет в игры с нулевой суммой. На процветающем предприятии создается богатство, приобретаются знания, а жизнь каждого сотрудника наполняется ощущением успеха.

С помощью процесса создания команды один кандидат выбирает и выбирается на ранее открытую вакансию, а другие кандидаты отвергаются или не получают предложений. Взаимное удовлетворение целей

всех сторон, включая тех, кого не принимают на работу, может иметь место, когда

- Мы привлекаем подходящих кандидатов.
- Мы отвергаем неподходящих кандидатов.
- Мы делаем предложения одному или нескольким кандидатам, обладающим надлежащей квалификацией.
- Выбранные кандидаты получают точную информацию, на основе которой принимают решение принять или отклонить сделанные предложения.

Этот процесс показан на рис. 9.1.

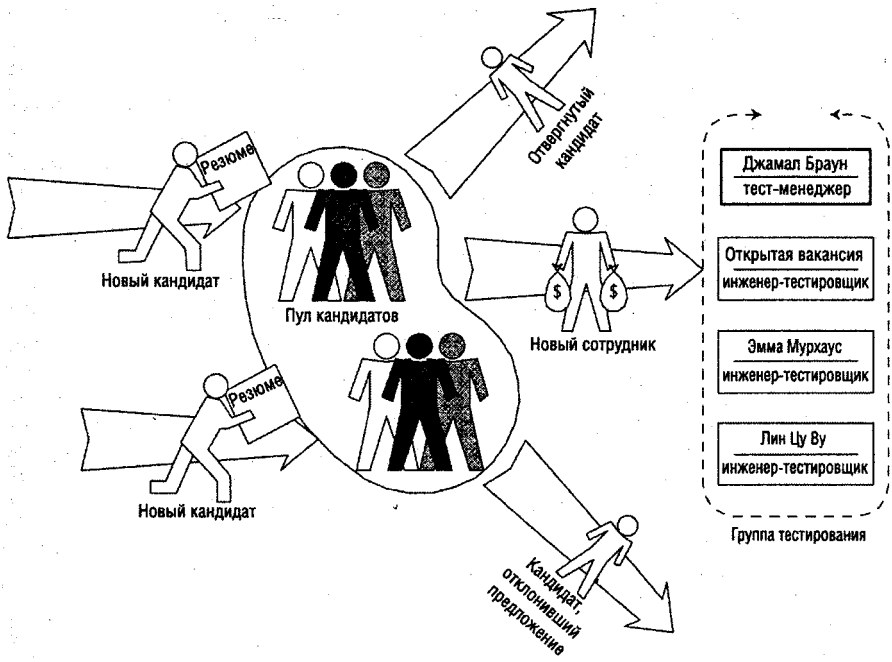


Рис. 9.1. От кандидата к сотруднику

На качество этого процесса влияют различные факторы и свойства в зависимости от того, чьими глазами вы смотрите на процесс: кандидата, работодателя или работника. С точки зрения кандидата или работника, главными вопросами являются следующие: "Это подходящая работа для меня? Удастся ли мне получить стимулы, опыт и освоить новые навыки, необходимые мне на этой работе, чтобы получить работу, которую мне захочется иметь потом?" Работодатель, в свою очередь, задается следующими вопросами: "Является ли этот кандидат подходящим для данной ра-

боты тестировщика? Способен ли этот человек совершенствовать свои навыки, чтобы помочь нам в решении новых задач?"

Определяя процесс построения команды, мы стремимся получить высококачественный процесс для всех участников, поскольку, в отличие от покера и шахмат, быть частью команды — не игра с нулевой суммой. Обе стороны хотят принять решение на основе полной информации и верных критериев. Процесс, не позволяющий какой-либо стороне получить полную информацию, может привести к одному или двум отрицательным последствиям:

1. Кандидат получает работу, о которой он позже сожалеет, что ведет к раздражению и, вероятно, непродуктивной работе.
2. Тест-менеджер делает предложение кандидату, о чем он впоследствии сожалеет, что ведет к корректирующим действиям с его стороны. Действия по коррекции отношения к делу и поведения сотрудника даже в случае успеха отвлекают внимание тест-менеджера.

Если проблемы остаются нерешенными, сотрудник либо покидает компанию со всеми вытекающими последствиями (текучесть кадров) либо продолжает работать недостаточно производительно, и, вероятно, возникнут проблемы с отношением к делу и увольнением. Лучшее, что может произойти с непрозрачным процессом, это если каждая сторона случайно найдет верное решение, что делает подход слишком непредсказуемым для столь критичных решений.

Грамотный процесс создания команды, связанный с выгодой для обеих сторон, участвующих в нем, проявляется в предоставлении всей необходимой информации. Следующие несколько разделов будут посвящены важным элементам информации как для новых сотрудников, так и для нанимающей компании. Однако также верно, что имеет значение отношение к делу и выбор философии. Компании, которые пытаются воспользоваться преимуществом работодателя с помощью вводящих в заблуждение и нечестных методов найма сотрудников, фиксирующие низкие зарплаты в периоды рецессий и использующие другие порочные тактические методы, часто пожинают плоды в виде недостаточной лояльности сотрудников и групп, которые не сделают ничего лишнего, когда компании потребуется напряжение сил для достижения цели. Кандидаты, пытающиеся получить работу, в которой они не являются специалистами, которые обманывают в своих резюме, которые не желают согласовывать совершенствование навыков с потребностями компании, часто заканчивают увольнением или застревают на неудовлетворяющих их позициях. Обе стороны, работодатель и работник, должны включаться в процесс создания команды с готовностью помогать друг другу.

Четкое определение должностных обязанностей

Взаимопомощь может иметь место только в том случае, когда обе стороны приходят к единому мнению по поводу того, какие услуги должен предоставлять работник. При отсутствии четко определенных ожиданий работодателя, работник только по случайности может выполнить свои обязанности, а работодатель не способен беспристрастно оценить качество его работы. С учетом основной цели группы тестирования мы можем определить, каким образом каждая вакансия позволяет достичь эту цель.

Определение должностных обязанностей должно быть дано с той степенью детализации, которая позволяет четко зафиксировать наши ожидания от работника и в то же время допускает свободу действий и индивидуальную оценку специалиста, соответствующую уровню работника.

Описание должностных обязанностей обычно соответствует категориям вакансий, влияние на которые оказывают навыки, образование, опыт и специализация. Обычно вакансии, открытые на низшем уровне, не требующие окончания колледжа или большого опыта, я называю младшими тестировщиками.

Поскольку эти роли не предполагают наличия большого опыта, у них нет больших возможностей для специализации. На более высоких уровнях обычно я даю следующие наименования вакансиям: инженер по ручному тестированию, инженер по автоматизированному тестированию, программист-тестировщик и администратор тестовой среды. В группы тестирования, в которых требуется высокий уровень знания предметной области, можно также включить бизнес-аналитиков или экспертов в предметной области. На любом уровне, как только сотрудники овладевают новыми навыками, слово "ведущий" может быть добавлено к наименованию их специальности, а младшие тестировщики могут становиться инженерами, программистами-тестировщиками или администраторами среды.

Если опираться на уровень детализации, который я поддерживаю, перечень должностных обязанностей, приведенный на рис. 9.2, не является адекватным. Он подходит для объявления в целях получения резюме или составления списка кандидатов, но настоящее описание должностных обязанностей, например, для письма с предложением о занятии вакантной должности, для личного дела или элемента ежегодной аттестации сотрудника, должно выглядеть примерно так, как показано на рис. 9.3.

Хотя важно прояснить ожидания работодателя, мы должны помнить, что точные способы количественной оценки должностей, навыков и обязанностей могут создать иллюзию контроля, которая не соответствует реальности. Одна менеджер, с которой я обсуждал этот вопрос, поделилась со мной, что она и ее непосредственный руководитель после принятия точного, основанного на метриках способах измерения производительности и способностей работника, решили протестировать его перед применением. Они обсудили заранее, не заглядывая в метрики, сколько очков наберет каждый сотрудник. Затем они провели измерения и, внимание! Каждый набрал примерно столько, сколько они ожидали. Они пришли к двум выводам: 1) они подогнали результаты под ожидаемые цифры; 2) эти измерения – один из ложных путей рационализации того, что они и так уже знали. Мы можем и должны стараться получить как можно более объективный процесс, но мы не можем устранить субъективность из процесса оценки кандидатов и сотрудников. Честно умерьте любые попытки строгого определения должностных обязанностей тестировщика с учетом того, что мы не готовим программу для машины, а описываем наши ожидания для человека[1].

---

[1] Для более подробного изучения вопросов управления персоналом посмотрите "*Winning at Project Management*" Роберта Гилбрета (Robert Gilbreath) или "*The Accidental Project Manager*" Патрисии Энсворт (Patricia Ensworth).

Должность:	Инженер по ручному тестированию
Назначение:	Разработка и выполнение ручных тестовых сценариев в соответствии с промышленными стандартами и со стандартами компании. Разработка и сопровождение инструментов отслеживания хода тестирования. Выполнение при необходимости других обязанностей, связанных с тестированием.

Должность:	Инженер по автоматизированному тестированию
Назначение:	Участие в процессе отбора инструментов автоматизированного тестирования. Разработка и выполнение автоматизированных тестовых сценариев с использованием промышленных инструментов тестирования и инструментов внутреннего производства. Интеграция тестовых сценариев в систему управления тестами и тестовые драйверы собственного производства. Выполнение при необходимости других обязанностей, связанных с тестированием.

Должность:	Программист-тестировщик
Назначение:	Разработка инструментов автоматизированного тестирования для программных (API) и графических (GUI) интерфейсов приложений. Поддержка и учет затрат на разработку инструментов модульного тестирования. Выполнение при необходимости других обязанностей, связанных с тестированием.

Должность:	Администратор тестовой среды
Назначение:	Выбор, установка и настройка тестовой сети, тестовых серверов, тестовых рабочих станций и тестовых баз данных (т.е. тестовой среды). Установка очередных тестовых версий в тестовой среде в соответствии с требованиями для поддержки процесса тестирования. Выполнение при необходимости других обязанностей, связанных с тестированием.

Должность:	Младший тестировщик
Назначение:	Выполнение ручных и автоматизированных тестов. Подготовка отчетов о ходе тестирования и о дефектах для инженеров-тестировщиков. Выполнение при необходимости других обязанностей, связанных с тестированием.

Рис. 9.2. Наименования должностей и роли тестировщиков

Должность: Инженер по автоматизированному тестированию

Подчиняется: Тест-менеджеру

Назначение: Участие в процессе отбора инструментов автоматизированного тестирования.
Разработка и выполнение автоматизированных тестовых сценариев с использованием промышленных инструментов тестирования и инструментов внутреннего производства.
Интеграция тестовых сценариев в систему управления тестами и тестовые драйверы собственного производства.
Выполнение при необходимости других обязанностей, связанных с тестированием.

Обязанности

A. Технические

- Проводить рецензирование требований, спецификаций, пользовательской документации, файлов помощи и другой проектной документации для проверки качества разрабатываемых продуктов и тестов, т.е. проводить статическое тестирование вручную.
- Выбирать и разрабатывать надлежащие инструменты автоматизированного тестирования, применяя самые современные методы автоматизации тестирования, например тестирование, управляемое данными.
- Использовать методы на основе рисков для разработки, сопровождения и выполнения автоматизированных тестовых сценариев для различных продуктов компании Software Cafeteria, т.е. проводить автоматизированное динамическое тестирование.
- Помогать группе разработки фиксировать и повторно использовать автоматизированные тестовые сценарии для модульного тестирования, тестовые заглушки, драйверы и другие объекты тестирования группы разработки.
- Принимать участие в заседаниях группы управления изменениями с целью установления последствий известных ошибок для качества продуктов и влияния предлагаемых изменений на определение продукта в процессе тестирования.
- Помогать группе сборки версий в создании и поддержке автоматизированного комплекта тестов для приемочного тестирования версий в ночное время.
- Обеспечивать надлежащее управление конфигурацией и контроль версий для всех разработанных объектов тестирования и используемых вариантов тестовой среды.

B. Отслеживание результатов и отчетность

- В соответствии с согласованным процессом проводить исследование и документировать отчет о дефектах немедленно после обнаружения проблемы качества.
- В соответствии с согласованным процессом обновлять информацию о ходе выполнения тестов в рамках регулярной отчетности о ходе тестирования в проекте.
- Отслеживать тестовые сценарии и результаты их выполнения относительно конкретных рисков качества.
- Помогать тест-менеджеру в подготовке отчетов о ходе работ по тестированию и измерению тестирования (например, с помощью инструментальной панели тестирования).

C. Управление и надзор

- Предоставлять тест-менеджеру тщательно проведенные точные оценки продолжительности выполнения порученных задач, а также данные о степени надежности оценок и предсказуемых зависимостей.
- Помогать тест-менеджеру в подготовке планов тестирования, бюджетов и графиков.
- Участвовать в интервью кандидатов на должности инженеров по ручному и автоматизированному тестированию и младших тестировщиков, включая подготовку и проверку контрольных заданий.
- Оказывать техническую поддержку младшим тестировщикам и начинающим инженерам-тестировщикам согласно данным поручениям.

Рис. 9.3. Подробное описание должностных обязанностей для письма с предложением о занятии вакансии и для проверки качества работы сотрудника

- Предоставлять информацию тест-менеджеру о производительности и качестве работы младшего персонала согласно данным поручениям.

D. Квалификация, образование и карьерный рост

- При отсутствии степени бакалавра поступить на утвержденные курсы для получения степени бакалавра согласно описанию в "Программе содействия развитию карьеры персонала".
- При наличии степени бакалавра начать подготовку по утвержденной программе продолжения образования согласно описанию в "Программе содействия развитию карьеры персонала".
- Совместно с тест-менеджером участвовать в ежеквартальной оценке состояния ключевых навыков группы тестирования, ставить и достигать цели повышения квалификации в трех (как минимум) согласованных разделах знаний посредством внутреннего взаимного обучения и внешних (семинары/курсы/конференции) возможностей обучения.
- Подписаться на один или более научных или отраслевых журналов по тестированию или контролю качества программного обеспечения, согласно описанию в "Программе содействия развитию карьеры персонала", и регулярно читать их.
- Приобретать ежеквартально не менее одной книги по тестированию или контролю качества программного обеспечения, читать их и готовить презентации для учебных семинаров группы тестирования согласно описанию в "Программе содействия развитию карьеры персонала".

E. Отношение к делу и самостоятельность

- Устанавливать и поддерживать хорошие рабочие отношения, особенно в рамках группы тестирования и с теми сотрудниками, которые регулярно взаимодействуют с ней.
- По согласованию с тест-менеджером направлять усилия на выполнение приоритетных задач тестирования и проекта в целом.
- Находить правильное соотношение между собственными инициативами, касающимися тестов, результатов тестирования и дефектов, и ограничениями бюджета и сроков проекта.
- Демонстрировать профессиональный пессимизм, т.е. работать с установкой на поиск, документирование и отстаивание необходимости исправления дефектов, не допуская конфликтов.
- Проявлять инициативу при определении и достижении целей в обстановке управляемых изменений.
- Понимать роль тестирования в жизненном цикле разработки программных продуктов и в рамках ограничений проекта, связанных с вопросами бизнеса, уметь защищать наилучшее из возможных мнений пользователя о системе при заданных параметрах.

Исполнитель:	**Наниматель:**
Я ознакомился и полностью понимаю должностные обязанности, описанные выше. Я также понимаю, что невозможность соответствовать ожиданиям, определенным выше, может стать причиной привлечения к дисциплинарной ответственности и/или увольнения.	Сотрудник, согласный с этой вакансией, должен подвергаться проверке и оценке на основе данной должностной инструкции, а также общеупотребительных стандартов, связанных с профессией. Никаких изменений этой должностной инструкции не допускается без согласия Исполнителя.
. . . . .	. . . . .
(Подпись)	(Подпись)
. . . . .	. . . . .
(ФИО печатными буквами)	(ФИО печатными буквами)
Должность: Инженер по автоматизированному тестированию	Должность: Тест-менеджер
Дата	Дата . . . . .

Рис. 9.3 (продолжение)

Предоставление сотруднику долгосрочного пути развития карьеры

Тестирование имеет плохую репутацию. Многим удалось построить успешную карьеру в этой области, но многие рассматривают тестирование как смертельный тупик для карьеры и пытаются выбраться из него как можно быстрее. Это представление является реальностью только в практике некоторых менеджеров ставить черную метку на тестировщиков. Однако непосредственные исполнители, которые видят тестировщиков, привязанных к своим обязанностям вопреки их желанию, обычно сами не расположены к обязанностям тестировщиков, даже на временной основе.

Например, однажды я как тест-менеджер беседовал с кандидатом на занятие вакантной должности, который не знал, на какую вакансию он проходит интервью. Он уже однажды работал тест-менеджером и описывал эту работу как свой горький опыт. Он сказал, что не займет вакансию в группе тестирования ни при каких обстоятельствах. Вероятно, никто не спросил его, чем он хочет заниматься. Просто кто-то увидел в его резюме запись про тестирование, и его привязали к этой категории кандидатов. По-моему, это очень грустно и никому не нужно.

Если посмотреть на мой опыт и на опыт слушателей моих семинаров, большинство вариантов развития карьеры для тестировщиков можно разбить на две категории: начальный этап карьеры и выбор карьеры тестировщика. В первой категории тестирование — это старт, или временная обязанность, или то и другое вместе. В случае старта люди начинают с тестирования, изучая продукт и компанию. В случае временных обязанностей люди проводят некоторое время в отделе тестирования, работая над данным проектом, а затем возвращаются к своей обычной деятельности, как правило, больше не занимаясь тестированием.

Есть две проблемы, касающиеся этих подходов. Во-первых, они недооценивают уровень квалификации, необходимый хорошему тестировщику, особенно навыки тестирования. Для занятия таких вакансий не требуется никаких навыков тестирования, поскольку вакансия навязывается, а не выбирается. Кроме того, из-за временной природы назначения отсутствуют стимулы улучшения необходимых навыков, за исключением страха лишиться возможности перейти на позиции, не связанные с тестированием, из-за низкой оценки качества работы.

Во-вторых, непрекращающаяся текучка кадров в группе тестирования существенно осложняет работу тест-менеджера. Если сотрудник получает нежелательное назначение или хочет завершить работу в этой должности, чтобы заняться действительно любимым делом, какое можно ожидать от него качество работы? Если люди непрерывно добавляются в группу, как повлияют на эффективность отсутствие экспертизы и необходимость постоянного обучения основам тестирования? Можно ли создать команду единомышленников и управлять уровнем навыков группы, если все сотрудники являются временными?

Выбор карьеры тестировщика делает тестирование отдельным техническим направлением развития карьеры в компании. Это направление должно находиться на одном уровне с программированием, поддержкой пользователей и всеми другими лестницами технической карьеры. Оно

должно также предоставлять возможность смены технического направления или перехода на менеджерское направление развития карьеры путями, аналогичными для других технических направлений. Люди должны присоединяться к группе тестирования, поскольку они хотят работать в ней, хотят оставаться в ней до тех пор, пока не достигнут объявленных и запланированных тест-менеджером целей карьеры и не займутся другой работой в компании, сохранив теплые воспоминания о времени, когда они занимались тестированием.

Подход, связанный с ротацией (см. главу 8), на самом деле является вариацией подхода к выбору карьеры тестировщика. (В небольших компаниях ротация может происходить довольно просто, поскольку в ней меньше организационных барьеров для введения ротации.) В этом случае каждый исполнитель в группе тестирования выбирает более широкий технический путь развития карьеры, приобретает знания, навыки и опыт во всех технических направлениях, связанных с разработкой и сопровождением. Тестирование, программирование, поддержка пользователя и другие технические роли рассматриваются в качестве равных и дополняющих друг друга. Люди меняются ролями, работая в одном проекте тестировщиками, в другом — программистами, в третьем — сотрудниками группы технической поддержки и т.д. Как уже отмечалось в предыдущей главе, менеджеры разных направлений должны совместно работать для того, чтобы каждый сотрудник получал все необходимые навыки в тестировании, технологиях и проблемной области для эффективной работы на всех этапах ротации.

Для меня предпочтительны подходы, связанные с выбором карьеры тестировщика и ротации. В то же время любой четко определенный путь развития карьеры лучше, чем ничего. Если отсутствует определение карьерного пути, по умолчанию группа тестирования становится тихой заводью и превращается в болото для некомпетентных и нелояльных сотрудников компании. В те времена, когда я наблюдал такие ситуации, властвовал закон джунглей. Те, кто чувствовал, что их пригвоздили к ролям тестировщиков, всеми возможными способами пытались отбиться от этих ролей. Такие действия наносили серьезный ущерб группе тестирования и ставили в тупик тест-менеджера. Если вы попали в такую ситуацию, мой совет: немедленно увольняйте нелояльных и некомпетентных сотрудников, приносящих наибольший вред группе. Затем включите в действие весь доступный вам политический капитал и всю энергию, чтобы совместно с руководством выработать какую-либо стратегию развития карьеры, которая является справедливой и приемлемой для всех участников.

Выплата сотрудникам справедливой заработной платы, соответствующей их квалификации

Избитое выражение старого менеджмента: "Деньги не являются фактором мотивации, но они могут выступать в качестве фактора отсутствия мотивации". Другими словами, нельзя побудить с помощью дополнительной оплаты кого-либо заниматься работой, которую тот ненавидит, в то же время можно вызвать отвращение к любимой работе, если ее оплата

не является достаточной. Другое важное соображение связано с тем, что зарплатная шкала должна быть справедливой и разумной внутри компании, а также должна отражать реалии местного рынка труда. Не уверен, что это справедливо для всех, но, что касается меня, удовлетворение от работы редко было линейной функцией от суммы денег, получаемой мной.

Деньги имеют значение в двух случаях. Во-первых, зарплата должна быть адекватной умениям, навыкам, образованию, а также вкладу, который сотрудник вносит в компанию. Действия компании говорят о том, что она ценит тестирование как часть направления деятельности компании, тестировщиков как личностей, вклад группы тестирования в общее дело, если она устанавливает зарплатную шкалу для тестировщиков, сопоставимую с зарплатными шкалами для других направлений развития технической карьеры.

Во-вторых, деньги имеют значение, поскольку зарплатные шкалы сами способствуют тому или иному выбору пути развития карьеры. Если зарплаты сопоставимы и существует возможность выбора карьерного пути, позволяющего переходить с направления тестирования на другие позиции и наоборот, единственным фактором, который должен рассматривать кандидат, является то, насколько он получает удовлетворение от работы тестировщика и насколько хорошо он с ней справляется. В то же время, если позиции тестировщиков оцениваются более скромной зарплатой или не дают возможностей для развития карьеры, только те люди, которые не умеют делать ничего другого, будут работать на этих позициях. Это ведет к оправданию низких заработков для тестировщиков с еще большим оправданием тупиковой ветки развития карьеры для тестирования.

Вероятно, вы сталкивались с проблемами, продвигая идею параллельной зарплатной шкалы для тестировщиков у вашего руководства. Это особенно верно, если в сочетании с отсутствием путей развития карьеры, тестирование становится парией компании. Довольно трудно аргументировать, что тестировщики заслуживают повышения зарплаты, если половина группы тестирования состоит из людей, которые, едва избежав увольнения, были направлены в тихую заводь тестирования. Однако, если вы соглашаетесь с этим статусом-кво, чтобы избежать конфликтов, вы позднее окажетесь перед лицом более серьезного конфликта и дополнительных трудностей. Если тестирование является нежелательным болотом в вашей компании, берите ситуацию в свои руки и занимайтесь ее разрешением. Начинайте бороться с самого первого дня пребывания в компании, даже до занятия должности, если это возможно, и не позволяйте проблеме загнивать.

Обзоры зарплат и отраслевые журналы рассказывают разные истории о конфликтах, связанных с паритетом зарплат тестировщиков и разработчиков. Конечно, многие обзоры зарплат тенденциозны, и содержание любой статьи, которую вы читаете, зависит от экономической ситуации, местного рынка труда, точки зрения журналиста и т.п. В некоторых случаях новые сотрудники и кандидаты сообщают о зарплатах тестировщиков на 10 — 15% ниже, чем у разработчиков. Однако другие обзоры зарплат показывают, что тестировщики больше не страдают от диспаритета

зарплат с другим техническим персоналом, когда такие факторы, как опыт работы и образование, одинаковы[1]. Можно воспользоваться этими зарплатными обзорами при разработке для компании презентации о том, что уравненная в правах профессиональная группа тестирования является шагом в будущее и что компания должна двигаться по направлению к такому будущему.

Поддержка сотрудников с достаточным образованием, опытом, умениями и навыками

В предыдущей главе рассказывалось о некоторых инструментах, позволяющих удостовериться в том, что мы берем на работу тех людей, которые нам действительно подходят. Эти инструменты доступны менеджеру, отвечающему за прием новых сотрудников. В моем арсенале ключевыми инструментами являются перечень должностных обязанностей и таблица ключевых навыков. Я применяю их с целью привлечения и найма сотрудников и для создания группы тестирования вокруг подходящих людей. Хотя пригодность людей характеризуется, естественно, несколькими параметрами, которые трудно измерить, нередко измеряемые параметры, например образование, опыт и навыки, помогают мне найти хороших кандидатов. В предыдущей главе рассматривались некоторые варианты прохождения обучения, доступные профессиональному тестировщику. Я не готов рекомендовать один набор образовательных критериев для всех тестировщиков, но предлагаю вам подумать о том, какой уровень образования подходит вашей группе тестирования с учетом общего контекста компании, долгосрочных целей развития карьеры в компании каждого тестировщика и конкретных потребностей группы тестирования в знаниях, умениях и навыках.

Я предпочитаю работать не только с квалифицированными, но и с опытными тестировщиками. Обычно мне хочется, чтобы инженеры-тестировщики имели опыт работы не менее пяти лет. Я нанимаю младших тестировщиков без опыта работы, поскольку столь недорогие ресурсы тестирования позволяют группе тестирования выполнять большие объемы регрессионного тестирования, которые в противном случае мы не смогли бы себе позволить. В то же время неопытным инженерам требуется больше времени для продвижения по карьерной лестнице. Опыт особенно важен для тестировщиков, поскольку большую часть времени занимает доведение до сведения людей информации, которую они не хотели бы знать, и работа в калейдоскопически меняющихся условиях. Люди изучают программирование в колледже, когда получают степень в области компьютерных наук, но их вряд ли учат тому, как справиться с недовольными коллегами из других групп и менеджерами.

Наконец, что касается навыков, я считаю, что полезным руководством при найме сотрудников является таблица ключевых навыков. Однако ее нужно использовать с осторожностью. Жесткое применение этого подхода может закрыть двери перед некоторыми людьми, которые могут ока-

---

[1] Ежегодные обзоры зарплат можно найти в "*Contract Professional*" и "*Quality Professional*". Также можно воспользоваться ресурсами, аналогичными www.salary.com.

заться отличными тестировщиками. Например, зачем утверждать, что каждый тестировщик должен уметь программировать, если большая часть вашей группы занимается тестированием поведения продуктов? Часто я вижу, как сообразительные люди быстро овладевают нужными навыками. Можно уточнить процесс поиска сотрудников, добавив практику ускоренного обучения в группе тестирования. В то же время кому-то из выбранных вами, но имеющих большие пробелы в знаниях по многим ключевым направлениям, нужно больше времени, чтобы стать полноценным сотрудником группы тестирования, и в течение всего этого времени потребуется обучение или другая поддержка со стороны остальных сотрудников группы.

Подходящая специализация

В некоторых случаях может оказаться, что пробелы в навыках большой группы тестирования являются результатом специализации. В нашем примере группа тестирования — это небольшая команда специалистов по общим вопросам тестирования лишь с одним различием: одни ориентированы на решение задач, связанных с автоматизированным тестированием, а другие на решение задач, связанных с ручным тестированием. Также отметим достаточно серьезные ожидания Джамала от инженеров-тестировщиков в связи с предъявляемыми требованиями относительно умений и навыков.

В небольших группах специализация опасна, поскольку каждый участник становится в определенном смысле незаменимым, по крайней мере, в короткий срок. С точки зрения оценки навыков, ключевыми являются минимальные значения уровня развития навыков, поскольку каждый из специалистов должен быть компетентным в различных областях знаний. Предположение, что в ходе проекта никто не заболеет, не забеременеет или не будет переживать несчастье, например, связанное со смертью родственников, является весьма рискованным. Но мы как раз неявно делаем такое предположение в случае специализации в небольших группах тестирования.

Для больших групп тестирования, вероятно, имеет смысл более развитая специализация. В одном из проектов, где я был тест-менеджером, тестировалось Интернет-приложение. Мы разбили тестирование на две части: серверную и клиентскую сторону. Тестировщики серверной стороны сконцентрировали усилия на проверке большого серверного кластера, который работал с электронной почтой, выполнял обновления программ и другие сервисы. Тестировщики клиентской стороны занимались устройствами доступа в Интернет.

Я взял двух инженеров-тестировщиков, одного для серверной стороны, второго для клиентской стороны. Тестировщик серверной стороны имел опыт работы с UNIX и автоматизации тестирования, тестировщик клиентской стороны обладал хорошим опытом работы с почтовыми системами, браузерами и Интернет-технологиями. Также я нашел инженера по тестированию удобства использования. Поскольку нужно было моделировать 50 000 пользователей, я принял на работу двух программистов-тестировщиков. Они покончили с многократным использованием боль-

шого количества заглушек и драйверов для модульного тестирования, разработав генераторы загрузки, которые дали возможность тестировать серверы и отклики клиентских машин, если те были соединены с загружаемыми серверами, на желаемом уровне загрузки. Поскольку мы использовали большое количество тестовых сценариев ручного тестирования, инженеры-тестировщики подготовили скрипты, выполнять которые было поручено шести младшим тестировщикам. Эта группа, будучи отражением проекта, работала очень слаженно.

Некоторые группы тестирования вынуждены обслуживать сразу несколько проектов. Например, один из моих клиентов располагал единственным, очень большим отделом тестирования, который назывался Центр тестирования отличного качества (Testing Center of Excellence). Они считали, что тестирование есть специализация внутри себя. В таких больших сервисных отделах группа управления тестированием может выбрать различные пути организации. Один путь – на основе проектов, когда тестировщики назначаются в единственный проект на все время его выполнения. Второй путь – на основе квалификации, когда тестировщики вводятся в проект и выводятся из него по мере необходимости навыков, присущих именно им. В этом случае вероятно, что в проекте работает на постоянной основе только тест-менеджер или ведущий тестировщик[1]. Еще один вариант организации – на основе предоставляемых сервисов, вокруг конкретных услуг, которые предоставляет компания[2].

Кандидат, который прекрасно подходит для организации группы тестирования на основе квалификации, может не подходить для организации на основе проектов. Некоторые тестировщики действительно хотят развиваться в определенных областях технологий. Только организация группы тестирования на основе квалификации позволяет такую специализацию. Аналогично, кандидат, подходящий для проектной организации группы тестирования, может не подойти для организации на основе квалификации. Например, однажды я работал с инженером-тестировщиком, который абсолютно не справлялся с работой в случае, если отвечал за выполнение нескольких задач одновременно. Он рассматривал назначение в несколько проектов одновременно как неподъемную для него обязанность.

Отличная группа тестирования станет за одну ночь слабой, если волевым решением руководства будет осуществлен переход от одной модели к другой. Я также наблюдал, как одна заурядная группа тестирования стала отличной в результате перехода к модели на основе квалификации и культивирования экспертизы и специализации. Я уверен, что обе модели могут работать, если руководство понимает суть специализации и ее влияние на группу, когда принимает решения по поводу процесса создания группы.

---

[1]. Подробнее об этих и других вопросах управления группой тестирования рассказывается в главе 8 книги "*Managing the Testing Process, Second Edition*".

[2] Подразделения тестирования, организованные на основе предоставляемых сервисов, обсуждаются в статье "At Your Service" Роберта Сабурина (Robert Sabourine), впервые опубликованной в "*Software Testing and Quality Engineering*", Volume 3, Issue 3 и доступной в настоящее время на www.stickyminds.com.

Решение проблем

Даже когда все вопросы, связанные с подбором кадров решены, могут возникать различные проблемы.

Что должны уметь тестировщики и что вы можете получить?

В мире тестирования продолжаются споры о том, какие из навыков являются ключевыми для тестировщиков. Я разбил все навыки на четыре категории: общие, технические, знание предметной области и тестирование. Я уверен, что относительное значение каждой из категорий зависит от того, какая система тестируется, включая ее технический аспект и бизнес-аспект. Однако некоторые люди выступают в защиту повышенной технической экспертизы, на том же уровне, что и у группы разработки, в качестве необходимого условия для всех тестировщиков. Другие выражают скепсис, если группа тестирования составлена из людей, не являющихся специалистами в проблемной области приложения. Как решить, какие и какого уровня навыки нужны квалифицированному тестировщику?

Есть три аспекта этой проблемы. Во-первых, остается вопрос определения компетентности. Что тестировщик должен знать, чтобы эффективно и качественно выполнять свои обязанности? Требования в терминах технических знаний часто являются функцией того, какие объемы задач структурного тестирования (белого ящика), контроля качества (просмотр кода) и автоматизации тестирования будет выполнять тестировщик. Необходимые знания предметной области зависят от сложности, неизвестности и рисков, относящихся к проблеме бизнеса, которую призвана решать система. Текстовый процессор — это понятное приложение, а система разведки нефтяных месторождений далеко не очевидна. Лишь виртуальные проблемы могут возникнуть в случае краха видеоигры, тогда как от надежности систем управления вооружениями зависят человеческие жизни.

Однако я встречался с ситуациями, когда тестировщик должен был (или мог) проводить только тестирование поведения системы (черного ящика) в некритичной для безопасности среде. Но когда я предложил воспользоваться услугами экспертов по тестированию или младших тестировщиков, людей с минимальными знаниями проблемной области и техническими навыками, реакция была отрицательной. Один из моих предполагаемых клиентов не хотел использовать непрограммистов в качестве тестировщиков для его видеоигры, поскольку он предполагал, что группа тестирования будет проводить и структурное и поведенческое тестирование. Когда я предложил использовать менее дорогих младших тестировщиков для поведенческого тестирования и нанять одного или двух высокооплачиваемых человек с техническими навыками и навыками тестирования, он отказался. Один из клиентов, менеджер центра обработки звонков по ссудам, скептически отзывался о тестировании приложения обработки ссуд, поскольку его проводили профессиональные тестировщики, а не банкиры центра. Этот скептицизм не исчез даже после того, как мы обнаружили серьезные ошибки в продукте.

Эти примеры иллюстрируют вторую сторону рассматриваемой проблемы. Поскольку тестирование – это создание новой информации, доверие к группе тестирования является ключевым фактором успеха. Независимо от того, насколько компетентны тестировщики, работающие с продуктом, если есть мнение ключевых участников разработки, что они некомпетентны, группе тестирования не будут доверять. В таком случае группа тестирования имеет большие шансы на неудачу в проекте и роспуск, даже если работа проведена отлично.

Таким образом, поскольку мы считаем, что тестировщики не только должны быть компетентными, но и слыть компетентными среди ключевых участников разработки, мы рассматриваем третью сторону проблемы. На самом деле непросто найти тестировщиков, которые хорошо знают проблемную область, имеют опыт работы с применяемыми технологиями, а также обладают отменными навыками тестирования и правильным отношением к делу. Можно повысить вероятность решения проблемы, если есть возможность нанять консультантов или контрактников, поскольку многие из тех, кто обладает всеми тремя наборами качеств, не доступны на свободном рынке труда и требуют огромной зарплаты в соответствии со своей квалификацией.

В то же время трудно убедить руководство финансировать группу тестирования, состоящую из дорогих консультантов. Кроме того, поскольку есть намерение создать постоянную группу, консультанты и контрактники смогут оказать помощь в ее совершенствовании, но не нужно создавать долговременной зависимости от этих ресурсов, если вы не рассчитываете убедить их стать постоянными сотрудниками. Таким образом, перед вами встает проблема создания компетентной группы, которая по средствам компании и которой доверяют.

Момент, когда вы должны добиваться баланса, зависит от ситуации. Нужно работать над этим вместе с руководством и другими ключевыми участниками проекта по мере того, как вы формулируете перечень должностных обязанностей, находитесь в непосредственном контакте с ключевыми участниками и вашими руководителями. Разберитесь с проблемами доверия и доступности средств, связанными с тем, насколько высоко будет поставлена планка. Убедитесь, что все принимают компромиссные решения, которые вам приходится предлагать. Не начинайте проект даже с безупречно компетентной группой тестирования, если неизвестны (иногда нереалистичные) ожидания других участников проекта относительно навыков группы или возврата вложений в тестирование.

Белые слоники

Когда я был маленьким, мама и бабушка иногда брали меня в походы по антикварным магазинам. Антикварные вещи были тогда менее модными, чем сейчас, и их часто называли "подержанными вещицами", "вещами, передаваемыми по наследству" и даже "белыми слониками". Последняя фраза всегда смущала меня (мама объяснила, что она происходит от большого числа глиняных статуэток, которые можно было найти в любом антикварном магазине), но она сохранилась в памяти до сих пор.

Иногда кажется, что тест-менеджеры заполняют группу тестирования в магазинах, продающих белых слоников. Одна из групп тестирования, которыми я руководил, состояла исключительно из людей, которым не удалось стать программистами. Один из них оказался отличным инженером-тестировщиком, а двое других не продемонстрировали больше способностей к тестированию, чем к программированию. Мне разрешили их уволить, но без гарантии, что удастся заполнить вакансии.

Сейчас я стал старше и умнее и научился сопротивляться набегам белых Слоников[1]. Однажды менеджер разработки, узнав, что один из инженеров сборки решил уйти, настаивал на том, чтобы я взял его в группу тестирования. Я отказался. Другой случай: бизнес-аналитик привел безработного родственника на интервью, поскольку он услышал, что группа тестирования ищет сотрудников. Из резюме этого человека было ясно, что он не обладает квалификацией. Чтобы окончательно убедиться в этом, я задал несколько провокационных вопросов на групповом интервью, на котором присутствовали коллеги бизнес-аналитики. Эти вопросы деликатно, но однозначно продемонстрировали отсутствие у кандидата навыков и опыта в ключевых разделах, которые мы хотели закрыть с приемом нового сотрудника.

Чтобы вы не сочли меня параноиком, позвольте заметить, что в своем мнении по этому поводу я не одинок. Борис Бейзер (Boris Beizer) писал: "Я получил свою дозу "бедных-несчастных", некомпетентных, невротиков, психов, [и] алкоголиков"[2]. Сем Канер (Cem Kaner), Джек Фолк (Jack Falk) и Енг Нгуен (Hung Nguyen) согласны, что тест-менеджеры должны сопротивляться усилиям наполнить группу тестирования программистами-неудачниками[3]. В группу тестирования должны приниматься сотрудники, которые хотят в ней работать и навыки, знания и личные качества которых подходят для группы. Чем включить в группу тестирования негодяя, лучше пусть у вас будет на одного человека меньше, чем требуется.

Итак, программист может попасть в неприятности по целому ряду причин, не последней среди которых является слабое руководство. Если у человека есть подходящие навыки и знания и его считают компетентным в группе тестирования, я не стал бы рассматривать факт, что он был гадким утенком в другой группе, как препятствие, мешающее принять лебедя в мою команду.

Использование контрактников и консультантов

В некоторых случаях верным решением для группы тестирования является не прием нового сотрудника, а приглашение контрактника или консультанта. Для временных потребностей, например в ходе пиковых загрузок тестировщиков в проекте, использование контрактников имеет большой смысл. Я выполнил приличное число проектов с большой за-

---

[1] Роберт Сабурин (Robert Sabourine), один из рецензентов, предоставил мне ссылку на web-сайт, phrases.shu.ac.uk/meanings/410050.html, так что теперь я знаю точное значение и происхождение термина "белые слоники".

[2] Boris Beizer, "*Software System Testing and Quality Assurance*", page 317.

[3] Cem Kaner, Jack Falk and Hung Nguyen, "*Testing Computer Software*", page 359.

грузкой, когда начинал работать с группой контрактников, по ходу проекта проводил тестирование, а затем осуществлял переход к постоянной группе, часто помогал при наборе персонала и иногда превращал часть контрактников в постоянных сотрудников компании-клиента. Такой подход отлично работал и для меня, и для моих клиентов. Вероятно, вы захотите воспользоваться услугами консультанта для проведения обучения или нечастых советов по трудным проблемам.

Перед тем как нанять консультанта или контрактника, необходимо определить, для чего вы нанимаете этого человека. Я часто получаю телефонные звонки и электронные сообщения от людей, которые можно сформулировать примерно так: "У меня есть ряд непонятных организационных трудностей, связанных с тестированием, и я бы хотел, чтобы вы помогли их устранить". В некоторых случаях дальнейшие уточняющие вопросы показывают, что указанные проблемы не имеют отношения к тестированию, а являются проблемами управления, которые проявились в ходе одной из фаз тестирования.

Если вы можете четко сформулировать, что должен делать контрактник или консультант, вам обоим понравятся результаты работы, в противном случае — берегитесь. Например, предположим, что вы сообщили представителю группы продаж компании по поиску временных сотрудников: "Тестирование требует очень много времени, мне нужно автоматизировать его часть". Практически сразу ваш факс переполнится резюме всех кандидатов, которые когда-либо видели инструмент автоматизированного тестирования, трогали его или были в той же комнате, где он находился. Если вы примете на работу кого-нибудь из этого пула кандидатов только на основе продолжительности опыта по автоматизации тестирования или потому, что они знают инструмент, который вы планируете приобрести, отрицательный результат очень вероятен.

Напротив, предположим, что вы позвонили своему приятелю агенту по подбору кадров, живущему по соседству, и сказали: "Послушай. Мы решили, что у нас слишком много проблем с регрессионным тестированием в версиях сопровождения. Вместо того чтобы нанимать еще десяток тестировщиков для проведения ручного тестирования, мы бы хотели вложить средства в подробный комплект автоматизированных тестов. Поэтому мне нужен кто-то, чтобы помочь выбрать инструмент. Этот человек должен иметь опыт работы с разнообразными инструментами и не бояться осваивать новые. Как только инструмент будет выбран и развернут, нам потребуется специалист, который поможет разработать этот комплект тестов, а затем обучит одного из старших тестировщиков, как применять тесты и сам инструмент. После этого контрактник должен сразу завершить свою работу в компании или постепенно свернуть свое участие в течение нескольких недель. Одним из показателей успешности его работы является то, насколько он может сделать процесс работы с тестами и инструментами независимым от его участия. Мы планируем, что он будет работать примерно шесть месяцев, но мы открыты для переговоров с кандидатом по этому вопросу. Наконец, каждый кандидат, которого вы направляете на интервью, должен предоставить три рекомендации от компаний, в которых он выполнял проекты, аналогичные нашему. Понятны ли наши требования, есть ли у тебя вопросы?"

Оценивая квалификацию консультантов и контрактников, вы должны ответить на целый ряд вопросов. Вы часто ищете специалистов с квалификацией, превосходящей вашу собственную. Как можно узнать, что вы говорите с экспертом, а не с умеющим себя подать неквалифицированным человеком? Как вы поймете, что контрактник или консультант, который получил сертификат в некотором разделе технологий, предметной области или тестировании, может применить и реально применял свои знания в реальном проекте? Как вы узнаете, что контрактник, который рекламировался агентом по персоналу в качестве идеального кандидата, не превратится в вашу головную боль через несколько дней после того, как агент получит комиссионные. Понимаете ли вы проблему, которую должен решить контрактник, достаточно хорошо для того, чтобы представить, кто должен ее решать, и понять, решал ли он ее когда-то в прошлом?

Квалифицированные консультанты и контрактники имеют большой перечень отменных рекомендаций, которые сообщают о хорошо выполненной ими работе. Приглашения специалистов обычно имеют свою специфику, однако я обращаю внимание на навыки, отношение к делу и образование, которые я определил как существенные для моей группы тестирования. Как контрактник применил навыки автоматизации тестирования для решения реальных проблем разработки или сопровождения? С какими видами проблем он уже сталкивался в предыдущей работе? Способствовало ли образование кандидата качественному выполнению работы?

Помимо рекомендаций, вероятно, кандидат оставил за собой право использовать один или несколько результатов, достигнутых у предыдущих клиентов, в качестве примера его работы. Предусмотрительные кандидаты и контрактники обычно включают в свои контракты условия, которые позволяют ему после удаления идентификационной информации использовать примеры своей работы.

Квалификация также может быть продемонстрирована с помощью списка публикаций или лекций по тестированию, прочитанных по приглашениям. Известные эксперты рецензируют большинство научных и отраслевых журналов и книг, посвященных тестированию. Хотя публикуются сомнительные и иногда спорные статьи, подавляющее большинство людей, незнакомых с тестированием, непонимающих его процессов и проблем, сочтут невозможным публиковаться. Электронные форумы, в частности Интернет-телеконференции и обсуждения, также могут показать знание тестирования, хотя отсутствие экспертных оценок иногда приводит к тому, что гуру и болтун отнимают одинаковую долю внимания аудитории.

Если разрешено местными законами, еще один способ гарантировать качественный подбор консультанта или контрактника для решения ваших проблем — испытательный срок и внесение в контракт условий его прерывания. Я не стану рекомендовать ничего непорядочного, хотя некоторые действительно пытаются необоснованно прервать контракт с консалтинговыми фирмами. Напротив, я говорю о том, чтобы определить измеряемые контрольные точки для оценки работы консультанта или контрактника. Эти контрольные точки должны помочь провести

оценку хода выполнения работ и их качества. Если работы не выполнены, контракт должен давать возможность справедливо, по взаимному согласию, но достаточно быстро для вас как заказчика прекратить действие контракта и уволить консультанта.

Политика моей собственной фирмы как консалтингового учреждения состоит в том, что мы хотим получать деньги только с клиентов, довольных нашей работой, поскольку мы получаем рекомендации и отзывы о нашей работе. Поэтому мы включаем в наши контракты условия прерывания. Другое преимущество таких условий проявляется следующим образом. Дополнительно гарантируя, что конкретная работа будет выполнения в соответствии с графиком, условия прерывания контракта предотвращают бурную реакцию заказчика в случае небольших инцидентов, а это не оказывает влияния на главную причину, по которой заказчик нанимает контрактника. Если прерывание контракта инициируется при особых условиях, четко прописанных в контракте, это способствует тому, что заказчик концентрируется на той работе, для выполнения которой приглашен контрактник или консультант.

Наконец, позвольте отметить, что сложный и иногда болезненный процесс приглашения консультантов и контрактников может быть существенно упрощен за счет создания и последующего использования обширных связей в сообществе тестировщиков программного обеспечения. Если вы читаете сообщение в телеконференции или статью в журнале о тестировании, которая задевает вас, то, вероятно, обнаружите, что ее автором является консультант, специализирующийся как раз на той области знаний, в помощи в которой вы нуждаетесь. Поговорите с коллегами, которых вы встречали на учебных семинарах и конференциях, чтобы узнать, чьими услугами они пользовались. Узнайте про профессиональную репутацию как самих консультантов, так и компаний, в которых они работают.

Внедрение усовершенствований

Как уже упоминалось в начале предыдущей главы, процесс создания группы тестирования очень зависим от конкретной ситуации и подвержен разнообразным влияниям. Следовательно, особенности совершенствования этого процесса будут зависеть от того, что вы делаете сейчас и что возможно сделать в рамках вашей компании. Тем не менее позвольте высказать несколько идей, которые, как я заметил, получили широкое применение.

1. Обеспечьте совместимость контекста тестирования и группы тестирования. Еще один результат процесса определения контекста тестирования, представленного в главе 1, состоит в том, что он позволяет понять, ту ли команду вы формируете. Как наилучшим образом вы можете обеспечить потребности компании?
2. Внедрите проведение оценки навыков и управление ими. Если вам известно, как обеспечить потребности компании, можно приступить к определению перечня ключевых навыков, необходимых для выполнения этих задач. Можно составить таблицу оценки навыков.

Если в группе тестирования уже есть сотрудники, привлеките их к процессу создания списка ключевых навыков.

3. Подготовьте и предложите пути развития карьеры для тестировщиков, постоянно работающих в компании. Определите средства для карьерного роста в рамках группы тестирования. Совместно с менеджером по работе с персоналом или коллегами-менеджерами других групп определите пути развития карьеры с начальных позиций до других позиций в компании.
4. При проведении интервью с кандидатами применяйте оценку навыков, вопросы о пути развития карьеры, о линии поведения и контрольную проверку знаний, чтобы подобрать людей с подходящими навыками, опытом, отношением к делу и способностями.

Создание и совершенствование групп тестирования — это тяжелая работа, но усилия, потраченные на нее, возвращаются сторицей. Быть частью сильной команды, и в роли менеджера и в роли непосредственного исполнителя, — одно из тех ощущений профессионала, которые приносят наибольшее удовлетворение от работы. Последние две главы были посвящены видению этого процесса со стороны менеджера, поскольку львиная доля ответственности падает на него. Однако непосредственные исполнители могут и должны принимать в нем участие, особенно в совершенствовании старых и приобретении новых навыков и в уточнении их отношения к делу и ожиданий.

Имея хорошо организованную и подготовленную группу тестирования, мы можем перейти к следующей стадии: к созданию системы тестов. Имея в наличии хорошую группу тестирования и хорошую систему тестов, мы сможем приступить к выполнению тестов.

ГЛАВА 10

Архимедова ванна: проектирование и разработка системы тестов

Легенда гласит, что греческий царь Гиерон приказал изобретателю и математику Архимеду придумать метод определения качества императорской короны. Для императора качество означало наличие в короне единственного элемента — чистого золота, и отсутствие многих нежелательных элементов — примесей других металлов. Архимед не знал, что делать, пока однажды он, сидя в ванной, не заметил, как вода переливается через край. Открывая закон погружения тел, названный впоследствии его именем и позволивший определить качество короны, он воскликнул: "Эврика!" Мы, тестировщики, возможно, тоже должны забраться в архимедову ванну, поскольку наша задача — проектирование и разработка эффективной и качественной системы тестов, которая позволяет оценить качество системы, которую мы тестируем.

Оценка качества имеет две основные цели, связанные с проектированием и разработкой системы тестов. Во-первых, мы должны нацеливать систему тестов на проверку тех рисков качества системы, которые считаются наиболее критичными на основе их относительной значимости (см. главу 2). Во-вторых, если мы хотим оценить качество, мы должны организовать тестирование так, чтобы тестировщики могли четко понимать, каким должно быть мнение пользователей и заказчиков о системе. Следовательно, мы должны предоставить тестировщикам высококачественную систему тестов, которая верно воспроизводит мнение заказчиков и пользователей о системе.

Моя модель системы тестов состоит из трех основных элементов, как показано на рис. 10.1. Глядя на рисунок, можно сделать вывод о формальных требованиях к системе тестов, однако это не совсем так. Проектирование и разработка системы тестов иногда начинаются за несколько лет до того, как система будет предоставлена группе тестирования, а иногда эти два процесса стартуют в один и тот же день. Иногда система тестов размещается в большой тестовой лаборатории на гигабайтах дискового пространства, в толстых папках с тремя кольцами, а иногда она целиком существует только в голове тестировщиков.

Тестовая среда (Test Environment)

Аппаратное и программное обеспечение, оборудование и помещения, необходимые для проведения тестирования.

Средства тестирования (Testware)

Тестовые сценарии, тестовые данные, тестовые скрипты, инструменты тестирования, протоколы тестирования, отчеты о тестировании, а также соответствующие руководства пользователя и документация.

Процесс выполнения тестов (Test Execution Process)

Методы, при помощи которых тестировщики и другие ключевые участники процесса тестирования используют тестовые программы в тестовой среде.

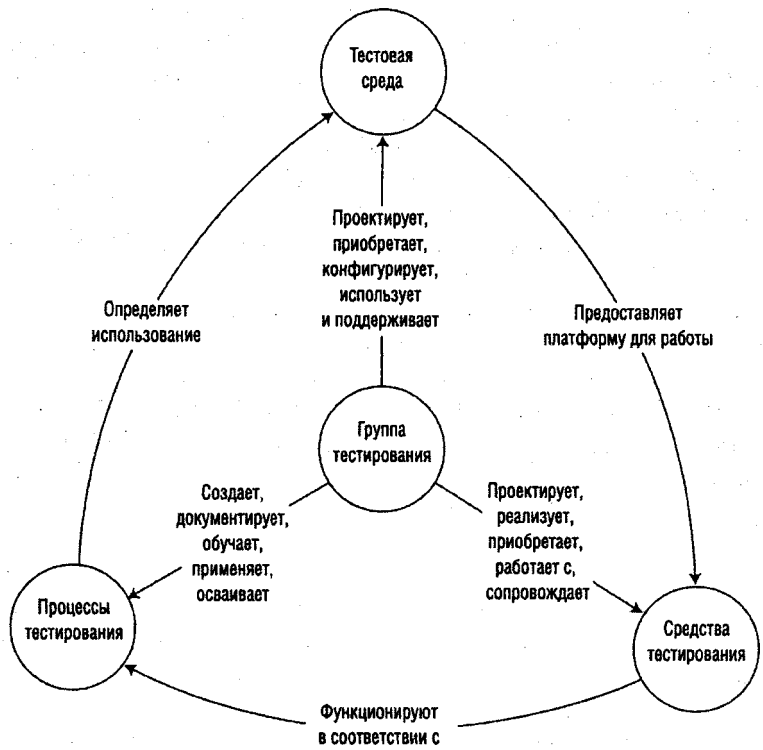


Рис. 10.1. Элементы системы тестов

Проектирование тестов происходит там и тогда, где и когда тестировщики расходуют время на принятие решений о том, как тестировать конкретные области системы и как искать в ней конкретные проблемы. Эффективное и качественное проектирование тестов означает формулирование точных вопросов по поводу того, что может сломаться.

Разработка тестов происходит там и тогда, где и когда тестировщики расходуют время на реализацию принятых при проектировании тестов решений. Эффективная и качественная разработка тестов — это построение системы тестов, которая отвечает на точные вопросы о том, что ломается и что не ломается, в порядке приоритетов рисков качества системы и путем, который непосредственно связан с мнением пользователей и заказчиков о системе.

Эти комбинированные усилия по проектированию и разработке тестов могут быть зафиксированы в любой форме, начиная от заметок, эскизов и моделей, выполненных на доске, и заканчивая кипами связанных тестовых сценариев и скриптов вместе с дисками и лентами, заполненными данными. Эффективное и качественное проектирование и разработка системы тестов могут потребовать и минуты, и месяцы. Различные подходы к учету соображений, касающихся контекста (см. главу 1), определяют степень формализации и неизменности, а также структуру системы тестов.

Процесс проектирования и разработки системы тестов

Конкретные детали процесса проектирования и разработки системы тестов существенно различаются для разных стратегий тестирования, контекста группы тестирования и технологий, поддерживающих систему тестов и используемых в системе, предназначенной для тестирования. Процесс 7 является общим процессом, который каждый может приспособить в соответствии с собственными потребностями.

Эмма создает напряжение в работе

В ходе фазы планирования Джамал поручил Эмме Мурхаус обновить тесты для проверки производительности, нагрузки, мощности и объема с учетом новых свойств, которые добавляются в Суматру. 11 ноября, на один день раньше срока, Эмма занялась этим.

Наиболее критичные риски качества возникают в поведении системы, когда достигаются границы объема данных и загрузки. Это как раз относится к комплекту тестов для проверки производительности, нагрузки, мощности и объема. (На рис. 10.2 показаны связанные с этими проблемами риски качества.) Система должна поддерживать 255 пользователей, подключившихся одновременно, и всего 32 767 учетных записей пользователей, позволяя управлять документами размером до 32 мегабайтов. Если оперативная память заполнена целиком, файлы должны направляться в почти оперативную память, а затем в автономное хранилище. Файлы, помещенные в почти оперативную память и в автономное храни-

лище, должны возвращаться в оперативную память при обращении к ним пользователей, замещая более старые файлы и направляя их в более медленные хранилища, чтобы освободить необходимое дисковое пространство. Должна работать миграция файлов из одного хранилища в другое. Причем это должно происходить с максимальной прозрачностью для пользователей и с быстрой эскалацией блокирующих ситуаций в библиотекаре документов (например, в устройстве чтения нет компакт-диска, на котором находится файл с данными), с 99%-ным временем работоспособности системы и без потери данных.

Процесс 7. Процесс проектирования и разработки системы тестов

№ этапа	Этап	Выполнено?
1.	Создать или обновить комплекты тестов, покрывающих те риски качества системы, для которых рекомендовано тестирование.	☐
2.	Выбрать надлежащий комплект тестов и придумать новое множество интересных тестовых условий в рамках наиболее критичного раздела системы, который еще не покрыт тестами, т.е. условий, которые могли бы спровоцировать определенный набор проблем или смоделировать практическое использование системы. Определить на верхнем уровне детализации, как проверить корректность результатов при данных тестовых условиях.	☐
3.	Выбрать надлежащие методы тестирования для исследования тестовых условий и верификации ожидаемых результатов с учетом доступных времени и бюджета, тестопригодности системы, предназначенной для тестирования, тестовой среды и квалификации группы тестирования.	☐
4.	Спроектировать, разработать, приобрести, усовершенствовать, сконфигурировать и документировать средства тестирования, тестовую среду и процесс выполнения тестов для воспроизведения определенных условий и проверки поведения системы с использованием выбранных методов тестирования. Усилить точность методов тестирования с помощью эмпирических методов с учетом данных заказчика или пользователей, отчетов о проблемах в эксплуатации (для ранних и конкурирующих версий, аналогичных продуктов и аналогичных наборов рисков качества системы), прогнозов отделов продаж и маркетинга о будущем использовании системы и других данных заказчика.	☐
5.	Если какие-либо элементы средств тестирования, тестовой среды или процесса выполнения тестов оказались недоступными в ходе выполнения этапа 4 из-за ограничений ресурсов, сроков и т.п., повторить этап 4 нужное число раз, чтобы приспособиться к этим ограничениям.	☐
6.	Проверить систему тестов, используя статические (например, рецензирование) и динамические (например, выполнение тестов) методы.	☐

Процесс 7 *(продолжение)*

№ этапа	Этап	Выполнено?
7.	Поместить средства тестирования, описание процесса выполнения тестов, информацию о конфигурации тестовой среды, документацию о тестировании системы тестов (полученную на этапе 6), всю прочую документацию и созданные файлы в библиотеку проекта или в систему управления конфигурацией. Привязать версию системы тестов к версии системы, предназначенной для тестирования. Вести контроль изменений.	☐
8.	Обновить перечень рисков качества на основе новых знаний, полученных при разработке последних наборов тестов. Оценить покрытие рисков качества новой системой тестов. Выявить остающиеся непокрытыми риски качества, для которых рекомендовано тестирование.	☐
9.	Повторить этапы 2 – 8, пока не будут покрыты все риски качества, для которых рекомендовано тестирование. Убедиться, что документация по рискам качества обновлена в проектном репозитории. Если ограничения бюджета и сроков вынуждают прекратить разработку тестов, подготовить отчет о непокрытых рисках качества и направить его руководству.	☐
10.	Если время и ресурсы позволяют, повторить этапы 2 – 8, используя структурный анализ (например, сложность Маккейба (McCabe complexity) и покрытие кода), чтобы определить области системы, требующие дополнительного тестирования.	☐

Тестовое состояние (Test Condition)

Состояние системы, выходные данные, результат или условие, порождаемые выполнением тестового сценария, преимущественно интересные с точки зрения оценки качества системы.

Тестовый сценарий (Test Case)

Последовательность шагов, состоящих из действий, которые необходимо выполнить при помощи системы, предназначенной для тестирования. (Эти шаги обычно называют тестовой процедурой или тестовым скриптом.) Такого рода действия обычно увязывают с некоторым набором данных (заранее загруженным или вводимым в процессе выполнения сценария). Комбинация выполняемых действий и данных, передаваемых системе в процессе выполнения теста, приводит систему к некоторому тестовому состоянию. В тестовом состоянии система выдает результаты, которые можно сравнить с ожидаемыми, т.е. оценить качество в данном тестовом состоянии. Такие действия могут выполняться последовательно, параллельно или в различных вариациях.

Набор тестов (Test Suite)

Совокупность логически связанных тестовых сценариев. Те или иные тестовые сценарии могут содержаться в нескольких наборах тестов.

№ этапа	Этап	Выполнено?
1.	Создать или обновить комплекты тестов, покрывающих те риски качества системы, для которых в рекомендовано тестирование.	☑
2.	Выбрать надлежащий комплект тестов и придумать новое множество интересных тестовых условий в рамках наиболее критичного раздела системы, который еще не покрыт тестами, т.е. условий, которые могли бы спровоцировать определенный набор проблем или смоделировать практическое использование системы. Определить на верхнем уровне детализации, как проверить корректность результатов при данных тестовых условиях.	☑

Эмма изучила документы с описанием требований, посмотрела в потолок, прошлась по офису, пролистала пару книг по тестированию, а также материалы конференций в поиске идей. Наконец она пришла к решению использовать три метода.

Во-первых, в качестве основы для тестирования она решила использовать подход "*мыльной оперы*" Ханса Бувальды (Hans Buwalda), построив очень большой, но поддающийся сопровождению сценарий. Предположим, рассуждала она, есть большая группа адвокатов, защищающих несколько больших транснациональных корпораций против определенного вида истцов. В частности, представим множество адвокатских контор с общим числом служащих в 32 767 человека, участвующих в большом судебном процессе, начатом против их клиентов, которые находятся в различных точках земного шара. 7 дней в неделю 24 часа в день 255 адвокатов будут подключены к системе, создавая документы, делая на них ссылки и архивируя их с помощью блока управления документами. Документы варьируются по размеру от 0 до 32 мегабайтов, поэтому сделаем избыточное предположение, что средний размер документа равен 16 мегабайтам при выборе документов случайного размера, близкого к среднему, и имеется, по крайней мере, по одному документу с размером, соответствующим одному из предельных значений. Это модель, возможно, далека от реальности, однако она соответствует требованиям и заставляет систему непрерывно работать на пределе возможностей[1].

Во-вторых, в ходе разработки функциональных тестовых сценариев Лин Цу обсуждала с Эммой, как движок редактирования взаимодействует с системой управления документооборотом, вынуждая ее проходить через разные состояния. Например, когда SpeedyWriter начинает работу, нет открытых файлов, но затем пользователь может запросить файл из системы документооборота, что приведет к тому, что система представит перечень известных ей документов. Теперь пользователь может выбрать документ или добавить новый документ в систему. Каждое действие переводит систему в новое состояние.

Как только пользователь начинает редактировать документ, управляемый системой управления документооборотом, периодические сохране-

[1] См. статью Ханса Бувальды (Hans Buwalda) "Soap Opera Testing", представленную на конференции SATR East 2000, опубликованную в "*Proceedings of the STAR East 2000*" и доступную в настоящее время на www.stickyminds.com.

	A	B	C	D	E	F	G	H	I
1	Failure Mode and Effect Analysis (Quality Risk Analysis) Form								
3	System Name: SpeedyWriter (with Document Management System add-on)						Model/Product: SpeedyWriter Release 3.1 (w/ DMS)		
4	Product Manager: Kate Hernandez						Target Release Date: 1Q2003		
5	Project Manager: Jenny Kaufman						Prepared By: Jamal Brown		
6	Other Stakeholders: Bobbi McDaniels (Engr), Max del Oro (Mkt), Cosimo Negraev (Support)						FMEA Date: September 11, 2002		
7	Test Manager: Jamal Brown						FMEA Revision Date: September 20, 2002		
9	Risk ID Number	Quality Risk Category	Failure Mode/Quality Risk/Effect	Severity	Priority	Likelihood	Risk Priority Number	Recommended Action	Who/Which Phase (Unit, Component, Integration, System)?
25	2.000	Load, Capacity, and Volume	Failures in scaling of system to expected peak concurrent usage levels						
26	2.001		System fails at or before 25 concurrent users.	1	1	3	3	Extensive testing.	Test/S
27	2.002		System fails at or before 255 concurrent users.	1	3	2	6	Extensive testing.	Test/S
28	2.003		System disallows 255 or fewer user accounts.	1	1	3	3	Extensive testing.	Test/S
29	2.004		System disallows 32,767 or fewer user accounts.	1	3	3	9	Balanced testing.	Test/S
30	2.005		System fails on documents larger than 100KB.	1	1	1	1	Extensive testing.	Test/S
31	2.006		System fails on documents larger than 1MB.	1	2	2	4	Extensive testing.	Test/S
32	2.007		System fails on documents larger than 32MB.	1	3	3	9	Balanced testing.	Test/S
33	2.008		Insufficient offline migration is too-frequent.	4	4	4	64	Report observed bugs.	N/A
34	2.009		Excessive offline migration results in "thrashing".	4	4	4	64	Report observed bugs.	N/A
35	2.010		Migration to offline fails.	1	2	4	8	Balanced testing.	Test/S
36	2.011		Migration to online fails.	1	2	4	8	Balanced testing.	Test/S
37	2.012		Librarian not notified when media load is needed.	4	3	4	48	Opportunity testing	N/A
38	2.013		User not allowed to wait for media load.	2	3	5	30	Opportunity testing	N/A
39	2.014		User wait for media load timing/escalation count wrong.	5	4	5	100	Report observed bugs.	N/A
40	Quality Risk Category Priority						1		

Рис. 10.2. Риски, связанные с нагрузкой, мощностью и объемом (из анализа рисков качества проекта “Суматра”)

ния могут привести к тому, что доступная оперативная память будет исчерпана, в результате чего произойдет миграция документов, находящих в оперативной и почти оперативной памяти, на другой уровень иерархии хранилищ. Другие действия приводят к возникновению иных состояний. Некоторые из этих состояний означают успешное сохранение или извлечение документа из системы, другие указывают на необходимость ожидания внешних действий библиотекаря документов, а третьи фиксируют ошибку. Лин Цу применяла метод тестирования на основе управляющей логики программы для проведения функционального тестирования, как это описано у Бориса Бейзера. Эмма ожидала, что этот метод также окажется полезным при нагрузочном тестировании[1].

Наконец, Эмма осознала, что ей вряд ли удастся задействовать 255 тестировщиков, тем более 24 часа в сутки и семь дней в неделю. Однако она могла воспользоваться инструментом нагрузочного тестирования, чтобы написать простые автоматизированные тестовые скрипты, которые могли бы зациклить прохождение системы через различные состояния и поддерживать ее непрерывную работу. Эти скрипты работали бы примерно так, как "тупые обезьяны", описанные Ноэлем Найманом (Noel Nyman). Такие скрипты могут только обнаружить, что система находится в непредвиденном состоянии, но не могут проверить, как система попала в это состояние. В то же время Эмма собиралась усилить автоматизированное тестирование за счет полного тестирования состояний и различных переходов между состояниями с помощью ручного функционального тестирования во время выполнения нагрузочных тестов[2].

№ этапа	Этап	Выполнено?
3.	Выбрать надлежащие методы тестирования для исследования тестовых условий и верификации ожидаемых результатов с учетом доступных времени и бюджета, тестопригодности системы, предназначенной для тестирования, тестовой среды и квалификации группы тестирования.	☑

Выбрав подходящие методы, Эмма начала готовить средства тестирования. Для сценария мыльной оперы она создала 32 767 учетных записей на web-сервере, используя несложную программу генерации учетной информации в формате ASCII и затем утилиту загрузки файлов в базу данных для помещения учетных записей в таблицу на сервере баз данных. Для создания файлов, предназначенных для редактирования, она воспользовалась TCL-скриптом. Для редактирования этих файлов она запланировала использовать те же самые скрипты. Скрипты создали файлы в соответствии с простым и предсказуемым образцом.

Ранее Лин Цу составила график состояний, на котором были изображены состояния системы и способы перехода от одного состояния к дру-

---

[1] См. описание этого и других полезных методов в книгах Бориса Бейзера (Boris Beizer) "*Black-Box Testing*" и Ли Копленда (Lee Copeland) "*Mastering Software Test Design*".

[2] См. статью Ноэля Наймана (Noe Nyman) "Using Monkey Test Tools", впервые опубликованную в журнале "*Software Testing and Quality Engineering*", Volume 2, Issue 1 (Jan/Feb 2000) и доступную в настоящее время на www.stickyminds.com.

гому. Эмма много раз обращалась к этому графику. Для создания "тупых обезьян"

Эмма с помощью инструмента автоматизированного тестирования разработала несложные скрипты. Они представляли собой основные повторяющиеся действия по созданию, загрузке, редактированию, сохранению и чтению документов в системе управления документооборотом. Эти скрипты могли сообщить об ошибочном состоянии, когда они оказывались в нем, но не могли проинформировать, как их действия повлияли на систему, предназначенную для тестирования. Поскольку файлы данных Эммы соответствовали определенному образцу, она могла разработать несложный скрипт, проверяющий корректность файлов, а, используя средства экспорта каталога системы управления документооборотом, она могла бы проверить, что утерянных (отсутствующих в перечне) и пропущенных (указанных в перечне, но отсутствующих) файлов в системе управления документооборотом нет.

№ этапа	Этап	Выполнено?
4.	Спроектировать, разработать, приобрести, усовершенствовать, сконфигурировать и документировать средства тестирования, тестовую среду и процесс выполнения тестов для воспроизведения определенных условий и проверки поведения системы с использованием выбранных методов тестирования. Усилить точность методов тестирования с помощью эмпирических методов с учетом данных заказчика или пользователей, отчетов о проблемах в эксплуатации (для ранних и конкурирующих версий, аналогичных продуктов и аналогичных наборов рисков качества системы), прогнозов отделов продаж и маркетинга о будущем использовании системы и других данных заказчика.	☑

При создании этих объектов Эмма поняла, что два нужных ей компонента отсутствуют в тестовой среде. Во-первых, у нее не было достаточного числа лицензий виртуальных пользователей для инструмента нагрузочного тестирования. Во-вторых, у нее имелись в наличии только две ограниченные системы управления иерархическими хранилищами. Эмма предполагала проводить тестирование для более крупных и развитых доступных систем.

Она решила обсудить это с Джамалом, которого она нашла в тестовой лаборатории. Джамал беседовал с младшим тестировщиком, который рассказывал об обнаруженной им ошибке.

"Классная ошибка, Эмамалини! Продолжай искать такие же нехорошие ошибки", – сказал с улыбкой Джамал. Он повернулся к Эмме и спросил: "Насколько я понимаю, я могу для тебя что-то сделать?"

"Изучая тестовую среду для тестирования производительности, я обнаружила, что ..., точнее, удивилась ситуации с управлением иерархическими хранилищами". Улыбка Джамала исчезла, как только заговорили на эту тему. Эмма торопливо продолжала: "Я помню все эти сентябрьские споры о серверах для иерархических хранилищ, но мне кажется, что нам не удастся загрузить систему несколькими интересными способами с помощью базовых моделей серверов, которые есть у нас".

"Расскажи мне подробнее о том, что ты не можешь протестировать".

Воодушевленная таким началом, Эмма пустилась в подробное описание некоторых тестов для проверки мощности и объема, которые требуют большей производительности и мощности процессоров, сети и жестких дисков, чем два сервера в лаборатории. Она описала два частных типа проблем производительности, о которых она читала и которые не удастся проверить.

"Хорошо, – сказал Джамал. – Что ты предлагаешь? Купить еще сервер? Или два?"

"Я думала об этом, – сказала она и вручила Джамалу несколько напечатанных страниц из Интернета.

Джамал быстро прочитал содержание листков и присвистнул: "Неслабые ценники".

"Да, – согласилась Эмма. – Но ошибки в этой части продукта могут поставить под вопрос жизнеспособность всего продукта".

Джамал думал минуту, потом медленно произнес: "Хорошо, я понял тебя. Ты знаешь, я тоже пытаюсь избежать рисков, поэтому не хочу рубить эту идею на корню, хотя мы уже рассматривали проблему ранее, – он сделал паузу. – Вот что мы сделаем. Давай устроим совещание и пригласим на него Кейт Эрнандес, Дженни и всех людей из группы разработки, кто отвечает за этот раздел системы. Ты расскажешь о своих соображениях по поводу технических рисков, мы выслушаем мнение системного архитектора, а затем заново оценим проблемы, связанные с бизнесом".

Совещание состоялось в этот же день после обеда. И Кейт, и Дженни поначалу были смущены тем, что сейчас в проекте вновь возник этот вопрос, как черт из табакерки. Однако Эмма и ведущий программист раздела управления хранилищами Боб Маршал быстро перешли к обсуждению серьезных технических проблем, и их руководители заблудились в деталях. Эмма и Боб пришли к общему выводу, что, хотя здесь есть некоторые риски, они ограничены ситуациями, когда используются файлы размером, близким к максимальному, и определенные конфигурации дисковых массивов RAID.

"Итак, – сказала Кейт. – Что вы думаете по поводу следующего предложения? Я попытаюсь найти одного или двух потенциальных заказчиков с подходящими файлами и конфигурациями, которые хотели бы провести для нас бета-тестирование Суматры, такое решение подойдет?"

Боб сказал: "Да, думаю, что вполне".

Эмма сначала колебалась, но потом возразила: "Что касается функциональности, то безусловно, но вряд ли в отношении производительности и нагрузки. Эти виды тестирования требуют специальной подготовки. При случайном бета-тестировании такие тесты не будут выполнены".

Это породило параллельную дискуссию между Бобом, Дженни и Кейтом.

Джамал, сидевший рядом с Эммой, быстро написал в своем блокноте: "Поедешь на место и проверишь там?" Она пожала плечами, покорно соглашаясь, и написала снизу: "Хорошо, но это будет менее эффективно и медленнее".

Джамал вмешался в разговор: "Видишь ли, Кейт, мы могли бы воплотить эту идею в жизнь. Как ты думаешь, смогла бы ты убедить наших по-

тенциальных заказчиков разрешить Эмме посетить их рабочие места и выполнить там тестирование производительности?"

Кейт посмотрела в окно, а потом сказала: " Да, я думаю, что мы сможем это сделать. Командировка обойдется дешевле, чем аппаратура". Она посмотрела Эмме в глаза и сказала по-дружески и с улыбкой, но оставаясь серьезной: "Не сомневаюсь, что Эмма не собирается пугать заказчиков рассказами об ошибках Суматры, не так ли?"

"Нет-нет, я не буду этого делать", — быстро ответила Эмма, отрицательно качая головой.

"Тогда договорились, — сказала Кейт. — У нас теперь два дела. Дженни и я будем искать одно или несколько подходящих мест для организации тестирования. Мы предоставим контактную информацию тебе, Джамал, до конца недели. Джамал, ты и Эмма должны определить, как выполнять ваши тесты на рабочих местах заказчиков. Поскольку мы будем использовать автоматизированные инструменты, я полагаю, что есть вопросы, связанные с материально-техническим обеспечением?"

"Да, наверное, — ответил Джамал. — Возможно, мы должны также запланировать еще одно совещание через пару недель, когда, по нашему мнению, эта часть системы тестов будет готова. Я бы предложил сначала завершить тестирование производительности внутри компании. Оно запланировано, — он с минуту пошуршал своими записями, — когда мы получим инкремент 3 в декабре, примерно 15-го числа. Таким образом, ориентировочная дата — начало января. Но думаю, что имеет смысл снова вернуться к этому вопросу 2-го декабря, чтобы посмотреть, где мы находимся".

№ этапа	Этап	Выполнено?
4.	Спроектировать, разработать, приобрести, усовершенствовать, сконфигурировать и документировать средства тестирования, тестовую среду и процесс выполнения тестов для воспроизведения определенных условий и проверки поведения системы с использованием выбранных методов тестирования. Усилить точность методов тестирования с помощью эмпирических методов с учетом данных заказчика или пользователей, отчетов о проблемах в эксплуатации (для ранних и конкурирующих версий, аналогичных продуктов и аналогичных наборов рисков качества системы), прогнозов отделов продаж и маркетинга о будущем использовании системы и других данных заказчика.	
5.	Если какие-либо элементы средств тестирования, тестовой среды или процесса выполнения тестов оказались недоступными в ходе выполнения этапа 4 из-за ограничений ресурсов, сроков и т.п., повторить этап 4 нужное число раз, чтобы приспособиться к этим ограничениям.	☑

Поскольку это предложение оказалось приемлемым для всех участников совещания, Джамал сделал для себя три пометки. Во-первых, нужно было запланировать очередное совещание. Во-вторых, обновить декомпозицию работ и график проекта, включив две недели работы за пределами офиса для Эммы и, следовательно, ее недоступность в офисе. Он подозревал, что придется сократить часть тестирования, которое могло бы

быть выполнено, пока Эмма будет в командировке. Тем не менее он надеялся, что к тому времени Динеш сможет продолжить автоматизированное тестирование, хотя и в сокращенном режиме. В-третьих, нужно было убедиться, что Эмма обучит Динеша выполнять автоматизированные тесты к моменту своего отъезда в январе. Эмма отправилась на свое рабочее место, чтобы закончить проектирование и разработку тестов для нагрузочного тестирования с учетом дополнительного требования сделать их переносимыми.

18 ноября, когда были написаны все тесты, Эмма приступила к проверке скриптов и данных. Сначала она организовала перекрестное рецензирование средств тестирования силами Лин Цу и Динеша. Это совещание, продлившееся пару часов, позволило выявить несколько дефектов, прежде всего в скриптах. Эмма провела вторую половину дня после совещания за исправлением ошибок в скриптах. Поскольку число дефектов, обнаруженных в скриптах, было невелико, она решила не планировать еще одно перекрестное рецензирование для сделанных исправлений. Это было одно из решений, принятие которых сотрудниками группы тестирования в рамках существующих процессов Джамал поощрял.

В настоящий момент в тестовой среде был установлен инкремент 2, в котором отсутствовал ряд ключевых свойств будущей версии. Написав несколько заглушек, Эмма увидела, что может выполнить скрипты и найти в них серьезные проблемы, если они там есть. Хитрость состояла в том, чтобы не помешать продолжающемуся тестированию. Она нашла окно в ночном графике выполнения тестов и настроила их запуск в это время. В результате были найдены еще ошибки в тестах, поэтому оставшуюся неделю Эмма провела за исправлением и перетестированием скриптов, пока не почувствовала, что они готовы для применения к инкременту 3. Наконец Эмма поместила все связанные файлы: тестовые данные, тестовые сценарии, инструменты и т.п. — в проектный репозиторий. На рис. 10.3 показан ход разработки тестовых сценариев для двух комплектов тестов по состоянию на среду, 20 ноября, когда Эмма завершала свою работу.

№ этапа	Этап	Выполнено?
6.	Проверить систему тестов, используя статические (например, рецензирование) и динамические (например, выполнение тестов) методы.	
7.	Поместить средства тестирования, описание процесса выполнения тестов, информацию о конфигурации тестовой среды, документацию о тестировании системы тестов (полученную на этапе 6) и всю прочую документацию, а также созданные файлы в библиотеку проекта или систему управления конфигурацией. Привязать версию системы тестов к версии системы, предназначенной для тестирования. Вести контроль изменений.	☑

В ходе разработки тестов Эмма также обнаружила ранее не рассмотренное требование — определение предельного числа файлов, которое может показать в перечне система управления документооборотом. Об этом была проинформирована группа управления изменениями (см. гла-

Заглушка (Stub)

Фрагмент программы (обычно небольшой), который вставляется в код либо заменяет компонент системы, предназначенный для тестирования, чтобы аппроксимировать работу ряда заменяемых с его помощью функций для выполнения определенных тестов.

ву 16), которая приняла коллективное решение, что разумным числом будет 1024, что соответствовало бы технической архитектуре и удовлетворило бы пользователей.

Эмма, узнав о принятом решении, взяла таблицу анализа видов ошибок и их влияния из репозитория документов и добавила два риска, связанных с этим ограничением. Первый относился к проблемам с обработкой каталогов нормального размера, например 100 файлов, второй — к проблемам на границе, 1024 файла.

№ этапа	Этап	Выполнено?
8.	Обновить перечень рисков качества на основе новых знаний, полученных при разработке последних наборов тестов. Оценить покрытие рисков качества новой системой тестов. Выявить остающиеся непокрытыми риски качества, для которых рекомендовано тестирование.	

	A	B	C	D	E	F	G	H	I	J	K
1	Sumatra Integration and System Test Case Tracking Spreadsheet										
2											
3											
4	Assigned	Test			Plan	Act	Plan	Actual	Exec	Exec	
5	To	ID	Test Suite/Case	Status	Ready	Ready	Effort	Effort	Effort	Duration	Comment
6											
7		1.000	*Functionality*								
8	EM	1.001	File	Ready	9/23	9/20	8.0	8.5	2.0	4.0	
9	EM	1.002	Edit	Ready	9/23	9/20	8.0	9.0	0.5	4.0	
10	EM	1.003	Font	Ready	9/23	9/20	8.0	8.5	0.5	4.0	
11	EM	1.004	Tables	Ready	9/23	9/24	8.0	8.5	3.0	4.0	
12	EM	1.005	Printing	Ready	9/23	9/22	8.0	7.5	2.0	4.0	
13	LTW	1.006	Managed Files	Ready	10/2	10/3	6.0	6.5	2.5	2.5	
14	LTW	1.007	Non-Managed Files	Ready	10/2	9/30	6.0	6.0	4.0	4.0	
15	LTW	1.008	DMS Check-In New	Ready	10/2	10/3	6.0	5.0	2.5	2.5	
16	LTW	1.009	DMS Check-In Existing	Ready	10/2	10/3	6.0	7.5	3.0	3.0	
17	LTW	1.010	DMS Check-Out	Ready	10/2	9/29	6.0	5.5	3.5	3.5	
18	LTW	1.011	DMS Load/Migrate Online	Ready	10/2	10/1	6.0	5.5	3.5	3.5	
19	LTW	1.012	DMS Load/Migrate NOL	Ready	10/2	10/3	6.0	5.5	3.0	3.0	
20	LTW	1.013	DMS Load/Migrate Offline	Ready	10/2	10/5	6.0	7.5	3.5	3.5	
21	LTW	1.014	DMS File Information	Ready	10/2	10/6	6.0	4.5	2.5	2.5	
22			**Suite Summary**		**10/2**	**10/6**	**94.0**	**95.5**	**36.0**	**48.0**	
23											
24		2.000	*Performance, Load, Capacity, and Volume*								
25	EM	2.001	Solaris Server	Ready	11/25	11/18	16.0	15.5	2.0	8.0	
26	EM	2.002	NT Server	Verify	11/25		16.0		1.5	8.0	
27	EM	2.003	Linux Server	Ready	11/25	11/17	16.0	9.5	1.5	8.0	
28	EM	2.004	AIX Server	Verify	11/25		16.0		1.0	8.0	
29	EM	2.005	Network	Ready	11/25	11/20	16.0	16.5	1.0	8.0	
30			**Suite Summary**		**11/25**	**11/20**	**80.0**	**41.5**	**7.0**	**40.0**	

Рис. 10.3. Выдержка из таблицы состояния дел по разработке тестов

Три решающих соображения

В следующей главе более тщательно рассматриваются каждый из этапов процесса проектирования и разработки тестов. В оставшейся части этой главы предлагаю поговорить о трех важных соображениях относительно проектирования тестов. Это метод определения корректности результатов тестирования, виды документации, необходимые для системы тестов, и способы учета комбинаторных взрывов, которые могут возникнуть при подсчете числа возможных тестовых условий. Старшие специалисты-тестировщики, проектировщики системы тестов и тест-менеджеры должны тщательно продумать все это до начала проектирования и разработки тестов. Логика в рассмотрении этих проблем имеет существенное значение для создания высококачественной системы тестов в долгосрочной перспективе, а также для проведения качественного и эффективного тестирования в каждом проекте тестирования.

Тестовые оракулы

Тестовый оракул — это система, метод или методика для предсказания или оценки корректности поведения системы, предназначенной для тестирования, в определенных условиях. Проблему определения подходящего оракула лучше всего показать на примере. Предположим, что вы хотите провести тестирование функции, возвращающей значение синуса аргумента, выраженного в радианах. Вы могли бы обратиться к курсу тригонометрии для средней школы и вспомнить некоторые из известных значений синуса:

$$\sin(0) = 0$$

$$\sin(\frac{\pi}{6}) = \frac{1}{2}$$

$$\sin(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$$

$$\sin(\frac{\pi}{2}) = 1$$

Теперь вы можете воспользоваться тем фактом, что для углов от $\pi/2$ до π радиан график функции синуса является зеркальным отражением относительно вертикальной прямой $x = \pi/2$, а для углов от π до 2π радиан график функции синуса является зеркальным отражением относительно оси абсцисс. В результате мы получаем 13 известных значений кривой, что позволяет нам обнаружить грубые ошибки. Чтобы получить более точные значения, необходимо аппроксимировать функцию синуса. (Например, разложение в ряд Тейлора представляет собой более точный и сложный способ аппроксимации функции путем увеличения числа чле-

нов ряда.) Однако чем больше членов мы добавляем, тем более сложным и, следовательно, подверженным ошибкам становится тестовый оракул.

Тестовый оракул, способный предсказать корректное поведение системы, предназначенной для тестирования, возможно, так же сложно разработать, как и саму систему. (При создании некоторых авиационных систем разрабатывается несколько версий программного продукта, а результаты сравниваются, чтобы определить, какой из многочисленных откликов систем является правильным.)

Для упрощения задачи мы можем ограничить оракулы за счет игнорирования определенных входных и выходных данных и вариантов поведения системы. В приведенном примере я сначала игнорировал все ошибки, за исключением грубых, а затем применил в качестве решения проблемы приближение рядами Тейлора высокого порядка, не учитывая ошибки низшего порядка. Но как быть в случае, если точность значений функции синуса является критической для верного нацеливания межконтинентальной баллистической ракеты или системы противоракетной обороны, которая должна вычислять курс перехвата для разрушения цели, летящей на сверхзвуковых скоростях?

Если у вас имеется конкурирующая, действующая или еще какая-либо система, которая может имитировать какие-то или все варианты поведения системы, предназначенной для тестирования, то у вас есть оракул. Например, однажды я работал в проекте тестирования системы для обработки потребительских кредитов. В том проекте старая клиент-серверная система стала оракулом для новой системы на основе браузера и интранета. Каждое заявление на кредит отправлялось в обе системы, а предлагаемые продукты, полученные в качестве результата обработки системами, сравнивались по возможности положительного решения, проценту, максимальному размеру ссуды, ежемесячной выплате и срокам кредита. Проблемами при таком подходе могут стать точность и кредитоспособность. Если ежемесячную выплату уменьшить на 1 цент, заметит ли это клиент банка? И кто говорит, что действующая система работает верно? Насколько вероятно, что различие между выводом, полученным в системе, предназначенной для тестирования, и ожидаемым результатом указывает на ошибку в оракуле?

В некоторых случаях можно пренебречь точным прогнозом и довериться тестировщикам. Так, в примере с продуктом SpeedyWriter мы можем и, вероятно, должны оставить определенные аспекты, касающиеся разумных ожиданий пользователя, на усмотрение тестировщика. Если предполагается выдача определенного сообщения об ошибке, когда пользователь пытается перезаписать файл, предназначенный только для чтения, тестировщик, вероятно, может сделать вывод о том, что всплываю-

Тестовый оракул (Test Oracle)

Система, метод или методика для предсказания или оценки корректности поведения системы, предназначенной для тестирования, в определенных условиях.

щее окно с сообщением об ошибке должно появиться на экране и быть активным. Более того, можно доверить тестировщику принятие решения о том, является ли сообщение об ошибке верным.

Хотя этот вид разумного оракула обычно применяется к продуктам, предназначенным для конечного пользователя, уровень знания предметной области тестировщиком для такого рода предсказаний варьируется. Большинство работающих методов исследовательского тестирования опирается в определенной степени на подходы, в которых в качестве оракула выступает тестировщик, хотя мой опыт исследовательского тестирования также включает в себя применение в качестве оракулов эталонных систем.

Даже точный оракул не сможет рассказать вам все, что вы хотите знать. В примере с синусом вычисления ряда Тейлора могут предоставить мне значение функции синуса, но они ничего не скажут о том, сколько для этого понадобится времени. В случае с системой противоракетной обороны или прогноза погоды скорость вычислений так же важна, как и точность, поскольку результат, полученный после того как ракета достигнет цели или над территорией пройдет торнадо, никому не нужен!

Понятие оракула пришло из греческой истории и мифологии. Оракулы предсказывали будущее, как правило, используя кости и внутренности домашних животных. Насколько мне известно, в процессе проектирования, разработки и выполнения тестовых сценариев пока не пострадало ни одно животное, но это лишь потому, что у нас пока нет того количества проблем, чтобы опробовать подобные решения в области проектирования тестов. В тех случаях, когда необходимы точные оракулы, могут потребоваться значительные усилия для их создания[1].

Подходящая документация

Некоторые тестировщики заполняют целые полки папками с документацией по тестированию. Другие не пишут практически никаких документов. Оба типа тестировщиков достигали успехов, и оба терпели серьезные неудачи. Между этими двумя экстремальными подходами находится целый спектр вариантов подготовки документации, приводящих как к успеху, так и к разочарованию пользователей. Каковы же соображения, которые позволяют определить объем документации, которую мы должны подготовить?

Когда мы документируем систему тестов, то обычно пытаемся точно описывать предстоящее тестирование. Это относится к тестовым сценариям, начиная с исследовательских и кончая сценариями, написанными с помощью скриптов. Тестовые сценарии, зафиксированные в скриптах, точно определяют действия, данные и ожидаемые результаты с незначительной неопределенностью или без неопределенности. Точно документированная система тестов четко фиксирует конфигурацию тестовой среды также с минимальной неопределенностью.

---

[1] Более подробно о тестовых оракулах см. в статье Дуга Хоффмана (Doug Hoffman) "*Using Oracles in Test Automation*" на сайте www.softwarequalitymethods.com.

Эталонная система (Reference System)

Особый тип *тестового оракула*, когда существующая система моделирует некоторые или все ожидаемые варианты поведения системы, предназначенной для тестирования.

Неопределенность, на первый взгляд, может показаться нежелательным свойством средств тестирования. Это, безусловно, нежелательное свойство при определении требований или рисков качества. Однозначные средства тестирования поддерживают воспроизводимые тесты, что важно для регрессионного тестирования. Однозначные средства тестирования также позволяют сделать интерпретацию результатов тестирования менее зависимыми от человека, выполняющего тесты. Однозначные средства тестирования легко контролировать в том смысле, что другие могут изучить средства тестирования и точно понять, что было подвергнуто тестированию.

Однако точные тестовые сценарии имеют и обратную сторону. Описание точного тестового сценария обычно очень многословное и сложное. Сложность является не линейной, а экспоненциальной функцией размера сценария, как известно нашим коллегам-программистам. Например, написание этой книги, которая примерно в 100 раз больше, чем мои типичные отчеты о ходе тестирования, потребовало более чем в 1000 раз больше времени. Иногда у меня нет времени или ресурсов, чтобы документировать систему тестов с предпочтительной для меня точностью.

Чем точнее документированы средства тестирования, тем точнее должно быть наше предвидение. Если в ходе разработки тестов я попытаюсь точно описать, как будет работать пользовательский интерфейс, возможно, я увижу в процессе выполнения тестов, что я был неправ. Это приведет к переработке средств тестирования, что вряд ли является эффективным расходованием времени. Наконец, хотя точность способствует повторному использованию тестов в регрессионном тестировании, повторное использование предполагает также возможность сопровождения тестов. Чем более точно подготовлены средства тестирования, тем больше усилий требуется для их изменения, когда в систему вносятся изменения.

Некоторая неоднозначность неизбежна. Представьте тест для ручного тестирования SpeedyWriter, зафиксированный в скрипте. Можно ли следующий тестовый сценарий назвать однозначным?

1. Запустить SpeedyWriter. Приложение должно открыть браузер.
2. Создать новый документ. Должен появиться чистый документ.
3. Набрать четыре строки текста. Проверить, что никакая часть текста не утеряна.
4. Сохранить файл. Проверить, что появляется окно "*Сохранить как*".

Заметим, что остается много вариантов неоднозначности. Какой браузер должен запустить SpeedyWriter? Как мы узнаем, что приложение от-

крылось правильно? Как мы должны создавать новый документ: вызвав с помощью мыши меню или используя "горячие клавиши"? Что означает "чистый документ": это чистый экран или сверху экрана должна быть панель инструментов, сбоку – полоса прокрутки и т.п.? Можно ли использовать различные шрифты для набора текста, или это означает добавление дополнительных тестовых условий? Как должно выглядеть окно "Сохранить как"?

Я могу получить абсолютно точные и однозначные тесты для ручного или автоматизированного тестирования в виде скриптов, только если я могу предсказать заранее точный образ каждого экрана в сети, содержащего систему, предназначенную для тестирования, каждый бит памяти (оперативной памяти, кэш-памяти процессора, дискового пространства, кэш-памяти диска, видеопамяти и т.д.) в каждой системе сети, содержащей систему, предназначенную для тестирования, и каждый сетевой пакет, проходящий по сети, содержащей систему, предназначенную для тестирования. И я должен уметь предсказать это в любой момент проведения тестирования. И даже в этом случае, я, вероятно, что-то упущу[1].

С одной стороны, следующие воздействия, соображения и факторы обычно способствуют увеличению объемов документации:

- Потребность в регрессионном тестировании и воспроизводимости тестов. Если мне нужно выполнять одни и те же тесты снова и снова, особенно если я должен многократно выполнять в точности те же тесты, мне потребуется точно документировать, что я делаю, как делаю, какие используются данные и что я ожидаю получить.
- Потребность в проверяемости. Если я думаю, что мне потребуется отстаивать результаты проведенного тестирования (например, в суде), то важно точно фиксировать все, что я делаю. Например, при выполнении работ по контракту (оборонные заказы или заказное программирование) или для законодательно регулируемых отраслей (программные средства для здравоохранения и для управления ядерными объектами в США) тестирование должно соответствовать определенным стандартам.
- Использование автоматизации тестирования в настоящее время или в планах на будущее. Если я намерен автоматизировать часть функциональных или других тестов, которые ранее выполнялись вручную, наличие стабильного понятного процесса выполнения тестов, в котором используются четко определенные тестовые сценарии, способствует успешной автоматизации тестирования.
- Потребность в формализации и стандартизации процесса тестирования и документации. Формализованные процессы обычно используют шаблоны и стандарты документации (см. главу 17), это означает, что минимальный уровень документации определяется стандартом.

---

[1] Несколько примеров тестовых сценариев разных степеней точности и документированных разными способами можно найти в главе 3 моей книги "Managing the Testing Process, Second Edition".

- Уровень риска системы. Системы с повышенными требованиями к безопасности и с особым целевым назначением должны работать очень точно при каждом их использовании. Педантичный подход к тестированию с тщательным документированием и применением перекрестных ссылок, как правило, является обязательным требованием для достижения необходимого уровня доверия к системе[1].
- Жесткие временные рамки для выполнения тестов/Однажды я должен был выполнить серию тестов для проверки потенциальных проблем обработки данных в семействе серверов (server farm). Тесты предполагали очень точную установку серверных часов и загрузку определенных множеств данных, а затем оценивали, что происходит в случае интересных событий, относящихся ко времени, например перехода от 31-го декабря 1999 г. к 1-му января 2000 года. Если хотя бы один из сценариев не выполнялся или выполнялся неверно, когда наступали такие состояния, мы вынуждены были воспроизводить, как минимум, соответствующий интервал времени. В некоторых случаях приходилось начинать с самого начала последовательности. Точная документация способствовала более качественному выполнению этих хрупких тестов.
- Сложность процесса выполнения тестов, тестовых сценариев и системы, предназначенной для тестирования. Мозг человека способен обрабатывать одновременно только конечное число элементов информации. Сложный тест, особенно тот, в котором порядок, время и данные должны быть четко организованы, требует исчерпывающего документирования, чтобы убедиться, что нужные события происходят в нужное время и мы ищем ошибки в нужных местах. Например, тестовый сценарий для тестирования Интернет-приложения может включать в себя следующий шаг, связанный с ожиданием результата: "Подтвердите, что присоединенный к сообщению файл воспроизводится корректно". Но будет ли шаг "Подтвердите, что приборы взлетно-посадочной полосы дают правильные показания" достаточным, если мы проверяем летные качества лайнера "Конкорд"?
- Необходимость фиксации знаний. Если тесты определены неточно потому, что всем известно, как выполнять эти тесты, это не вызовет проблем. Но тестовые сценарии и процедуры, которые находятся в голове у одного человека, порождают организационный риск и риск управления для группы тестирования.
- Необходимость руководить неопытными тестировщиками и делиться знаниями. При правильном руководстве определенные люди, которые были бы в противном случае недостаточно квалифицированными для участия в тестировании системы, могут внести свою лепту в процесс тестирования. Часть руководящих указаний может быть сделана в форме документированных тестовых сценариев. Документированные тестовые сценарии дают возмож-

---

[1] Обратите внимание на главу 2 в книге "*Computer-Related Risks*" Петера Ноймана (Peter Neuman).

ность гораздо шире забросить сети в поисках тестировщиков и более успешно справиться с задачей поиска персонала для проекта. Документированные тестовые сценарии помогают людям вносить свой вклад в тестирование, в котором они вряд ли принимали бы участие в противном случае. Документированные тестовые сценарии позволяют разделить коллективное знание группы тестирования с другими участниками проекта.

- Степень, в которой система тестов, сама является продуктом. Если система тестов является документом, передаваемым в какую-либо другую организацию, либо если по какой-либо причине она будет или может быть передана после завершения тестирования, документирование позволяет упростить передачу знаний о том, как выполнялось тестирование, а также об инструментах, данных, скриптах и другой работе. Хорошо подготовленная документация улучшает впечатление от выполненного тестирования и профессиональный имидж группы тестирования, которая это сделала.

С другой стороны, следующие воздействия, соображения и факторы обычно способствуют уменьшению объемов документации:

- Время и ресурсы, доступные для проектирования и разработки тестовых сценариев. Малобюджетный проект с короткими сроками выполнения может заставить меня разработать меньше документации, чем я предпочитаю. Время, которое я трачу на документирование одного тестового сценария, — это время, которое я не потрачу на написание другого тестового сценария, поэтому, если я должен выбирать между тем, чтобы не тестировать некоторые области повышенного риска или оставить некоторые детали тестирования на усмотрение тестировщика, я выберу последнее.
- Квалификация тестировщиков, в первую очередь тех, кто выполняет тестовые сценарии. Менее квалифицированные тестировщики могут нуждаться в более детальном руководстве.
- Вопросы мотивации. Со временем тестировщики могут рассматривать выполнение одного и того же множества точно документированных тестовых сценариев как утомительное и немотивированное занятие. Если другие факторы, в особенности, связанные с требованиями повышенной безопасности, требуют точного документирования, то обмен выполняемыми задачами и выделение времени на исследовательское тестирование могут способствовать решению проблемы.

Вообще говоря, следующие воздействия, соображения и факторы связаны с надлежащим уровнем документирования различными способами и способствуют как уменьшению, так и увеличению объемов документации:

- Вопросы эффективности, повторного использования и сопровождения. Иногда я считаю, что некоторые части процесса тестирования должны быть определены точно для того, чтобы сделать тестирование эффективным. Например, как гарантировать, что два тестировщика не начнут по случайности одновременно выполнять один и тот же тест. В то же время я должен быть менее точным в де-

талях, относящихся к пользовательскому интерфейсу, если я считаю, что он станет объектом частых изменений. Если требуется точность, я могу сослаться на другую проектную документацию, а не повторять уже однажды зафиксированную информацию в тексте документов по тестированию. Аналогично, если доступны стандартные списки контрольных вопросов для определенных видов тестирования, я могу сделать ссылки на эти списки, а не воспроизводить их в системе тестов[1].

- Ожидания и требования лиц, заинтересованных в тестировании и в проекте в целом. Иногда менеджеры проектов ожидают увидеть определенные типы или форматы документации. Иногда система тестов является передаваемым документом. Один раз я работал с клиентом, который пользовался услугами компании, поставлявшей ему большую и сложную подсистему для интеграции с системой, предназначенной для тестирования. Я сказал моему клиенту, что разумно сделать передачу тестовых сценариев, результатов и инструментов тестирования поставщика частью контракта, чтобы мы могли далее успешно провести тестирование этой подсистемы в контексте нашей системы.
- Разумные представления и не очень. Некоторые тестировщики, программисты и менеджеры думают: "Ага, у нас нет времени на всю эту бумажную работу, мы должны просто направить все силы на тестирование". Другие полагают: "Знаете, документация и стандарты представляют квинтэссенцию тщательных размышлений и прошлого опыта, и мы должны следовать наилучшим практикам, чтобы минимизировать проектные риски". В той или иной степени обе позиции могут быть верными. Я также считаю, что у меня есть некоторые пристрастия, появившиеся в результате предыдущего опыта, которые склоняют меня ко второму направлению мыслей. Вероятно, у вас есть собственные пристрастия, которые помогают вам, и вполне логично следовать подходам, которые работали раньше. Однако неправильно позволять вашим представлениям о документации (или о других практических вопросах) становиться столь жесткими, что вы начинаете следовать ритуалам, а не принимать рациональные решения.
- Отсутствие знаний и хаос. Наконец, рассмотрим два нехороших, но не являющихся необычными фактора, оказывающих влияние на подходы к документированию. Некоторые просто не знают, как документировать системы тестов, или не думают, что существуют различные способы делать это. В некоторых проектах изменения являются столь пронизывающими и непрерывными, что любая попытка зафиксировать что-нибудь приводит к появлению документа, который устаревает раньше, чем завершается его разработка.

---

[1] См., к примеру, достаточно подробный перечень списков контрольных вопросов для тестирования web-приложений в отличном ресурсе Стива Сплайна (Steve Splaine) и Штефана Яскиля (Stefan Jaskiel) "*The Web Testing Handbook*".

Столь же длинным, как только что приведенный, может быть список других воздействий, соображений и факторов, оказывающих влияние на вопросы документирования. Выбор различных уровней и подходов к документированию тестирования является серьезным решением, которое требует тщательного рассмотрения всех сил, воздействующих на этот процесс[1].

Комбинаторные взрывы

Предположим, что у вас есть экранно-ориентированная система, например, программа, принимающая заявления на получение кредитов. По четырем экранам разбросано около 100 полей. Некоторые поля — двоичные, например "Объявляли ли вы о банкротстве в последние семь лет?" Некоторые поля допускают целочисленные значения, например "Сколько лет вы живете по этому адресу?" Есть несколько полей, принимающих произвольный текст, например "Введите фамилию вашего работодателя". Получается более 10^{30}, вероятно, даже больше 10^{60}, потенциально интересных комбинаций данных, которые могут быть введены в это множество экранов, хотя большое число бессмысленных и невозможных комбинаций можно удалить. Например, если заявитель отвечает "Нет" на вопрос, имеет ли он неуплаченные ссуды на автомобиль, последующие поля по поводу неуплаченной ссуды на автомобиль: ежемесячные выплаты, остаток долга, срок и наименование кредитора — будут наверняка пустыми. Даже если природа приложения такова, что большинство этих бессмысленных комбинаций можно устранить, скажем, 99.999% потенциальных комбинаций данных, все равно останется более 10^{26}. Даже если вы сможете найти способ обработать все эти данные в течение некоторого реального периода времени, вы наверняка не сможете найти время проверить все ожидаемые результаты.

Другой пример: комбинаторный взрыв возникает, когда вы тестируете автономное приложение для ПК, которое работает в различных версиях основной операционной системы и на различных платформах. Каждый вариант операционной системы, включая служебные пакеты и патчи, комбинируется со всеми поддерживающими приложение платформами и всеми возможными сосуществующими программами, что в результате дает огромную матрицу возможных конфигураций тестовой среды. Также может существенно различаться порядок установки служебных и сосуществующих программ. Ошибки в сосуществующих программах на персональных компьютерах на платформе Windows обычно называют адом динамически *присоединяемых библиотек* (DLL hell). Библиотеки общего доступа перекрываются новой версией при установке дополнительной программы, эта новая программа оказывается несовместимой с ранее установленными программами, которые ее должны использовать, что в результате делает ранее установленные программы неработоспособными.

---

[1] По электронной почте я обсуждал эти вопросы с Грегори Дейчем (Gregory Daich), Россом Коллардом (Ross Collard), Майклом Болтоном (Michael Bolton) и Робином Голдсмитом (Robin Goldsmith), они помогли сделать более четкими мои представления об этой области. Благодарю их за потраченное время и признаю их вклад в этот раздел книги.

Если у вас имеется 100 программ и вы хотите провести исчерпывающее тестирование влияния порядка установки этих программ, то вы получите примерно 9x10**157 условий для тестирования.

Эти два примера иллюстрируют общую проблему при проектировании тестов — комбинаторный взрыв. Были созданы различные методы сокращения этих больших и неуправляемых наборов тестов до разумного размера. Подробное обсуждение этих методов выходит за рамки книги, поэтому я кратко расскажу о методах, а затем укажу источники для дальнейшего изучения, если вы посчитаете какой-либо из этих методов полезным.

Что касается комбинаций конфигураций, метод, которым я обычно пользуюсь, включает в себя три этапа. Во-первых, я выявляю важные конфигурации. Конфигурация считается важной, если есть вероятность ошибок в ней (технический риск) и конфигурация является распространенной (бизнес-риск). Во-вторых, я распределяю различные тестовые сценарии, которые предполагаю выполнять, по различным конфигурациям. В-третьих, я планирую изменять конфигурацию каждый раз при повторном выполнении теста в последующих тестовых циклах[1].

Другое семейство методов, применяемых к комбинаторным взрывам, называют по-разному: планирование экспериментов, методы Тагучи, дробные двухфакторные (реже — трехфакторные) планы и ортогональные массивы. Эти методы обычно рассматривают взаимодействия между парами или тройками различных факторов и выбирают небольшое подмножество из потенциально огромного множества, которое покрывает интересные взаимодействующие пары. Название "ортогональные массивы" происходит от того, что метод предполагает построение массива, состоящего из столбцов, каждый столбец представляет один из факторов или переменных, т.е. полей на экране. Взаимодействующие пары этих факторов или переменных, исключая неверные, недостижимые или невозможные комбинации, заполняют столбцы, пока каждая взаимодействующая пара не появится хотя бы в одной строке массива.

О методе Тагучи написано множество книг, хотя они не посвящены специально программным продуктам. Мне известны две статьи на эту тему, касающиеся исключительно программ. Одна из них написана Генити Тагучи (Genichi Taguchi) и Раджешом Джагалумом (Rajesh Jugulum) "Principles of Taguchi Methods for Software Testing". Другая написана Элфрид Дастин (Elfriede Dustin) "Orthogonally Speaking". Возможно, три из наиболее понятных описаний этого метода в применении к программному обеспечению можно найти в книге Рика Крейга (Rick Craig) и Штефана Яскиля (Stefan Jaskiel) "*Systematic Software Testing*", на сайте Джеймса Баха (James Bach) и в книге Ли Копленда (Lee Copeland) "*Mastering Software Test Design*"[2].

---

[1] Этот метод описан более подробно в главах 3 и 5 моей книги "Managing the Testing Process, Second Edition".

[2] Статья Тагучи и Джагалума появилась в материалах Второго всемирного конгресса по вопросам качества программного обеспечения (*Proceedings of the Second World Congress for Software Quality*), а статья Дастин была опубликована в журнале "*Software Testing and Quality Engineering*", Volume 3, Issue 5 (Sep/Oct) 2001). Крейг и Яскиль рассматривают вопрос в главе 6 своей книги. Инструменты для создания тестов по методу ортогональных массивов можно найти на www.satisfice.com, www.phadkeassociates.com и www.lumsoft.com.

К комбинаторным взрывам относится также метод под названием "анализ эксплуатационной безопасности" (hazard analysis) или "причинно-следственный анализ" (cause-and-effect analysis). Сначала выделяются определенные типы проблем (нежелательных эффектов), например перезапись разделяемой динамической библиотеки несовместимой библиотекой, и делается попытка вычленить все возможные пути (коренные причины), которые могут к этому привести. В результате тестирование может быть направлено на проверку только тех комбинаций из огромного числа возможных, которые могут вызвать отказы и сбои в системе. Можно найти описание конкретного случая применения этой идеи к тестированию программ в статье Ясухаре Ниши (Yasuharu Nishi) "Resource Path Testing: A Framework for Design of System Testing"[1].

Аналогично можно применять различные типы разделения на классы или области эквивалентности. С помощью этих методов огромное множество комбинаций разделяется на небольшое множество областей. Мы считаем, что внутри каждой области система будет работать со всеми комбинациями одинаково. Внутри областей есть несколько интересных точек; тестирование в этих точках показывает, является ли работа системы со всеми элементами области, которую представляют эти точки, корректной. Способ, с помощью которого система, предназначенная для тестирования, определяет или реализует эти области, может сделать этот анализ и результирующее множество тестов проще или сложнее. Поэтому применение правильного подхода непосредственно в период проектирования архитектуры программы, вероятно, позволит упростить тестирование. Борис Бейзер подробно разбирает эту тему в своих книгах "*Software Testing Techniques*" и "*Black-Box Testing*". То же вы можете найти у Ли Копленда в "*Mastering Software Test Design*".

Наконец, еще один метод применяет для отбора конкретных наборов данных сценарии использования системы (use cases), парадигматическое представление последовательности выполняемых действий в комбинации с анализом граничных значений. Этот подход я использовал с переменным успехом для тестирования экранно-ориентированной системы с множеством полей для обработки заявлений на получение кредитов при линейном характере последовательности выполняемых действий. Большая часть работы системы – есть обработка полей экранов и самих экранов в строго заданной последовательности, что позволяет нам исключить огромное множество возможных вариантов порядка ввода данных, которые маловероятны и даже невозможны при реальном использовании системы. Росс Коллард (Ross Collard) дал отличное описание этого метода в статье "Test Design: Developing Test Cases from Use Cases"[2].

---

[1] Статья Ниши была впервые опубликована в материалах Второго всемирного конгресса по вопросам качества программного обеспечения *Proceedings of the Second World Congress for Software Quality*), а затем перепечатана в журнале "Software Quality Professional", Volume 3, Issue 3 (June 2001). Диаграммы причинного анализа рассматриваются в книге Каору Исикавы (Kaoru Ishikawa) "*Guide to Quality Control*".

[2] Статья Колларда была впервые опубликована в журнале "*Software Testing and Quality Engineering*", Volume 1, Issue 4 (Jul/Aug} 1999) и может быть в настоящее время найдена на www.stickyminds.com.

От технического взгляда к точке зрения менеджмента

В этой главе проектирование и разработка тестов рассмотрены с технической точки зрения. Мы изучили один набор тестов из основного примера книги и увидели, как Эмма применяет наилучшие способы проектирования и разработки тестов, чтобы выстроить свою часть системы тестов. Мы обсудили некоторые важные практические соображения, касающиеся проектирования и разработки тестов. Однако нам еще нужно исследовать процесс целиком, более общую картину и точку зрения менеджмента. Если время и деньги были бы неограниченными, мы могли бы написать идеальную систему тестов, которая рассказала бы нам все о качестве системы, предназначенной для тестирования. Но у нас есть реальные ограничения. В следующей главе мы рассмотрим эти ограничения, а затем исследуем процесс проектирования и разработки тестов глобально.

ГЛАВА 11

Заполнение ванны: тестовое покрытие и качество системы тестов

В этой главе мы завершим исследование процесса проектирования и разработки системы тестов. Прежде всего мы рассмотрим ключевой момент: как понять, когда разработано достаточно тестов. Различные способы, с помощью которых мы можем измерять полноту тестирования, обычно называются *тестовым покрытием*. За этой простой фразой скрывается бесконечное количество методов. Все эти взаимодополняющие методы позволяют специалистам-тестировщикам оценить пути, посредством которых тесты нацеливаются и не нацеливаются на определенные разделы, интересующие тестировщиков.

Полнота тестирования — это, безусловно, один из важных аспектов хорошей системы тестов, но есть и другие. Я завершаю эту главу рассмотрением всего процесса проектирования и разработки системы тестов. Как мы узнаем, что процесс качественный? Как мы можем перейти от текущего процесса, который, возможно, не работает, к более качественному процессу?

Краткий обзор методов анализа тестового покрытия

Перед тем как мы вернемся к нашему другу Джамалу и его частным проблемам с тестовым покрытием, позвольте кратко рассмотреть некоторые общие методы тестового покрытия. На нескольких страницах невозможно разместить исчерпывающее введение в сложную и многообразную тему тестового покрытия. Тем не менее я рассмотрю идеи, которые используются в основном примере книги.

Я подготовил ссылки на случай, если читатель захочет исследовать тему более подробно. Если вы и ваша группа в основном занимаетесь поведенческим тестированием, не слишком тревожьтесь, если вы не поня-

ли всех деталей материала, посвященного структурному покрытию. Я не случайно рассматриваю эти методы тестового покрытия, поскольку они наиболее широко используются на практике.

Когда говорят о получении хорошего тестового покрытия, интуитивно имеют в виду, что система тестов нацелена на большинство тестовых условий для тестируемой системы или, по крайней мере, на большую часть важных тестовых условий. Если это определение выглядит нечетким, то лишь потому, что так оно и есть на самом деле. Нечеткое, неточное применение фразы "тестовое покрытие" является обычным явлением. Попробуем снять часть этой неопределенности, чтобы можно было понять, о чем речь, поскольку я использую это словосочетание для обозначения четкого и измеряемого понятия.

Для начала скажем, что анализ тестового покрытия — это отображение тестов на некоторое свойство системы, предназначенной для тестирования. Свойства могут быть поведенческими (т.е. описывать то, что система должна делать), структурными (т.е. описывать то, как работает система) или некоторой комбинацией этих двух типов. Механизм, с помощью которого происходит отображение, позволяет провести количественную оценку тестового покрытия.

Для покрытия поведения (или черного ящика) представим, что для каждого выявленного риска качества мы определяем по крайней мере один тест. Считая, что мы хорошо провели анализ рисков качества, а затем также хорошо справились с проектированием и разработкой тестов, мы покрыли, таким образом, риски качества.

Мы можем измерить это покрытие с помощью матрицы, где по одной оси представлены тестовые сценарии, а по другой риски качества (см. ниже). Для каждого пересечения тестового сценария и риска качества, т.е. каждого элемента матрицы, мы можем измерить покрытие риска качества с помощью данного тестового сценария[1].

В дополнение к трассировке рисков качества мы можем отслеживать покрытие требований, поддерживаемых конфигураций и сценариев использования системы. Если мы извлекаем риски из требований, конфигураций и сценариев использования, то покрытие рисков качества будет объединением множеств покрытий этих трех областей.

Документы, аналогичные анализу рисков качества и спецификации требований, являются статическими моделями функционирования системы. Мы также можем построить другие модели поведения системы. Эти модели могут фиксировать потоки и виды данных системы, последовательность экранов, состояния, в которых может находиться система, и последовательность их обработки. Эти модели графические, с узлами и связями между ними, построенные графы можно трансформировать в таблицы. Можно провести оценку покрытия относительно этих моделей, проверив, что каждому узлу и каждой связи в графе соответствует тестовый сценарий[2].

---

[1] Рик Крейг (Rick Craig) и Штефан Яскиль (Stefan Jaskiel) в книге "*Systematic Software Testing*" дают неплохое описание этого метода, используя понятие реестра целей тестов (test objective inventories).

[2] Этот метод подробно излагается в книге Бориса Бейзера "*Тестирование черного ящика*".

Некоторые измеряют покрытие бинарно: есть или нет, 0 или 1, ставят контрольную метку или нет. Однако более точная оценка покрытия требует от нас понимания, в какой степени тесты нацелены на интересующие нас области системы. Точность измерений степени покрытия ограничена, но выделение сильных и слабых связей между тестами и интересующей нас областью (в этой книге — рисками качества) является неплохим началом.

Оценки покрытия поведения системы, например покрытия требований, наиболее тесно связаны (явно или неявно) с бизнес-рисками. В какой степени и какие отказы могут подвергнуть риску ценность системы? Мы должны проверить каждую из областей пропорционально влиянию сбоя на возможность использования системы.

Еще один вопрос, который мы могли бы задать, относится к техническому риску. Какова вероятность того, что произойдут определенные сбои? Мы также должны проверить каждую из областей пропорционально каждому риску.

Чтобы оценить покрытие технических рисков, можно обратиться к элементам архитектуры. Можно построить модель системы, а затем проверить покрытие этой модели. Простое выявление основных подсистем программного и аппаратного обеспечения в большой системе, а также данных и управляющей логики программы позволяет получить модель для оценки покрытия тестами технических рисков. (Тестовое покрытие, проектирование и разработка тестов, основанные на этих видах информации, иногда называются *тестированием серого ящика.*) Переход к следующему этапу и анализ наиболее вероятных сбоев каждой подсистемы и каждого интерфейса между двумя и более подсистемами, а затем сопоставление с этими элементами определенных тестовых сценариев обеспечивают более полное покрытие (см. раздел "Выполнение тестов" главы 3).

Продолжая движение к более низким уровням системы, отдельным строкам исходного кода, мы можем посмотреть на технические риски с точки зрения сложности. Поскольку человеческий мозг способен оперировать ограниченным числом объектов одновременно, то чем сложнее фрагмент кода, тем более вероятно, что программист допустил в нем ошибку. В 70-х гг. прошлого века Томас Маккейб (Thomas McCabe) придумал способ измерения сложности, он ввел понятие "*цикломатическая сложность*". Для цикломатической сложности каждая точка принятия решения, место, где программа может выбрать один из двух и более путей на основе определенного условия, увеличивает сложность кода. Чем сложнее фрагмент кода, например функция или метод, тем более тщательному тестированию он должен подвергаться (если не происходит восстановления структурной схемы и алгоритма работы по исходным текстам)[1].

---

[1] Дополнительно об измерении цикломатической сложности Маккейба можно прочитать во введении в цикломатическую сложность в "Software Engineering Institute's Software Technology Review", http://www.sei.cmu.edu/str/descriptions/cyclomatic_body.html. Можно также посмотреть исчерпывающее введение в измерение с подробной библиографией в "*Encyclopedia of Software Engineering, Second Edition*".

Маккейб ссылается на множество тестов, которое увеличивается в размерах с ростом цикломатической сложности, как на базовые тесты (basis tests). Это такой вид тестового покрытия, где по крайней мере один тест проверяет передачу управления в каждой точке принятия решения и в каждом операторе между каждой точкой решения. Число базовых тестов для функции обычно такое же, как сложность функции. Покрытие базовыми тестами — это один из видов покрытия кода, которое измеряется статически с помощью анализа кода и базовых тестов.

Чаще измерение покрытия кода происходит динамически, благодаря отслеживанию управляющей логики программы в ходе тестирования. Обычно инструменты по измерению тестового покрытия позволяют получить процент строк кода, выполненных в ходе прогона всего набора тестов. Грубые инструменты, например профайлеры, позволяют узнать, выполнялись ли функции[1]. Просматривая, что не было выполнено в ходе тестирования, тестировщики и менеджеры могут решить, нужно ли усиливать тесты для смягчения определенных технических рисков.

Обычно тестировщики поведения системы, особенно в независимых группах тестирования, фокусируют свои усилия на покрытии поведения, а тестировщики структуры, особенно программисты, обычно концентрируются на тестовом покрытии структуры. Однако при таком подходе можно упустить возможности заполнения пропусков в тестировании. Если используются методы черного ящика для проектирования и разработки тестов, последующее применение анализа покрытия поведения системы может сообщить только о том, не забыли ли мы что-нибудь. В то же время важно знать, не оставил ли метод покрытия поведения системы существенные технические риски без покрытия. Последнее означает, что применение методов анализа тестового покрытия структуры системы к множеству тестов, подготовленному средствами анализа поведения системы, может стать эффективным методом обнаружения и устранения пропусков в покрытии.

Тестовое покрытие (Test Coverage)

1. Структурное: степень покрытия тестами структуры — кода, подсистем или компонентов — в системе, предназначенной для тестирования.
2. Поведенческое: степень покрытия тестами поведения — рисков качества, режимов работы, функций, активностей и прочего применения — системы, предназначенной для тестирования.

[1] Оказывая услугу всему сообществу тестировщиков, главный консультант компании Tejas Consulting Данни Фот (Danny Faught) поддерживает обширный перечень инструментов тестирования, включая инструменты анализа кода, на сайте www.tejasconsulting.com.

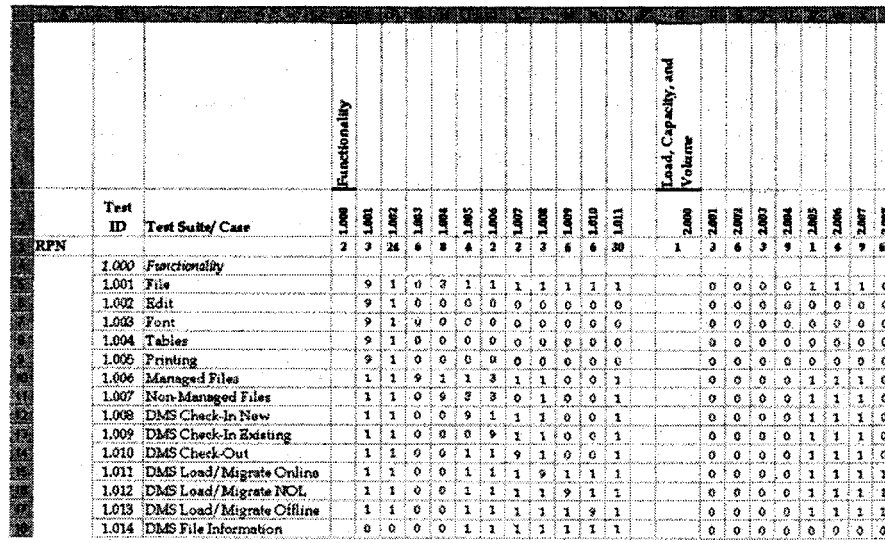
Джамал оценивает покрытие

Ноябрь приближался к концу, и разработка тестов близилась к завершению, лишь немного отставая от графика. Джамал был рад заметить, что он завершил изучение статистики покрытия рисков качества. По мере того как Эмма, Лин Цу и Динеш разрабатывали тесты, он просил их оценить релевантность тестов по отношению к каждому из рисков качества, определенных в ходе анализа FMEA. Для оценки релевантности использовались четыре категории, которым были присвоены числовые значения.

- 0 — покрытие отсутствует. Тестовый сценарий не затрагивает поведения системы, связанного с этим риском качества, и, таким образом, не воспроизводит связанных с поведением функций.
- 1 — вероятностное покрытие. Тестовый сценарий затрагивает незначительный пример поведения системы, связанного с данным риском качества, что приводит к случайному поиску связанных с риском сбоев.
- 3 — сбалансированное покрытие. Тестовый сценарий затрагивает существенный пример поведения системы, связанного с данным риском качества, что приводит к широкому поиску связанных с риском сбоев.
- 9 — подробное покрытие. Тестовый сценарий затрагивает существенный пример поведения системы, связанного с данным риском качества, и глубоко исследует поведение в этом месте системы, что приводит к расширенному поиску связанных с риском сбоев.

По мере получения этой информации о тестовых сценариях Джамал переводил детальное представление о покрытии на более высокий уровень системного тестирования. Он делал это с помощью таблицы, в которой с левой стороны сверху вниз были перечислены все тестовые сценарии, разработанные его группой. В верхней части таблицы он выписал основные категории рисков, их идентификаторы и значения приоритетов рисков из таблицы анализа рисков качества (видов ошибок и их влияния). Для каждого пересечения тестового сценария и риска он ввел значение релевантности, полученное при оценке покрытия тестировщиком, который разрабатывал данный тестовый сценарий. Фрагмент этой таблицы показан на рис. 11.1, где во второй строке сверху записаны идентификаторы рисков качества.

Джамал заметил, что единственный риск качества, который остался непокрытым или находится за пределами уровня тестирования, определенного в строке "Рекомендованные действия" в таблице FMEA, относится к удобству использования. "Да, — понял он, — я забыл добавить набор тестов для этого, хотя мы даже выделили средства на изучение удобства использования. Я должен пойти в бухгалтерию и узнать, утвержден ли заказ на закупку по тому контракту". Он добавил набор тестов в таблицу отслеживания тестов и обновил таблицу покрытия рисков. Наконец он положил обновленный документ в проектный репозиторий.



	Test ID	Test Suite/ Case	Functionality	1.000	1.001	1.002	1.003	1.004	1.005	1.006	1.007	1.008	1.009	1.010	1.011	Load, Capacity, and Volume	2.000	2.001	2.002	2.003	2.004	2.005	2.006	2.007	2.008
RPN				2	3	24	6	8	4	2	2	3	6	6	30	1		3	6	9	9	1	6	9	54
	1.000	*Functionality*																							
	1.001	File		9	1	0	3	1	1	1	1	1	1	1			0	0	0	0	1	1	1	0
	1.002	Edit		9	1	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0
	1.003	Font		9	1	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0
	1.004	Tables		9	1	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0
	1.005	Printing		9	1	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0
	1.006	Managed Files		1	1	9	1	1	3	1	1	0	0	1			0	0	0	0	1	1	1	0
	1.007	Non-Managed Files		1	1	0	9	3	3	0	1	0	0	1			0	0	0	0	1	1	1	0
	1.008	DMS Check-In New		1	1	0	0	9	1	1	1	0	0	1			0	0	0	0	1	1	1	0
	1.009	DMS Check-In Existing		1	1	0	0	0	9	1	1	0	0	1			0	0	0	0	1	1	1	0
	1.010	DMS Check-Out		1	1	0	0	1	1	9	1	0	0	1			0	0	0	0	1	1	1	0
	1.011	DMS Load/Migrate Online		1	1	0	0	1	1	1	9	1	1	1			0	0	0	0	1	1	1	1
	1.012	DMS Load/Migrate NOL		1	1	0	0	1	1	1	1	9	1	1			0	0	0	0	1	1	1	1
	1.013	DMS Load/Migrate Offline		1	1	0	0	1	1	1	1	1	9	1			0	0	0	0	1	1	1	1
	1.014	DMS File Information		0	0	0	0	1	1	1	1	1	1	1			0	0	0	0	0	0	0	0

Рис. 11.1. Фрагмент результатов анализа покрытия рисков качества в проекте "Суматра"

№ этапа	Этап	Выполнено?
8.	Обновить перечень рисков качества на основе новых знаний, полученных при разработке последних наборов тестов. Оценить покрытие рисков качества новой системой тестов. Выявить остающиеся непокрытыми риски качества, для которых рекомендовано тестирование.	☑
9.	Повторить этапы 2 – 8, пока не будут покрыты все риски качества, для которых рекомендовано тестирование. Убедиться, что документация о рисках качества обновлена в проектном репозитории. Если ограничения бюджета и сроков вынуждают прекратить разработку тестов, подготовить отчет о непокрытых рисках качества и направить его руководству.	☑

В ходе планирования комплексного тестирования Джамал, Лин Цу и Эмма совместно с Дженни Кауфман и ее ведущими программистами провели анализ технических рисков, связанных с сетевой и системной архитектурой. Данный анализ рисков определяет порядок и объем тестов комплексного тестирования, которые должны быть спроектированы и разработаны инженерами-тестировщиками. Замыкая круг, Джамал попросил Эмму, ведущего инженера по комплексному тестированию, проверить каждый сценарий для интеграционного тестирования в отношении покрытия технических рисков. Теперь, с точки зрения архитектуры верхнего уровня, Джамал мог быть спокоен за покрытие технических рисков в ходе комплексного тестирования.

"Хорошо, – думал Джамал, – а что у нас происходит с проблемами архитектуры низкого уровня и ее реализации? Как насчет проведения анализа покрытия структуры в ходе раундов комплексного и системного тестирования? Что будет и что не будет протестировано с точки зрения покрытия кода? Мы немного вышли за временные пределы проектирования и разработки тестов. Однако по выделенным на эту работу часам все в порядке – получается около 30 часов недоработки". Джамал решил рассмотреть три варианта:

1. Приобрести инструмент оценки сложности кода, научиться им пользоваться, измерить сложность системы и увеличить тестовое покрытие особенно сложных областей.
2. Приобрести инструмент оценки сложности кода, научиться им пользоваться, измерить покрытие операторов и ветвей за счет тестирования системы, а затем заполнить пробелы, связанные с повышенным риском.
3. Использовать встроенные профайлеры, которые имеются в некоторых средах разработки, или бесплатные инструменты покрытия кода, чтобы попытаться провести недорогое измерение покрытия на верхнем уровне.

Джамал понимал, что для вариантов 1 и 2 нет ни времени, ни денег. Если бы разработка тестов не вышла за пределы бюджета и у Эммы, Лин Цу и Динеша было много свободного времени, он, вероятно, мог бы выбрать один из этих вариантов. Однако он мог попросить Динеша, который показал, что способен быстро обучаться, и имел некоторый опыт покрытия кода, посмотреть вариант 3. Если ничего не выйдет, не страшно, но, возможно, есть некоторые недорогие и не оказывающие сильного влияния пути решения проблемы. Он направился к Динешу, чтобы обсудить с ним эту идею.

"Привет, Динеш, есть минутка?"

"Да, Джамал, что случилось?" – сказал Динеш, поворачиваясь к Джамалу.

"Я знаю, что к настоящему времени ты завершил разработку скриптов для тестирования надежности и стабильности и до февраля не будешь начинать разработку скриптов для функционального тестирования системы управления документооборотом. Я вот думаю, можно ли поручить тебе побочный проект, помимо руководства выполнением тестов?"

"Еще бы!"

Джамал сказал: "Я закончил просматривать результаты анализа покрытия рисков качества – ты знаешь, я запрашивал значения релевантности по мере написания тестов, – и анализа технических рисков, который провела Эмма в ходе подготовки к комплексному тестированию. С точки зрения тестирования черного и серого ящика, у нас все хорошо".

"Это хорошо", – вставил, кивая, Динеш.

"Итак, – продолжал Джамал, – мне не известна только одна информация о покрытии: степень покрытия структуры, или белого ящика. Какой объем кода мы тестируем? Я знаю, что группа Дженни при проведении модульного и компонентного тестирования имела стандартное значение 90% и выше. Однако это другой тип тестирования. Мы могли бы найти различные виды дефектов, покрывая тот же код нашими тестами. В об-

щем, я хочу попросить тебя о следующем. Во-первых, можешь ли ты найти инструмент для быстрого и недорогого анализа покрытия кода? Возможно, бесплатный инструмент покрытия кода или что-то подходящее, что имеется в среде разработки".

"Должен ли он быть сложным? — спросил Динеш. — Я имею в виду, должны ли мы уметь измерять только покрытие операторов или нам нужно измерить покрытие условий и путей?"

"Вообще-то, — ответил Джамал, — я бы предпочел получить оценку покрытия условий и путей, но подозреваю, что нам придется остановиться на покрытии операторов или, возможно, только на покрытии функций. Но я открыт для любых твоих предложений. Если бы ты смог провести небольшое исследование и предложить несколько вариантов в ближайшие 3 — 4 дня, мы бы приняли решение".

"Хорошо, — ответил Динеш. — Думаю, что это выполнимо".

"Вторая часть поручения — выполнение анализа покрытия кода. Возможно, тебе потребуется помощь кого-то из группы сборки версий Ясухиро, чтобы подготовить инструментированные тестовые версии, подходящие для измерений, а затем установить их в тестовую среду. Чтобы узнать, какое у нас имеется покрытие, мы должны выполнить все тесты. Если мы сможем двигаться достаточно быстро, это может быть сделано в ходе раунда 3 комплексного тестирования и раунда 4 системного тестирования. Я бы не хотел делать это слишком поздно, поскольку всегда есть вероятность, что инструментированная версия программы ведет себя иначе, чем обычная версия".

"Понятно", — согласился Динеш.

"Наконец, когда мы выявим пробелы в покрытии (я не сомневаюсь, что мы их обнаружим), мне бы хотелось, чтобы ты поработал вместе с Эммой и Лин Цу над усилением тестирования. Думаю, что какая-никакая, но польза от этого будет, в зависимости от того, сколько времени и денег останется, по крайней мере, мы сможем уделить внимание элементам наибольшего риска. Вероятно, ты смог бы сравнить наше покрытие с покрытием, полученным при компонентном и модульном тестировании, чтобы понять, есть ли куски программы, которые никто не проверял".

"Понятно, — ответил Динеш, — думаю, что это интересное поручение. Я начну после обеда, сразу после того, как завершу анализ некоторых результатов тестирования, которые прислала Эмамалини сегодня утром".

"Хорошо. Встретимся завтра, чтобы посмотреть, как у тебя дела?"

"Хорошо", — ответил Динеш, поворачиваясь к своему дисплею.

№ этапа	Этап	Выполнено?
10.	Если время и ресурсы позволяют, повторить этапы 2 — 8, используя структурный анализ (например, сложность Маккейба (McCabe complexity) и покрытие кода), чтобы определить области системы, требующие дополнительного тестирования.	☑

Построение успешного процесса проектирования и разработки системы тестов

Полнота — это, конечно, одна часть проблемы. Есть и другие соображения по поводу правильного построения системы тестов. В своей основе проектирование и разработка тестов — это созидательный инженерный процесс, поэтому результаты этого процесса наряду с его атрибутами очень важны. Что делает и чего достигает группа тестирования при наличии успешного процесса?

Предоставление работоспособных инструментов для оценки качества

Работая над системой тестов, я стремлюсь построить нечто полезное для достижения более глобальной цели, которой в данном случае является оценка качества системы, предназначенной для тестирования. Что означает для системы тестов обладание такими возможностями?

В первую очередь для меня это означает, что система тестов покрывает риски качества системы, как это обсуждалось ранее. Иначе говоря, те разделы системы, которые с наибольшей вероятностью являются проблемными или наиболее важными для заказчиков и пользователей, я должен исследовать на наличие потенциальных отказов. Определенная степень риска всегда присутствует. Таким образом, в процессе анализа рисков качества совместно с рабочей группой проекта я оцениваю степени рисков и сопоставляю с ними объем тестирования. Система тестов, следовательно, должна позволить мне выполнить все необходимые виды тестов в определенном объеме, чтобы надлежащим образом покрыть риски.

В ходе тестирования мы постоянно ищем равновесие между глубиной понимания и широтой покрытия, между определенностью и продвижением работ. Стремление к определенности ведет к тому, что тестировщик создает излишне детализированные, повторяемые тестовые сценарии, которые затрагивают каждый уголок и каждую частичку свойств, функций и поведения системы в рассматриваемой области. Пока тестировщик направляет все усилия на написание таких тестовых сценариев, он не может заняться написанием других тестовых сценариев, покрывающих другие условия и другие риски качества. Таким образом, при желании покрыть еще одно условие и потребности покрыть достаточно хорошо все ключевые риски качества в рамках выделенных времени и бюджета требуется компромиссное решение.

В процессе анализа рисков в основном примере книги, представленном в главе 2, группа проекта "Суматра" присвоила один из четырех уровней рекомендованных действий большинству рисков качества.

- Подробное тестирование
- Сбалансированное тестирование
- Тестирование по возможности
- Отчет о наблюдаемых ошибках

Различные уровни тестирования — это точки в бесконечном спектре вариантов. Тестирование на том конце спектра, где оно осуществляется подробно, дает наивысший уровень доверия к оценке качества. Тестирование, при котором выдаются отчеты о наблюдаемых ошибках, только информирует нас о том, что поведение обычных пользователей системы позволяет тестировщику пропустить эту область рисков, если все наблюдаемые ошибки обнаружены.

В моем перечне имеются только четыре из возможных точек. Если есть желание, можно сделать более тонкую шкалу. Рекомендуемые действия служат подсказками инженеру-тестировщику. Я предполагаю, что инженеры-тестировщики будут использовать профессиональные оценки при проектировании и разработке подходящих тестов в рамках имеющихся рекомендаций.

К различным системам могут применяться различные шкалы рисков. Проект "Суматра" готовит пакет обработки текстов для профессионального применения. Таким образом, риски существенны. Влияние сбоев на производительность и даже на непрерывное осуществление бизнеса может быть очень серьезным. Системы, предъявляющие повышенные требования к безопасности, в частности медицинские программы, средства управления производством, военные компьютерные системы, имеют шкалу рисков, которая заставляет группы тестирования проводить более тщательное и подробное тестирование. Развлекательные системы, например компьютерные игры и DVD-плееры, имеют шкалу рисков, которая позволяет применить более мягкий подход. Независимо от того, какие риски качества есть у системы, предназначенной для тестирования, одни риски являются более важными, чем другие. Тестировщикам приходится тратить большую часть усилий на проектирование и разработку системы тестов, а также на выполнение тестов нацеленных на покрытие этих рисков.

Покрытие ключевых рисков имеет большое значение, но мы можем сделать это таким способом, что тесты не будут иметь отношения к мнению о системе заказчиков и пользователей. Как-то я работал с группой талантливых программистов, которая проектировала тест производительности для большого сервера под UNIX, использовавшегося в системе, предназначенной для тестирования. Производительность была действительно основным риском качества для той системы. Однако способ, с помощью которого они спроектировали и реализовали инструменты тестирования, позволял оценивать производительность только при 60%-ной загрузке системы. Группа тестирования обнаружила, что при 100%-ной загрузке система не просто не справляется с достижением заданных параметров производительности, а терпит крах. Система тестов, которую построили мы как тестировщики, правдиво представляла реальное поведение пользователей, в то время как система тестов, созданная разработчиками, этого не делала.

Эффективная система тестов должна быть высококачественной, правдиво отражающей мнение о системе пользователей и заказчиков. Процессы выполнения тестов должны способствовать использованию системы, предназначенной для тестирования, в манере, аналогичной поведению пользователя. Система тестов должна позволять тестиров-

щикам оценивать правильность поведения системы при всех последовательностях действий и пользовательских сценариях с применением всех важных наборов данных. Система тестов должна порождать все тестовые условия, необходимые для уравнивания мнения о системе тестировщиков с мнением пользователей и заказчиков. Это означает также, что тестовая среда должна быть похожа на среду заказчика или среду поставки системы, чтобы можно было обнаружить проблемы, связанные со средой, конфигурациями и совместимостью[1].

Эффективная реализация, применение, повторное использование и сопровождение системы тестов

Высококачественные системы тестов, ориентированные на критические риски качества, будут эффективными, но, честно говоря, я не обладаю всеми деньгами и всем временем мира. Это означает, что система тестов должна бы столь же эффективной, сколь рациональной. Словарь "*Merriam-Webster's Collegiate Dictionary*" определяет, что "*рациональный*" (efficient) означает для объекта, системы или человека быть "продуктивным с желаемым эффектом ... и без потерь". Другими словами, это определение говорит, что нужно потратить правильное количество времени и денег на каждую задачу, необходимую для достижения желаемого эффекта, в нашем случае – работоспособной системы тестов. Это определение имеет множество следствий для системы тестов.

Устранение ненужных потерь, с одной стороны, означает, что для создания системы тестов необходимо использовать минимальное количество времени и денег. Например, если бы я игнорировал столбец "Рекомендованные действия" в таблице анализа видов ошибок и их влияния и задался целью построить систему тестов, которая обеспечивала бы подробное покрытие всех выявленных рисков, это было бы непродуктивно. Покрытие рисков в требуемом объеме, и не более, есть часть эффективной работы.

С другой стороны, не надо тратить деньги на бесполезные элементы системы тестов. Приобретение инструментов без предварительного проведения анализа затрат и прибыли обычно приводит к феномену складирования закупленного на полку (shelfware). Некоторые компании в буквальном смысле напичканы автоматизированными инструментами тестирования, закупленными на миллионы долларов и сложенными в пыльных углах офисов. Рабское подражание неподходящим стандартам документирования может также привести к созданию огромного числа папок с тестовыми сценариями и процедурами, к которым обращались один или два раза, а затем никогда больше не открывали.

---

[1] В проектах, в которых поставляются большие и сложные конфигурации или предполагается, что различные пользователи будут работать с различными конфигурациями среды, тест-менеджер, возможно, обнаружит, что перед ним стоит задача, сопоставимая с "мини-проектом", цель которого собрать такую среду в тестовой лаборатории. Как работать в этих условиях, описано в главах 6 и 7 моей книги "*Managing the Testing Process, Second Edition*".

Необходимо уравновешивать эффективность проектирования и разработки с эффективностью применения системы тестов. В ранее приведенных примерах я показывал, что незначительное документирование тестирования в сочетании с общим низким уровнем квалификации группы тестирования приводит к значительным объемам тестирования ad hoc, покрывающего вновь и вновь одни и те же риски, тогда как ключевые риски качества остаются непокрытыми. Я также видел, что разумное применение автоматизированных инструментов тестирования приводит со временем к значительному повышению эффективности[1].

Что касается использования денежных средств, то я считаю полезным четко отделять бесполезные расходы и инвестиции. Потратить $1000 на инструмент управления дефектами — это, конечно, инвестиция. Первая же ошибка, которая не попала в эксплуатационную версию из-за того, что использовался хороший инструмент управления дефектами, вероятно, покроет все затраты на инструмент. Потратить $5000 на то, чтобы установить тестовый сервер в том же офисе, где сотрудники разрабатывают и выполняют на нем тесты, а не попытаться использовать такой же сервер, находящийся на удалении 500 миль, — также инвестиция. Я наблюдал, что потерянное время стоит дороже, чем $5000, уже в случае нескольких недель проекта тестирования.

Отфутболивание коленом, жадное "нет" покупке подходящих инструментов и среды, расходованию времени на разработку хороших тестовых сценариев, реализации процедур, которая требует больше времени, но каждый раз дает отличный результат, приводит к болезненной неэффективности. Недостаточные, низкокачественные системы тестов облагают налогом каждую задачу тестирования, которую выполняете вы и ваша группа, налогом в виде потерянного времени. Возможно, это — несколько минут здесь, несколько минут там, но иногда это выливается в потерю целого дня на тестирование. Как гласит пословица: "Сэкономил копейку — потратил рубль". Это еще одна из форм бесполезных расходов, точно такая же, как покупка инструментов тестирования, которые не будут никогда применяться. Использование системы тестов обычно включает в себя работы по настройке тестового сценария для выполнения и по его демонтажу. Например, вам может потребоваться загрузить большую базу тестовых данных заказчиков для подготовки к тестированию объема. Иногда эти работы можно ускорить за счет использования в нескольких тестовых сценариях. Это делает тестирование более рациональным, поскольку время настройки и демонтажа является чисто накладными расходами и не оказывает влияния на оценку качества.

Хотя имеют значение и другие соображения, в целом я предпочитаю, чтобы тестовые сценарии и другие активности тестирования выполнялись настолько быстро, насколько это возможно. Таким образом, при равном покрытии функциональное тестирование, которое выполняется вручную и требует три часа времени, обычно более предпочтительно, чем функциональное тестирование, которое выполняется вручную и тре-

---

[1] Как разумно применять инструменты автоматизированного тестирования, чтобы получить существенный и обоснованный возврат инвестиций в тестирование на основе повышения эффективности тестирования, можно прочитать в книге Марка Фьюстера (Mark Fewster) и Дороти Грэм (Dorothy Graham) "*Software Test Automation*".

бует четыре часа времени. Даже для автоматизированных тестов скорость выполнения имеет значение. Я предпочту автоматизированный тестовый сценарий, который выполняется в течение обеденного перерыва, сценарию, который запускается на всю ночь. (Конечно, я тоже использую чудесное ночное время, если это возможно, для выполнения автоматизированных тестов, поскольку рациональная система тестов позволяет мне выстроить целую последовательность автоматизированных тестовых сценариев, каждый продолжительностью в один или два часа, в 16-часовой суперсценарий.) Я смогу быстрее улучшить тест, который выполняется за один час, чем тест, завершения которого я должен ждать до следующего дня.

Кроме того, я считаю, что должен уравновешивать эффективность начальной реализации и применения с эффективностью повторного использования, особенно сопровождения. Часто бывает легко соединить простой инструмент тестирования и файл тестовых данных. Однако, если не продумать заранее, повторное использование этого инструмента или файла в других тестах или последующих проектах довольно сложно. Я полагаю, что для тех частей системы тестов, которые предполагается повторно использовать и сопровождать, требуется значительный объем вспомогательных действий и документирования.

В нашем примере в предыдущей главе диаграмма состояний для функционального тестирования, построенная Лин Цу, была одним из инструментов, примененным Эммой для проектирования и разработки тестов производительности. Лин Цу, вероятно, смогла бы подготовить свои тесты, мысленно представив переходы состояний и не потратив времени на составление диаграммы. Однако создание диаграммы — запись того, что ей было известно, для общего пользования — дало возможность повторно применить ее.

Эффективность по важности сопоставима с надежностью. Выполнение одних и тех же тестов много раз для получения правдоподобных результатов неэффективно. Автоматизированные тесты, которые нельзя выполнять без присутствия тестировщика, также неэффективны. Тесно связанные тесты, т.е. когда один тестовый сценарий определяет данные и начальные состояния для другого сценария, обычно страдают от последовательных сбоев. В этом случае один тест не проходит, в результате система оказывается в неподходящем для выполнения следующих тестов состоянии. Последующие тесты тоже не проходят, но не потому, что система, предназначенная для тестирования, неисправна, а потому, что система тестов хрупкая. Все эти ситуации приводят к пустой трате огромных ресурсов времени на тестирование и анализ и, более того, могут привести к различным проблемам интерпретации результатов тестирования (см. главу 13).

Два потенциальных источника неэффективности системы тестов обусловлены взаимозависимостью тестовых сценариев и общих наборов данных, которые используются в различных тестовых сценариях. В некоторых случаях тестировщики создают тестовые сценарии, которые устанавливают предусловия для последующих тестовых сценариев. Это означает, что если по какой-либо причине тестовый сценарий не может быть выполнен, все последующие тестовые сценарии, зависящие от

Тестовые данные (Test Data)

Набор входных данных, который используется в тестовом сценарии. Данные вводятся либо при помощи пользовательского интерфейса, либо из хранимого файла, либо из другой взаимодействующей системы.

Инструменты тестирования (Testing Tools)

Любая система, способствующая выполнению задач тестирования, таких как проектирование, реализация, верификация, выполнение, измерение и интерпретация тестовых сценариев, тестовых данных и результатов тестов. В качестве инструментов могут выступать таблицы, шаблоны, базы данных, комплексные заказные утилиты собственной разработки и коммерческие системы. Инструменты бывают комбинированные и интегрированные. Некоторые инструменты тестирования, в частности системы захвата/воспроизведения и утилиты покрытия кода, разрабатываются сразу как инструменты тестирования, другие (отладчики, языки скриптов) лишь используются в этом качестве.

Инструменты управления тестированием (Test Management Tools)

Инструменты, управляющие объектами систем тестирования и их результатами. Классическими примерами являются системы управления дефектами и тестами. Более продвинутые системы предоставляют полномасштабные средства для проектирования, реализации и выполнения тестов. Такие системы обычно интегрированы с инструментами тестирования, хотя иногда эти дополнительные возможности ограничивают полезность инструментария для тестов, поддерживаемых инструментом тестирования.

него, блокируются. Не проблема, если блокирование происходит из-за неработающей функциональности. Однако, если это просто вопрос логики тестирования, происходят ненужная задержка и ограничение процесса тестирования. В идеале я бы хотел иметь возможность выбирать любой тестовый сценарий и поручать его выполнению любому тестировщику в любой выбранный день в период выполнения тестов исключительно на основе приоритета, доступности тестовой среды и навыков тестировщика. Необходимость учитывать, кто и когда выполняет другие тестовые сценарии, усложняет этот процесс и часто приводит к ненужным помехам в выполнении запланированных тестов.

Общие наборы данных порождают аналогичные проблемы. Если один тестовый сценарий создает или обновляет данные для других тестовых сценариев, это порождает зависимость тестовых сценариев. Если два теста совместно используют одну и ту же запись или зависимые записи из множества таблиц или файлов, то эти два теста не могут быть выполнены в одно и то же время. Если эти тесты по случайности выполняются одновременно, это, вероятно, будет воспринято как обнаружение ошибки. В результате, в лучшем случае, нужно будет заново выполнить тест, а в худшем случае будет иметь место неверное толкование ситуации как ошибки. В первом случае будет израсходовано впустую время группы тестирова-

ния. Во втором случае не только время группы тестирования, но также время группы разработки и, вероятно, время всей группы проекта будет потрачено напрасно, кроме того, будет нанесен удар по доверию группе тестирования.

Наконец, вопросом эффективности является также удобство использования. Система тестов, которую группа тестирования считает удобной для применения, – это такая система, которую можно использовать эффективно и быстро. Чем больше автоматизирована система тестов, тем больше она становится похожей на программную систему. Даже ручные системы тестов должны проектироваться и разрабатываться так, чтобы упростить их применение группой тестирования.

Выбор подходящих методов

Эффективность, рациональность, надежность и повторное использование будут свойствами системы тестов с большей вероятностью, если мы выберем правильные методы. Еще в 1979 г. Гленфорд Майерс в книге "*Искусство тестирования программ*" выделил небольшое количество методов. Эти методы используются и по сей день. С тех пор было написано огромное количество книг и статей, в которых улучшаются методы, определенные Майерсом, и описываются новые. Разнообразная палитра методов, доступных профессионалам, позволяет нам работать более эффективно и рационально. В то же время широкий простор для выбора может привести нас в смущение и до некоторой степени к сектантству. Какой метод лучше? Это зависит от обстоятельств. Любой из методов *может быть* лучшим, но ни один *не является* таковым.

Общий принцип состоит в следующем: риски системы качества определяют тестовые условия, которые должны разработать тестировщики. В свою очередь тестовые условия определяют выбор методов. Помимо организационного и технологического контекста, есть два главных соображения. Первое – это уровень автоматизации (если есть) тестирования как противопоставление ручному тестированию. Другое соображение относится к тому, какое проводится тестирование: статическое или динамическое, структурное или поведенческое, на основе кода, данных или требований.

Автоматизация тестирования обычно выражается в способности системы тестов выполняться без участия или при частичном участии человека. Его действия могут заключаться в работе с системой, предназначенной для тестирования, в загрузке данных для нее и в сравнении полученного результата с ожидаемым. Это может быть сделано с помощью инструментов, как доступных на рынке, так и разработанных на заказ. Автоматизация тестирования, вероятно, может позволить (и некоторые производители коммерческих инструментов тестирования утверждают, что позволяет) тестировщику спроектировать и разработать полный комплект тестовых сценариев, а затем с небольшими затратами или вовсе без них бесконечно повторять тестовые сценарии. Те, кто пытался это сделать, говорят, что этот план не всегда срабатывает.

Некоторые виды тестов хорошо подходят для автоматизации. Например, я использую инструменты автоматизированного тестирования для

большей части тестирования производительности и надежности. Функциональное регрессионное тестирование стабильной системы также хорошо подходит для автоматизации.

Однако некоторые виды тестов совсем не приспособлены к автоматизации. Например, в большинстве случаев тестирование обработки ошибок состоит в умышленном создании условий отказов сети, дисководов и подачи электроэнергии. Для человека не составит труда отключить кабель, дисковод или сетевую вилку, но машина этого сделать не может.

Один из моих клиентов попытался автоматизировать конфигурационное тестирование и обнаружил, что большая часть работы состоит в изменении операционной системы и приложений, установленных в тестовой среде. Автоматизация 10 – 25% работы, которая заключается в проверке системы после конфигурирования, не имеет большого смысла, особенно с учетом того, что работа разбита на части продолжительностью по часу, которые перемежаются часами работы вручную для комплекта тестов, выполняемого в течение недели.

Даже если автоматизация действительно имеет смысл, это попрежнему остается большим и рискованным мероприятием. Затраты, необходимые для автоматизации тестов, не равны затратам на написание тестовых скриптов, выполняемых вручную. Мой опыт показывает, что для создания автоматизированных тестов, удобных для сопровождения, должна быть выстроена целая инфраструктура, позволяющая тестировщику впоследствии определить действия, данные и ожидаемые результаты в некотором формате, отличном от программы. Простая запись действий, входных и выходных данных, а затем их воспроизведение не срабатывают в долгосрочной перспективе, за исключением особых обстоятельств, когда пользовательский интерфейс действительно не изменяется, а изменения касаются только бизнес-логики программы, не затрагивающей интерфейс.

Помимо затрат на создание тестовой инфраструктуры, часто серьезные вложения приходится делать в сами инструменты, которые могут стоить сотни тысяч долларов. Наконец, автоматизация тестирования предполагает внесение изменений в процесс тестирования и наличие других навыков, необходимых группе тестирования, причем все это должны быть управляемо. Поскольку существует много вариантов, при которых эта затея может пойти в неверном направлении, ходят ужасные рассказы о провалившихся проектах автоматизации тестирования. Один из моих клиентов никогда не произнесет в разговоре словосочетание "ав-

Тестовый скрипт (Test Script)

Описание того, как выполнять один и более тестовых сценариев при проведении тестирования вручную. Часто называют *тестовой процедурой*. При автоматизированном тестировании говорят о соответствующих последовательностях команд, выполняемых компьютером.

томатизация тестирования", поскольку однажды уже потратил сотни тысяч долларов на работы, которые не дали никакого результата. Инструмент лежит на полке без движения, скрипты, не приспособленные к сопровождению, удалены.

Тестирование не должно быть ручным или автоматизированным, оно может быть ручным и автоматизированным. Тесты, удобные для автоматизации, могут и во многих случаях должны быть автоматизированы, а методы ручного тестирования должны применяться ко всем другим тестам. В некоторых случаях единственный тестовый сценарий может быть и ручным, и автоматизированным. Функциональное тестирование Интернет-приложения, в котором я участвовал, включало в себя генератор загрузки для моделирования 50 000 пользователей, который работал в то время, когда мы выполняли функциональные тесты, чтобы посмотреть, как система будет реагировать на запросы и вести себя под нагрузкой.

В разговорах с опытными тестировщиками можно обнаружить широкое разнообразие мнений о правильном использовании автоматизации тестирования и окупаемости вложений в нее. Некоторые, особенно производители инструментов, склонны к излишнему энтузиазму относительно выгоды применения таких инструментов. Другие демонстрируют более прагматичные и даже скептические точки зрения[1].

Тесты, независимо от того, являются они ручными или автоматизированными, могут быть статическими или динамическими. Статический тест не приводит к выполнению исходного кода программы. Например, некоторые классифицируют рецензирование кода как статическое тестирование вручную. Анализ сложности кода, программы анализа стиля кода, например lint в C и C++, и даже компиляторы могут рассматриваться как автоматизированные статические тесты. Динамическое тестирование – это то, что представляют себе большинство людей, когда они говорят о тестировании. Это реальное выполнение некоторой части или всего кода системы, предназначенной для тестирования, и сравнение ожидаемых результатов с реальными.

Динамические тесты могут быть ориентированы на разные уровни системы, обычно их называют структурными и поведенческими (см. главу 1). Структурные тесты ищут ошибки на основе внутреннего устройства программы, а поведенческие тесты ищут ошибки на основе поведения системы, которое имеет внешние проявления. Более ранние фазы тестирования обычно больше ориентируются на структурные тесты, а более поздние, как правило, ориентируются на поведенческие тесты.

---

[1] Варианты точек зрения приведены в книгах Канера (Kaner), Баха (Bach) и Петтикорда (Pettichord) "*Lessons Learned in Software Testing*", Фьюстера (Fewster) и Грэм (Graham) "*Software Test Automation*", Дастин и др. "*Автоматизированное тестирование программного обеспечения*", на персональной страничке Канера www.kaner.com в разделе "Architectures of Test Automation" и в статье Баха "Test Automation Snake-oil" на www.satisfice.com. Обсуждая автоматизацию тестирования и тенденцию некоторых людей переоценивать возможности инструментов автоматизации, Канер писал мне: "Есть серьезная польза от применения многих инструментов тестирования, но эти продукты так активно рекламируются, а советы некоторых горячих сторонников столь ужасны, что на практике эти инструменты и их продавцы и консультанты часто приносят больше вреда, чем пользы".

Однако методы и инструменты структурного тестирования находят свое место и в системном тестировании. Генераторы загрузки для Интернет-приложения, о которых я упоминал выше, были получены из драйверов для структурных тестов, созданных программистами в ходе модульного тестирования для проверки основных программных интерфейсов в функциональности сервера. Это также один из примеров повторного использования средств тестирования.

Обычно я не выбираю какой-то один метод тестирования. Я рассматриваю каждое из условий, которые хочу выстроить в системе, предназначенной для тестирования, и выбираю наиболее подходящий метод для этого условия. В основном примере книги Эмма применяла разные методы для тестирования нагрузки, мощности и объема. Это можно сравнить с тем, что художник не выбирает наилучший цвет для создания картины, он использует целую палитру цветов, не задумываясь, смешивает их и меняет по ходу работы над картиной.

Предотвращение дефектов

В главе 2 отмечалось, что анализ рисков качества позволяет не только принять решение по поводу того, где искать ошибки. Он также может создать условия для того, чтобы избежать ошибок. Хороший процесс проектирования и разработки системы тестов может сделать то же самое. В основном примере Эмма обнаружила, что для диалогового окна открытия документа не задано, сколько файлов оно может показывать. Это могло бы выразиться в ошибке надежности, если бы программист не вставил проверки, что список слишком велик, поскольку переполнение могло бы вызвать крах системы. Либо могла возникнуть ошибка функциональности, если бы программист вставил проверку ограничения по своему усмотрению, но пользователи сочли бы его недостаточно большим.

Так часто бывает. Проектирование и разработка тестов заранее, до завершения программирования, позволяет вступить в дискуссию с программистами о том, как система должна работать. В этот диалог включаются системные проектировщики и программисты, пользователи и аналитики, а также все другие заинтересованные лица проекта. По мере создания системы тестов, я могу обсудить с заинтересованными лицами вопрос о том, что я собираюсь тестировать и какие результаты ожидаю получить. Довольно часто я получаю такие ответы: "Знаете, мы не думали о такой возможности". Каждый из таких ответов — это, по крайней мере, одна предотвращенная ошибка[1].

---

[1] В книге Гетцеля (Hetzel) "*Complete Guide to Software Testing*" обсуждается статическое тестирование, направленное на предотвращение дефектов. Борис Бейзер в "*Software Testing Techniques*" пишет, касаясь раннего начала тестирования в проекте, что иногда "угрозы тестирования" достаточно для предотвращения ошибок, поскольку программист, зная, что вы будете выполнять тот или иной тест, пишет программу так, чтобы не допускать в ней ошибок, которые этот тест может обнаружить. Другие отмечают, что все, что происходит до компиляции, является контролем качества, а не тестированием. Например, посмотрите книгу Уотса Хемфри (Watts Humphrey) "*Introduction to the Personal Software Process*".

Решение проблем

При проектировании и разработке хорошей системы тестов возникает много проблем, часть которых связана с технологиями и методами. Тесты для системы на web-основе отличаются от тестов для системы, предназначенной для универсальной машины. Автоматизированные и ручные, структурные и поведенческие тесты требуют разных подходов. Однако есть ряд общих проблем, которые заслуживают упоминания.

Сколько условий в тестовом сценарии?

Соображения эффективности и производительности, о которых говорилось выше, могут создать представление, что *большие* тестовые сценарии лучше, чем *маленькие*. (Слово "большие" я употребляю, имея в виду создание и оценку большого числа тестовых условий.) Если тестовые условия вытекают из рисков качества, то как только мы покрыли все тестовые условия, мы обеспечили должное управление рисками, для чего нас, собственно, и нанимали на работу. Тестировщики создают тестовые условия, выполняя тестовые сценарии. Подробности этого процесса рассматриваются в главе 13. Таким образом, основной единицей работы по выполнению тестов является тестовый сценарий. Таким образом, может показаться, что повышение эффективности работы связано с тем, что каждым тестовым сценарием надо постараться покрыть максимально возможное количество тестовых условий. Однако сочетание большого числа условий в одном тестовом сценарии может привести к повышению числа неудачных завершений теста.

Во-первых, вы можете наткнуться на несколько ошибок на одном шаге тестового сценария, что не позволит определить, какое изменение в состоянии системы привело к сбою. Во-вторых, один из сбоев может быть настолько исключительным по эффекту, что может скрыть другие, менее очевидные (но, возможно, столь же важные) сбои, а это приведет к пропуску дефекта в следующем тестовом цикле или попаданию его в эксплуатационную версию. В-третьих, можно получить отказ, который заведет тестирование в тупик и не позволит проверить условия, которые могли бы открыть последующие отказы, подчас не менее важные. В-четвертых, тестовый сценарий со многими условиями с большей вероятностью завершится неуспешно в отличие от сценария с меньшим числом условий, если в остальном они похожи. В результате отчет о ходе выполнения тестов, представленный в виде информации об успешно и неуспешно завершенных тестах, по всей вероятности, создаст пессимистическую картину, которая может повредить доверию группе тестирования.

Однако существует несколько исключений. Например, тесты, которые часто называют *приемочными*, проектируются для того, чтобы продемонстрировать стабильность системы и соответствие требованиям. В первом случае это делается для поставки версии заказчику, во втором – группе тестирования. Ожидается, что эти тесты пройдут без ошибок. Любые отказы означают, что необходимо вернуть систему передающей организации для дальнейшей работы. Основная цель – завершить эти тесты

Приемочное тестирование, тестирование на исправность (Smoke Test, Sanity Test)

Тестирование, проверяющее, насколько стабильна предлагаемая на тестирование версия, чтобы можно было начинать штатное тестирование. Обычно это подмножество всего комплекта тестов, как правило, автоматизированное, затрагивающее все части системы, по крайней мере, поверхностно. Качественные приемочные тесты обычно довольно долго проверяют работу системы, чтобы проявились серьезные проблемы надежности и работоспособности. Термин "smoke test" (тест на задымленность) взят из электротехники: когда инженер включает цепь, первичный тест проверяет, не дымятся ли компоненты.

как можно быстрее, поэтому наличие большого числа условий в одном тесте — отличная идея.

Еще один пример. Не удается создать некоторое число интересных условий в одном тестовом сценарии, потому что набор действий, необходимых для получения целевого тестового условия требует прохождения управляющей логики, которая направляет вас через ряд других тестовых условий. Возвращаясь к основному примеру, заметим, что тестирование рисков качества, относящихся к эффективности миграции файлов (см. 2.008 и 2.009 в таблице анализа видов ошибок и их влияния), обязательно покроет риски, связанные с тем, работает ли вообще миграция файлов (риски 2.010 и 2.011).

Этот пример также описывает другую ситуацию, при которой покрытие нескольких условий неизбежно, когда одно и более действий связаны с интересующими нас тестовыми условиями, которые устанавливают значения данных или внутреннее состояние, необходимое для оценки целевого условия. Приведем более понятный пример. Предположим, что вы тестируете базу данных на обновление записи. В некоторых случаях, как в примере с базой данных, вы можете создать тестовые условия, которые вы не вычисляете явно, но можете сделать вывод об их состоянии. Пусть вы добавили запись, проверили, что счетчик записей увеличился на единицу, удалили запись, проверили, что счетчик уменьшился на единицу, а затем подтвердили, что добавленная запись отсутствует в базе. Вы можете сделать вывод, что добавление записей работает верно, хотя все, что вы реально проверили, это общее число записей и условие удаления.

Перетестирование сторонних компонентов

Другой проблемой, которая становится все более традиционной для тестирования, является интеграция сторонних компонентов в систему, которая подвергается тестированию. Здесь я имею в виду, что одна группа разрабатывает систему (группа разработки) для второй группы (пользователей и заказчиков). Система включает в себя фрагменты, предоставленные группе разработки третьей стороной (производителями, поставщиками или партнером по аутсорсингу).

Ситуация может быть еще более сложной из-за наличия большего числа сторон, т.е. нескольких внешних групп, поставляющих части системы. Сторонние компоненты могут быть разработаны на заказ или представлять собой часть коммерческого пакета. Проблемы, которые здесь возникают, бывают и политическими, и техническими.

С технической точки зрения, здесь есть две фундаментальные проблемы тестирования. Во-первых, тестирование, выполненное третьей стороной, может не соответствовать вашей оценке и расстановке приоритетов рисков качества системы. Во-вторых, тестирование, выполненное третьей стороной, вероятно, не предполагало тестирование компонента, интегрированного в вашу систему, поскольку группа тестирования третьей стороны независимо от квалификации может не прочувствовать, каким будет мнение о системе ваших заказчиков и пользователей. Вы можете преодолеть эти проблемы, изучив тестирование, проведенное поставщиками и заполнив пропуски в вашем собственном тестировании. Однако часто бывает полезно объединить усилия групп, направленные на тестирование компонента[1].

Политические проблемы, связанные с любым из подходов, имеют большое значение. Поставщик компонента может отказаться афишировать результаты тестирования и предоставлять вам систему тестов. Такая информация о тестировании может содержать нелестные отзывы о качестве компонента, который они продают вам, или о беспорядке, который выдается за процесс разработки в компании поставщика. Даже когда результаты тестирования хорошие, а процесс поставлен отменно, поставщик на законном основании может не доверять пользователю, которому эта информация могла бы быть предоставлена. В частности, я сталкивался с ситуациями, когда менеджеры проектов пытались использовать результаты тестирования для обсуждения скидок после состоявшейся продажи и даже для отмены контракта.

Я также встречался с хитростями, которыми пользуются поставщики при общении с заказчиками. Один из моих клиентов был приглашен на прогулку поставщиком программных решений. Поставщик продавал ему большую серверную программу, про которую утверждалось, что она была тщательно протестирована. Я сказал: "Отлично, предоставьте нам результаты тестирования и инструменты тестирования". Поставщик отказался, утверждая, что они являются конфиденциальными. Будучи довольно циничным, я немедленно предположил, что результаты тестирования показывают, что программа полна известных ошибок и, вероятно, протестирована неадекватно.

Они действительно пообещали предоставить инструмент, но потом взяли свои слова обратно. Оказалось, что инструмент был столь ужасным и столь слабо документированным, что никто, кроме одного инженера-тестировщика, не мог с ним работать. Затем они осмелились выставить дополнительный счет моему клиенту более чем на $2000 в день плюс накладные расходы за этого инженера-тестировщика, который потратил две недели, показывая, как использовать инструмент. Мой цинизм был

---

[1] См. некоторые идеи по этому поводу в главе 10 книги "*Managing the Testing Process, Second Edition*".

позже вознагражден, когда, как я и подозревал, тестирование обнаружило массу проблем в их программе, которые они должны были исправить.

Какой бы подход вы ни выбрали, советую четко зафиксировать решение этого вопроса до того, как подписан контракт и завершена работа над планом проекта. Ваши поставщики и другие представители третьей стороны будут вести себя более гибко и помогать ориентироваться в их сложном и опасном мире, если об этом договориться заранее. Кроме того, ваши менеджеры проекта должны знать заранее, что наличие сторонних компонентов означает дополнительное тестирование. Многие менеджеры проектов ошибочно полагают, что раз поставщик тестировал свои компоненты или утверждает, что он это делал, не требуется никакого дополнительного тестирования при интеграции компонента в систему.

Неясно сформулированные требования, профили пользователей и описания конфигураций среды

Я сделал акцент на значении высококачественной системы тестов, которая правдиво воспроизводит опыт качества пользователей и заказчика. Это предполагает, что я смогу понять, как и в каких средах будет применяться система. Иногда эта часть информации не предоставляется группе тестирования.

В некоторых случаях это просто проблема процесса описания требований, который либо слаб, либо отсутствует. Например, в одном из проектов моя группа получила документ с требованиями для сложной системы, который состоял из одной странички, а сами требования формулировались примерно так: "Пользователю предоставляется возможность работы с электронной почтой", "Электронная почта пользователя будет поддерживать вложения" и "Браузер будет обеспечивать доступ к популярным web-сайтам". Мы начали задавать вопросы: "Какая кодировка электронной почты поддерживается?", "Какие конкретно допускаются виды вложений в почту?", "Насколько большими по размеру могут быть вложения?", "Что произойдет, если "популярные" web-сайты содержат элементы ActiveX, cookies и т.д.?" Этим мы заставили проектную группу рассмотреть вопросы, которые, если их не решить, превратятся в дефекты. Но это потребовало 20–30% дополнительных расходов на разработку тестов[1].

Иногда процесс сбора требований недостаточно детализирован, поэтому трудно точно определить, что требуется. Мои коллеги и я создали инструмент тестирования для клиента, который был в середине процесса реинжиниринга системы. Было трудно предсказать заранее, какие функции должен иметь инструмент и в какой среде он будет работать. В результате мы применили спиральный жизненный цикл разработки системы, создавали и поставляли прототип каждые три-четыре недели и спрашивали пользователей (в данном случае, группу тестирования клиента): "Это то, что вам нужно?" Обычно мы получали ответы следующего содержания: "Да, но нам нужно уметь делать также это и это".

---

[1] Мне нравится книга Карла Вигерса (Karl Wiegers) "*Software Requirements*" за рассмотрение хороших требований и за грамотно выстроенные процессы их сбора.

У меня накоплен большой опыт работы в ситуациях, когда по разным причинам требования определены нечетко. Мои тестировщики обнаружили, что в таких ситуациях можно эффективно работать за счет увеличения взаимодействия с проектной командой. Мы задаем большое количество вопросов о том, как система должна работать, что пользователи будут с ней делать, в каких средах она будет работать. Также помогут прототипы, имитации и модели. Сотрудники технической поддержки, систем сетевой поддержки, аналитики и пользователи текущих версий системы могут рассказать, что пользователи делают с системой в большинстве случаев. Специалисты по продажам, маркетингу, бизнес-аналитики расскажут об ожиданиях от новой системы.

Я также считаю, что в группу тестирования должны входить сотрудники с более высоким уровнем знания предметной области, если требования, применение системы и эксплуатационные конфигурации среды слабо определены или это трудно сделать. Если группа тестирования не располагает точным определением ожиданий пользователей о поведении системы при конкретных условиях тестирования, мне нужно, чтобы это определение могла дать сама группа.

Одно из эмпирических правил, которое я применяю в проектах с неясно определенными требованиями, заключается в следующем: "Есть сомнения – пиши отчет о дефекте". Это означает информирование о дефекте всякий раз, когда невозможно с основанием утверждать, прошел тест без ошибки или нет. Это приводит к тому, что проектная группа, использующая процессы управления дефектами и управления изменениями (см. главы 14 и 16), исследует непонятное поведение системы. Поскольку это происходит чаще всего в процессе выполнения тестов, а не в процессе их разработки, необходимо документировать тестовые сценарии с достаточной степенью неоднозначности[1].

Уравновешивание покрытия, сроков и бюджета

Процесс, представленный в этой и предыдущей главах, страдает от свойственного ему риска размывания границ. Следуя ему буквально, группа тестирования разрабатывает тесты в порядке приоритета, а это означает, что полное покрытие для элементов с высоким приоритетом появляется раньше, чем хоть какое-то покрытие будет создано для элементов с низким приоритетом. Это нормально, если у вас достаточно времени и есть дополнительное время на разработку всех тестов, которые были зафиксированы в период планирования. Но что если вы вышли за временные рамки?

Независимо от тщательности выполнения оценки затрат иногда я делаю в ней ошибки. Я обнаружил, что наиболее часто занижаю оценки для задач, которые связаны с разработкой инструментов и автоматизированных тестов. Поэтому я тщательно отслеживаю ход работ относительно графика, по крайней мере, еженедельно. Если задача выходит за рамки

---

[1] Дополнительные идеи по этому вопросу можно найти в статье Джоанны Ротман "Testing in the Dark" в журнале "*Software Testing and Quality Engineering*", Volume 1, Issue 2 (Mar/Apr 1999), которая доступна в настоящее время на www.jrothman.com и www.stickyminds.com.

сроков, я должен быстро понять, как скажется эта задержка на выполнении тестов. Нежелательно и часто политически недальновидно задерживать начало тестирования компонента, сборки или полной системы только из-за того, что средства тестирования, процесс выполнения тестов или тестовая среда не готовы.

Я полагаю, что должен быть честным перед самим собой в таких ситуациях. Я бы предпочел не использовать в этом случае план на случай непредвиденных обстоятельств. Все-таки сначала я попытаюсь следовать части моего основного плана, а не плана на случай непредвиденных обстоятельств. Однако мои предпочтения не всегда способны управлять реальной ситуацией. Поэтому я должен четко представлять, что происходит. Если какой-либо элемент проекта не находится под контролем, наступает время, когда я должен пересмотреть старый подход и выбрать новый.

Навязанные извне изменения плана или доступности ресурсов также могут повлиять на возможности выполнения всех запланированных тестов. Я обнаружил, что могу свести к минимуму влияние этих ситуаций, если разрабатываю тесты небольшими частями. Другими словами, разработка тестов должна обеспечивать для начала некоторый уровень тестирования и достаточно часто добавлять дополнительные инкременты системы тестов, а не вестись по принципу: все или ничего. Спиральный и итерационный жизненный цикл по этой причине наиболее удачно подходит для разработки системы тестов, особенно для инструментов тестирования и автоматизированных тестов.

В некоторых случаях вы, вероятно, обнаружите, что нужно вновь рассмотреть соотношение между подробным и широким покрытием. Выше говорилось о шкале: подробное тестирование — сбалансированное тестирование — тестирование по возможности — отчетность о наблюдаемых дефектах. Подробное тестирование обычно стремится аккумулировать непропорционально большой объем всего тестирования, включая разработку системы тестов.

Итак, имеет смысл для начала разработать сбалансированные тесты для этих рисков качества системы, а затем вернуться и улучшить полученный набор тестов, как только будет завершена разработка остальной части тестового покрытия. Если сначала сосредоточиться на разработке подробных тестов для наиболее критических рисков качества и проигнорировать риски более низких приоритетов, можно исчерпать время, отведенное на разработку тестов, раньше, чем удастся приступить к рискам низшего приоритета. Это означает, что у вас будет много тестов, покрывающих очень небольшое множество рисков системы качества, и совсем не будет тестов для проверки других областей. Это, пожалуй, не то, что нужно. Честное обсуждение и корректировка курса совместно с руководством и другими заинтересованными в тестировании лицами в случае, если разработка тестов выходит за рамки сроков, — наилучший способ провести переоценку соглашений и пересмотреть решения, принятые ранее.

Внедрение усовершенствований

Как улучшить разработку системы тестов? Многие ответы на этот вопрос включают в себя овладение новыми методами и технологиями. Это, безус-

ловно, важно, но зависит от конкретной ситуации и поэтому не обсуждается в моей книге. Однако есть несколько общих соображений, касающихся самого процесса, которые я могу предложить для рассмотрения.

1. Оцените качество существующей системы тестов. Отлично, если вы начинаете с нуля. Вы можете построить совершенную систему тестов с самого начала. Однако если у вас уже есть система тестов, спросите себя, хороша ли она. Если вы ответите "нет", можно ли изменить процесс и внедрить усовершенствования, которые позволят повысить качество тестирования, какой бы проект ни осуществлялся? Конечно, если вы увеличите тестовое покрытие в середине проекта, вам могут предъявить претензии, что вы необъективно завышаете планку качества. Очень важно получить одобрение со стороны заинтересованных лиц. Но я бы поддержал вас, если вы не игнорируете задачу при условии, что она политически неприятна. Когда я поступал таким образом, то несколько раз сталкивался с тем, что люди говорили: "Мы не хотим повышать планку качества в середине проекта", но в то же время они не были преисполнены оптимизма относительно связанных с этим пропусков в тестовом покрытии.
2. Изучайте и применяйте достижения в технологиях тестирования. Часть процесса, который я предложил в этой главе, включает в себя выбор подходящих методов тестирования. С развитием инструментов тестирования, быстрой эволюцией технологий системной разработки и развитием новых идей тестировщиками-профессионалами методы тестирования, подходящие для решения определенных задач тестирования, постоянно меняются. Эти изменения не означают, что старые методы больше не подходят. Они означают, что доступны новые, более прогрессивные методы. Можно привести подходящую аналогию из медицины. Врачи обязаны проходить непрерывное обучение для сохранения своих лицензий, чтобы они могли применять последние достижения в области охраны здоровья и лечения заболеваний пациентов. Область тестирования программ развивается не менее быстро, поэтому мы должны изучать новые достижения, чтобы не сомневаться, что мы проводим тестирование настолько эффективно и рационально, насколько это возможно.
3. Примите долгосрочный план. Все эти рекомендации работают лучше, если реализуются в течение продолжительного времени. Системы тестов аналогично системам, которые мы тестируем, обычно достаточно инерционны. Способ движения вперед состоит в том, чтобы следить за возможностью сделать процесс разработки тестов лучше, особое внимание следует обращать на выявление того, как слабости в системе тестов, существующие сейчас, влияют на возможность проведения качественного тестирования, а затем отвечать на эти вызовы с помощью долгосрочного плана усовершенствований. Этот долгосрочный план должен также учитывать, что, поскольку вы пытаетесь улучшить систему тестов, должен развиваться и контекст тестирования. Любой долгосрочный план улучшения системы тестов и процесса ее разработки должен быть нацелен в будущее. Какими свойствами и каким поведением будет обладать система? Какие планируются изменения в компании? Ка-

кие существуют возможности для изменений, например желание руководства что-то усовершенствовать, и как их можно усилить?

4. Одобрите подходящую документацию. Существует множество шаблонов для документирования системы тестов. Но простое заполнение пропущенных мест в шаблонах не обязательно приведет к получению подходящей документации. Если у вас есть существующая система тестов, вы можете обнаружить, что стиль ранее сделанной документации слишком точен. В этом случае перевод документации к новому, одобренному стилю может быть началом вашей новой работы, которая должна быть со временем завершена. В случае слишком подробного документирования, вероятно, вам захочется, чтобы устаревшая документация постепенно была заменена документацией, оформленной в новом стиле. В любом случае маловероятно, что у вас окажется достаточно времени и средств, чтобы выполнить основные изменения стиля в один присест для большой системы тестов, поэтому постепенное изменение может стать наилучшим способом решения проблемы.
5. Оцените и согласуйте навыки тестировщиков. Это оказывает большое влияние на проведение тестирования. Это значит, что вам нужно выстроить группу тестирования вплоть до приема новых сотрудников на работу. По мере формирования вашего долгосрочного плана вам необходимо проверять перечень критических навыков, который вы составили ранее. Какие новые навыки неявно вытекают из изменений, которые вы намереваетесь сделать в системе тестов и процессе ее разработки? Вам потребуется план, для того чтобы группу тестирования подтянуть до предлагаемого уровня процесса тестирования. И в принципе, что хорошего в идеальном плане и в идеальном процессе, если у вас нет квалифицированной команды для их проведения в жизнь?

Как показывает мой опыт, в проектах, где мы использовали существующую систему тестов, наиболее простым подходом было постепенное усовершенствование. Начало с нуля приводит к другой совокупности проблем.

Итак, мы завершили подготовительную часть процесса тестирования. Мы подготовились к выполнению тестов, чтобы выяснить качество системы, предназначенной для тестирования. Пора перейти к этому.

№ этапа	Этап	Выполнено?
2.	Подготовка: собрать людей и подготовить тесты.	☑
2.A	Посредством найма и обучения создать команду специалистов-тестировщиков, обладающих соответствующими умениями, навыками, отношением к делу и мотивацией.	
2.B	Спроектировать, разработать, получить и верифицировать систему тестов, которую группа тестирования будет использовать для оценки качества системы, предназначенной для тестирования.	

Часть III

Проведение

Мы достигли момента основного логического перехода в процессе тестирования. Джамал и его группа завершили важную часть процесса тестирования. До сих пор описывалась *подготовка к тестированию.* Теперь обратим наше внимание на проведение тестирования.

Проведение тестирования – более понятная часть, чем планирование тестирования и подготовка к нему, но и эту фазу тестирования не всегда понимают правильно. Неверное определение цели, часто сопровождаемое стремлением "убедиться в том, что система работает», может породить новые проблемы. Если это происходит внутри группы, то при обнаружении ошибок могут иметь место недовольство и превышение сроков. Если это происходит вне группы тестирования, то при информировании о результатах тестирования могут возникать конфликты и неприятные неожиданности. Когда проектная группа, которая не смогла предвидеть необходимости изменений, должна принять решения в связи с плохими результатами тестирования, может возникнуть хаос.

№ этапа	Этап	Выполнено?
3.	Проведение: выполнение тестов и сбор результатов.	☐
3.A	Получение и установка тестовой версии, состоящей из всех или нескольких компонентов системы, предназначенной для тестирования.	☐
3.B	Назначение ответственных за выполнение комплекта тестов на каждой из тестовых версий, его отслеживание и управление им.	☐

ГЛАВА 12

Передача материалов: управление тестовыми версиями

Даже когда люди понимают необходимость передачи тестовой версии группе тестирования, лишь немногие осознают сложность этого процесса и проблемы, связанные с ним. Управление тестовой версией является завершающим этапом управления конфигурацией системы, контроля изменений, сборки системы и поддержки системной библиотеки. Как и тестирование, эти действия могут оказать очень большое влияние на проект, как положительное, так и отрицательное.

Чтобы получить при тестировании точную и имеющую смысл оценку качества, управление тестовыми версиями и все предыдущие этапы должны выполняться очень точно и быстро. Представьте испытание лекарства, когда все или некоторые дозы лекарства, принятые пациентами, были фальсифицированы путем добавления токсинов или ослаблены при помощи инертных материалов. С каким доверием вы отнесетесь к результатам испытаний безопасности и действенности лекарства? Аналогично, если невозможно установить с точностью до каждого составляющего модуля состав системы, тестируемой в каждом цикле, что мы сможем сказать о качестве этой системы?

Поэтому, чтобы процесс управления тестовыми версиями был работоспособным, мы должны начать с известных компонентов, которые известным образом отличаются от компонентов, исследованных на предыдущих циклах. Мы должны собрать эти компоненты в тестопригодную версию с учетом определенного уровня полноты (который соответствует фазе тестирования). Мы должны передать, установить и настроить эту версию в тестовой среде. Только после этого мы можем приступить к оценке качества.

Без программы, предназначенной для тестирования, не бывает тестирования. Каким бы очевидным ни было это утверждение, многие компании, разрабатывающие программное обеспечение, очень слабо поддерживают процессы управления тестовыми версиями. Часто бывает, что ответственность за сборку тестовой версии передается отделу тестирования, который плохо подготовлен к выполнению этих задач. В таких случа-

ях эти процессы определяются как ответ на возникающие проблемы. Мы же рассмотрим, как управлять этими процессами при помощи тщательного планирования, до начала сборки и рационально[1].

Процесс сборки тестовых версий

Процесс 8 – это реальный процесс сборки тестовых версий. Этого процесса будет вполне достаточно для продукта, который является приложением для ПК, системой на основе браузера, размещенной на сервере приложений, или простой клиент-серверной системой. Сложности, возникающие с разделяемыми (совместно используемыми) базами данных, заказным аппаратным обеспечением, и другие проблемы будут рассмотрены в конце главы.

Процесс 8. Процесс сборки тестовой версии

№ этапа	Этап	Выполнено?
1.	Отобрать содержимое для конкретной тестовой версии (исправления ошибок, новые свойства и документацию).	☐
2.	Реализовать изменения, необходимые для исправления дефектов и добавления новых свойств, по завершении кодирования и модульного тестирования разместить изменения в репозитории исходного кода.	☐
3.	Извлечь исходные файлы из репозитория; откомпилировать, скомпоновать или оттранслировать сборку; присвоить сборке (внутри нее и в репозитории) номер версии.	☐
4.	Провести приемочное тестирование сборки. Если тестирование завершается успешно, перейти к следующему этапу; если тестирование неудачно, выяснить, в чем проблема, исправить сборку и вернуться к предыдущему этапу.	☐
5.	Создать инсталлируемый образ сборки на внешнем носителе данных, упаковать его надлежащим образом и передать лицу, ответственному за его установку в тестовой среде.	☐
6.	Установить сборку в тестовой среде.	☐
7.	Провести приемочное тестирование сборки в тестовой среде. Если тестирование завершается успешно, начать тестовый цикл. Если тестирование неудачно, деинсталлировать сборку, возобновить предыдущий тестовый цикл и вернуть сборку группе разработки для повторения процесса, начиная с этапа 1.	☐

[1] В оригинальной версии этой главы, опубликованной под названием "Test Release Processes" в журнале "*Journal of Software Testing Professionals*", Volume 1, Issue 2, приведены многочисленные цитаты коллег- тестировщиков. Я удалил цитаты, чтобы сохранить общий стиль книги, но содержащиеся в них идеи те же. Я бы хотел поблагодарить за полезный вклад: Randall Becker, Ross Collard, Gary Dawson, Harvey Deutsch, Tim Dyes, Danny Faught, Peggy Fouts, Darcy Martin и Barbara Ruf.

Сборка, версия, тестовая версия (Build, Release, Test Release)

Устанавливаемый компонент аппаратно-программного обеспечения или системы либо система целиком, передаваемая группе тестировщиков для тестирования.

Большая сборка поступает в тестовую лабораторию

6 декабря в 14:01 Джамал присоединился к группе управления проектом "Суматра" для участия в еженедельном совещании комитета по управлению изменениями. Одним из вопросов повестки дня было завершение работы над составом инкремента 3. По плану инкремент должен быть передан на тестирование 13 декабря, а тестирование первой тестовой версии должно начаться 16 декабря. Поскольку группа тестирования "Суматры" к этому времени уже обнаружила ряд ошибок, Джамал хотел убедиться в том, что некоторым из ошибок, замедлявшим проведение тестирования, был присвоен достаточно высокий приоритет для первой сборки инкремента 3.

Кроме того, поскольку согласно плану разработки в инкремент 3 должна быть добавлена большая часть оставшейся функциональности, он хотел проверить, что этот план не поменялся. В самом деле, инженеры-тестировщики Эмма Мурхаус и Динеш Кумар назвали первую версию инкремента 3 "большой сборкой". Некоторое число тестов было заморожено в ожидании функциональности, запланированной для большой сборки. Джамал рассчитывал, что у него есть немного времени на то, чтобы обсудить со своей группой приоритеты и логику этих тестов.

"Итак, — сказала менеджер проекта Кейт Эрнандес, — все на месте". Джамал кивнул, извиняясь за минутное опоздание. "Начнем. Дженни, проинформируй о том, что у нас с инкрементом 3".

"В целом все очень хорошо, — ответила менеджер разработки Дженни Кауфман. — Гораздо лучше, чем это было лишь несколько дней назад. Двое моих сотрудников смогли исправить ошибку данных, которую группа Джамала обнаружила в понедельник в ходе комплексного тестирования инкремента 3 и которая мешала правильной работе всей программы. И Джон, наконец, понял, как исправить проблему синхронизации, которая блокировала работу программы".

Кейт бросила на Дженни лукавый взгляд, и Дженни добавила: "Помните, ее нашла Эмамалини в ходе системного тестирования инкремента 1, когда не отвечал менеджер иерархического хранилища? Оказалось, что это проблема синхронизации в серверном коде, который взаимодействует с сервером управления иерархическим хранилищем".

Дженни повернулась к Джамалу и сказала: "Я знаю, Джон уже поблагодарил Эмамалини, но, пожалуйста, передай ей и от меня спасибо, по-

скольку эту ошибку, вероятно, было бы трудно найти, если бы она не написала небольшой скрипт для воспроизведения отказа».

Джамал, кивнув, ответил: "Точно, Эмамалини — молодец! Мы должны подыскать ей местечко для постоянной работы".

Кейт ответила: "Я уже сообщила об этом наверх, Джамал. Я разговаривала с Гарольдом и Джоном, и мы должны обсудить несколько идей после совещания.

Мы думаем, что Эмамалини сможет реально помочь группе разработки в подготовке тестовых драйверов для модульного и компонентного тестирования"[1].

Джамал улыбнулся и подумал о том, что вновь хорошее отношение к делу группы сервиса окупается.

"Но мы здесь собрались не для этого, — подвела итог Кейт, возвращаясь к главной теме. — В любом случае, насколько я понимаю, Дженни, это означает, что мы завершаем инкремент 3 в срок — в следующую пятницу, 13 декабря. Это так?"

"Да, — ответила она. — Я думаю, что так. Остались лишь некоторые из расширенных административных функций. Вероятно, нам потребуется перенести их на инкремент 4".

Еще до того, как Джамал успел поднять руку, Кейт повернулась к нему и спросила: "Джамал, что это означает для тестирования?"

"Не знаю наверняка. Я бы хотел получить список всех свойств, которые не должны работать. Некоторые из тестов безопасности и секретности пытаются обойти ограничения доступа, чтения, изменения и удаления объектов, созданные администраторами. Также некоторые из функциональных тестов и тестов перегрузки зависят от не являющихся иерархическими свойств безопасности, которые должны работать с администраторами рабочих групп. Некоторые тесты могут быть заблокированы", — озабоченно закончил свое выступление Джамал.

"Я поняла, — ответила Кейт и обратилась к Дженни. — Дженни, после обеда подготовь для Джамала список свойств системы, которые могут быть не закончены". Дженни кивнула, соглашаясь. "Джамал, изучи, пожалуйста, список и сообщи мне завтра, как он влияет на тестирование. Если действительно есть проблемы, предложи пару вариантов смягчения рисков для каждого теста, который нельзя выполнить. Дженни, постарайся с утра найти время для того, чтобы помочь Джамалу". И Джамал, и Дженни согласно кивнули головами.

"Хорошо, помимо этих свойств, Дженни, ты разобралась с планом и проверила, все ли, что запланировано для инкремента 3, собрано или будет готово на следующей неделе?"

"Да, — согласилась Дженни, — все элементы, кроме свойств, которые мы только что обсуждали, либо завершены, либо являются несложными элементами с незначительным риском, так что мы не сомневаемся, что они будут завершены не позже следующей пятницы, а возможно, и раньше".

---

[1] Применение таких методов в реальных проектах рассматривается в статье "Mission Made Possible", написанной Рексом Блэком (Rex Black) и Грегом Кубачковски (Greg Kubaczkowski), опубликованной впервые в "*Software Testing and Quality Engineering*", Volume 4, Issue 3 (July/Aug 2002) и доступной в настоящее время на www.stickyminds.com и на www.rexblackconsulting.com.

"Хорошо. Следующий вопрос – отчеты о дефектах. Верно, Джамал?" – спросила Кейт.

"Абсолютно верно. Эта тема мне по душе", – улыбнулся Джамал.

"Дженни, пожалуйста, перечисли перечень дефектов, которые согласно графику должны быть устранены до передачи инкремента 3".

Дженни стала медленно перечислять дефекты, ожидая комментариев и вопросов: "1792, ошибка блокировки файла. 1793, 1798, 1799 и 1800, сообщения с ошибками правописания. 1699, ошибка блокировки, обнаруженная Эмамалини. 1787 и 1788 оказались дублированными ошибками; основная причина ошибки данных одна и та же".

Джамал, отмечавший ошибки в своем отчете по ходу выступления Дженни, прервал ее: "Ты в этом уверена? Эти ошибки были обнаружены при выполнении разных тестовых сценариев на разных данных".

Дженни ответила: "Да, я знаю, но на сервере базы данных были странные установки конфигурации, мы тут недосмотрели. Оказалось, что там есть целый блок ошибок, о которых вы могли бы сообщить, но основные пути к ним были заблокированы проблемами в инкременте 3".

Джамал пожал плечами и сказал: "Да, возможно, ты права. Не хотелось бы прослыть назойливым или предугадывать действия твоей группы, но если ты попросишь своих ребят все проверить еще раз, я буду очень признателен. Эти две ошибки заблокировали огромное число сценариев для комплексного тестирования, которые я бы хотел обязательно выполнить".

"Я проверю, – согласилась Дженни и вернулась к списку. Он перечислила еще десяток ошибок, которые должны быть исправлены. В завершение она спросила Джамала: "Есть ли в твоем списке еще какие-нибудь блокирующие ошибки или накладки, которые я не упомянула?"

"На самом деле, только одна, – ответил Джамал. – 1815. Я понимаю, что это новая ошибка, и просить исправить ее на этой неделе – требовать от тебя слишком многого. Но поскольку она не позволяет выполнить большую часть тестов с использованием действующих браузеров, я не сомневаюсь, что мы не сможем покрыть несколько важных конфигураций, если быстро не исправим ее".

"Хм, – сказала Дженни, – мне нужно обсудить этот вопрос со своей группой, но, похоже, мы должны с этим справиться. Среднее время закрытия ошибки у нас равно примерно 10 дням, включая накладные расходы, связанные с процедурой, верно?"

"Да, верно, – согласился Джамал, проверяя свои графики, – если это обычная ошибка, которая может быть исправлена в стандартные сроки, ты сможешь это сделать".

№ этапа	Этап	Выполнено?
1.	Отобрать содержимое для конкретной тестовой версии (исправления ошибок, новые свойства и документацию).	☑

Участники совещания перешли к другим темам, в том числе к обсуждению следующих инкрементов. Вскоре после совещания точный перечень функций инкремента 3 высокого риска, составленный Дженни, появился в почтовом ящике Джамала. Джамал распечатал четыре экземпляра сооб-

щения, а затем пригласил Эмму, Динеша и Лин Цу в небольшую переговорную.

"Итак, – объяснил Джамал, вручая списки, – в инкременте 3 эти свойства имеют большой риск. Я хотел бы, чтобы каждый из вас изучил, как отсутствие этих свойств или серьезные проблемы в них могут повлиять на выполнение ваших тестов, запланированных для инкремента 3. Если есть проблемы, попробуйте, пожалуйста, составить план смягчения рисков или план обходных путей. Пришлите мне результаты вашего анализа в середине первой половины дня, хорошо?"

Все согласно кивнули, и короткое совещание завершилось. На следующее утро Джамал получил сообщения от Эммы, Динеша и Лин Цу. Все предсказывали незначительные проблемы либо их отсутствие, если будут задержки с реализацией перечисленных свойств. Джамал быстро собрал всех в своем кабинете. "Я бы хотел убедиться в том, что у нас одна точка зрения: я посылаю сообщение группе управления проектом о том, что группа разработки может перенести эти свойства в инкремент 4".

Все кивнули, но Лин Цу добавила: "С обычными предупреждениями, верно?"

Джамал вскинул голову и спросил: "Обычные предупреждения?"

"Да. Есть риски, которые мы можем предусмотреть, и есть риски, которые мы не можем предусмотреть. Из опыта нам известно, что всякий раз, когда происходит задержка в получении некоторого свойства на тестирование, не сопровождающаяся откладыванием выпуска версии, у нас имеется меньше времени на тестирование свойства, а это означает, что свойство будет иметь более высокий уровень риска качества при эксплуатации".

Джамал согласно кивнул и пробормотал: "Да, конечно".

"Кроме того, – продолжала Лин Цу, – есть риски, которые мы не можем предсказать. Мы не знаем, как построена система с точки зрения организации архитектуры, верно? Поэтому мы не знаем, как проявит себя отсутствие этих свойств: замаскирует или вскроет поведение других свойств".

У всех поднялись брови, а Лин Цу продолжила. "Когда бы эти свойства ни были добавлены, они могут привести к возникновению других ошибок. Некоторые будут ошибками в новом коде, некоторые окажутся ошибками регрессии. Мы не можем точно оценить вероятность этого, поскольку мы не знаем подробностей того, как код, реализующий эти свойства, взаимодействует с другими частями системы. Когда эти свойства будут добавлены, проблемы, которые сейчас кажутся серьезными, могут оказаться менее серьезными. И наоборот. Программы – не автомобили. Если у меня проблема с настройкой радио, я могу, по крайней мере, предположить, что эта проблема не станет причиной спущенной шины[1]. Иногда возникают неразличимые и трудные для предупреждения последствия и взаимодействия. Я размышляла по поводу использования итерационного жизненного цикла и думаю, что чем короче период времени,

---

[1] Это меткое сравнение позаимствовано из серии из трех статей Бориса Бейзера "Software Is Different», которую можно найти в архиве Software Research's Quality Techniques Newsletter на www.soft.com/News/QTN-Online/qtnapr01.html, www.soft.com/News/QTN-Online/qtnmay01.html и www.soft.com/News/QTN-Online/qtnjun01.html.

в течение которого завершается последняя версия системы, предназначенная для тестирования, и само системное тестирование, тем больше вероятность того, что заключительная оценка качества системы – я имею в виду то, что происходит на совещании, посвященном завершению системного тестирования, - - не будет точной».

Четыре тестировщика довольно эмоционально отреагировали на это заявление.

"Итак, – продолжала Лин Цу, – я полагаю, что сейчас подходящее время для того, чтобы объяснить группе управления проектом, каково влияние задержек передачи версии на системное тестирование и на жизненный цикл итерационной разработки в целом". "Не хочу сказать ничего плохого об итерационном подходе, – поторопилась добавить она. – До сих пор все шло значительно лучше, чем в некоторых других проектах. Но риск – это всегда компромисс, не так ли? Чтобы получить небольшое пространство для маневра в последней версии, мы, вероятно, должны немного увеличить риск. Полезно заметить, что мы – "группа управления рисками качества", верно?"

Джамал медленно расплылся в улыбке: "Очень мудрые мысли, Лин Цу, особенно о рисках, которые мы не можем предвидеть. Я еще раз расскажу об этом Кейт, когда буду писать ей письмо".

После этого разговора Джамал подготовил черновик письма Кейт с копией для Дженни, в котором объяснил, что поскольку задержка свойств администрирования не окажет влияния на тестирование с точки зрения его эффективности, можно договориться о смягчении рисков. Он, в частности, писал: "Как мне подсказала Лин Цу, есть два вида рисков: предсказуемые и непредсказуемые. Мы можем предвидеть, что меньший объем тестирования этих свойств приводит к большему риску при поставке. Мы не можем предсказать, однако, направления, в которых отсроченное тестирование системы с полным набором свойств искажает точность наших оценок качества. (Между прочим, нужно сделать предложение Лин Цу стать менеджером.) Моя группа согласна с тем, что мы можем разрешить перенести эти свойства в инкремент 4. Однако, если какое-либо из основных свойств, аналогичных этим, не войдет в инкремент 4, советую обсудить на совещаниях группы по контролю над изменениями, можно ли исключить это свойство из последующих версий сопровождения".

Дженни и Кейт одобрили это сообщение. Джамал был рад услышать, как Кейт и Дженни благодарили Лин Цу за ее наблюдательность.

Утром в пятницу Дженни отправила письмо группе управления проектом, сообщающее о том, что инкремент 3 помещен в проектный репозиторий. Она приложила к письму список изменений в версии. К счастью, только одного из шести свойств повышенного риска не было в новом инкременте. Она попросила менеджера сборки версий Ясухиро Канагаву подготовить версию и провести приемочное тестирование как можно быстрее, чтобы она смогла в этот же день или в предстоящие выходные, если потребуется, найти и разрешить возникшие проблемы.

Пару часов спустя Ясухиро прислал короткое сообщение: "Версия отличная. Результаты приемочного тестирования прилагаются. Кейт, пожалуйста, договорись с Джамалом об установке".

№ этапа	Этап	Выполнено?
2.	Реализовать изменения, необходимые для исправления дефектов и добавления новых свойств, по завершении кодирования и модульного тестирования разместить изменения в репозитории исходного кода.	☑
3.	Извлечь исходные файлы из репозитория; откомпилировать, скомпоновать или оттранслировать сборку; присвоить сборке (внутри нее и в репозитории) номер версии.	☑
4.	Провести приемочное тестирование сборки. Если тестирование завершается успешно, перейти к следующему этапу; если тестирование неудачно, выяснить, в чем проблема, исправить ее и вернуться к предыдущему этапу.	☑

Кейт и Джамал столкнулись на пути к друг к другу и рассмеялись. "Близится вечер пятницы, а, Кейт?" — пошутил Джамал.

"Не похоже, что ты собираешься здесь торчать после пяти, пижон", — ответил Кейт.

"Ты прав. Не собираюсь. Семья приезжает на выходные, и я должен быть с ними. Это Остин, поэтому здесь много развлечений в пятницу вечером".

"Еще бы! Итак, — сказал Кейт, переходя к делу. — Какой у нас план?"

"Я бы предложил сделать, как раньше? У нас есть принятый процесс для установки сборок, поэтому будем следовать ему. Если ты не думаешь, что раньше были проблемы?"

"Нет-нет, я думаю, что он отлично работает, Джамал, — ответил Кейт. — Ты прекращаешь тестирование в 4 часа дня, а я затем прошу Петру установить сборку. Как только она закончит, думаю, не позже, чем через час, твой младший тестировщик... Напомни, как его зовут?"

"Дейл".

"Верно, Дейл. Дейл сможет запустить приемочный тест в тестовой среде перед уходом с работы".

"Отличный план", — сказал Джамал.

"Конечно, я думал о нем, — пошутил Кейт. — Петра и Дейл уже договорились о деталях, поэтому я думаю, что мы должны предоставить им кредит доверия".

"Знаешь, — сказал Джамал, — есть управленческое клише: нет конца процессу кредитования, если Вы не следите за тем, кто получает кредит. На самом деле более точной является фраза: нет конца процессу кредитования, если Вы не следите за тем, когда сообразительные люди делят кредит".

"Подписался бы под этой фразой, — ответил Кейт. — В любом случае я собираюсь сказать Петре, чтобы она ждала твоего звонка, хотя не сомневаюсь, что она уже ждет. Ты планируешь заехать в выходные и проверить результаты приемочного тестирования?"

"Возможно, я проверю их через Интернет. Нет смысла приезжать в выходные в офис, если проект идет так хорошо".

Проект и дальше шел хорошо. Перед тем как Джамал отправился в аэропорт Остина встречать семью, он подошел к Дейлу.

"Как там приемочный тест, Дейл?"

"Уже запущен. Пока проблем не возникло".

"Похоже, вы с Петрой наладили процесс?"

"Она инсталлирует, я запускаю приемочный тест, а потом мы все тестируем. Все просто, как дважды два", – ответил Дейл, улыбаясь.

"Думаю, что хороший процесс всегда прост, Дейл. Чтобы так поумнеть, пришлось убить немало проектов, но мы становимся умнее благодаря упрощению".

"Хороших выходных, босс", – сказал Дейл.

"Да, и тебе тоже, Дейл". – Джамал взглянул на часы. Они показывали 16:50. Джамал улыбнулся и пошутил: "Знаешь, Дейл, учитывая твою напряженную работу, почему бы тебе ни отдохнуть остаток дня?"

№ этапа	Этап	Выполнено?
5.	Создать инсталлируемый образ сборки на внешнем носителе данных, упаковать его надлежащим образом и передать лицу, ответственному за его установку в тестовой среде.	☑
6.	Установить сборку в тестовой среде.	☑
7.	Провести приемочное тестирование сборки в тестовой среде. Если тестирование завершается успешно, начать тестовый цикл; если тестирование неудачно, деинсталлировать сборку, возобновить предыдущий тестовый цикл и вернуть сборку группе разработки для повторения процесса, начиная с этапа 1.	☑

Построение успешного процесса подготовки тестовых версий

Как отметил наш вымышленный герой Джамал, хороший процесс прост, но это не значит, что создать хороший и простой процесс также просто. Участвуя в большом количестве проектов, я заметил, что это особенно верно для управления тестовыми версиями. Есть общий соблазн передать поскорее то, что сделано, на тестирование, как будто разработка программ – это форма позиционной войны. Требуется тщательное и вдумчивое отношение к внедрению качественного процесса подготовки тестовых версий. Давайте посмотрим, что делает и чего добивается проектная группа, применяя качественный процесс сборки версий для тестирования.

Подготовка инсталлируемых, усовершенствованных и стабильных тестовых версий

Однажды я работал с компанией, в который был выстроен отменный процесс взаимодействия с заказчиками. Когда я отправлял запросы при помощи быстрого, автоматизированного web-сайта, она присылала приятные письма о своей приверженности качеству и решению моих проблем. Была только одна помеха: они не оказывали качественных услуг. Они не решили ни одной из проблем, о которых я сообщил их службе поддержки клиентов. Поэтому я больше не являюсь заказчиком этой компании.

То же самое применимо к процессу подготовки тестовых версий. Ключевым индикатором хорошо поставленного процесса является то, что процесс каждый раз завершается передачей сборки, пригодной для тестирования. Все другие признаки качественного процесса могут присутствовать. Но предположим, что каждая тестовая версия превращается в кошмар из-за впустую потраченного времени на установку сборки, из-за путаницы при регрессионном тестировании и позорной ямы заблокированных тестов, аварий в середине тестирования и других показателей нестабильности. Разве вы не придете к выводу, что процесс сборки версий имеет серьезные проблемы?

Качественный процесс подготовки тестовых версий передает сборку, которая может быть инсталлирована. Процесс инсталляции должен, по крайней мере, разместить элементы сборки в соответствующих местах, с соответствующими правами доступа и в правильном порядке. Если для успешной установки требуется выполнение предваряющих инсталляцию или последующих за ней действий, они должны быть либо автоматизированы, либо документированы надлежащим образом для сотрудников, которые будут заниматься установкой. Для многих системных сред, включая Windows и UNIX, существуют коммерческие инструменты для создания инсталлируемых версий и пакетов.

Качественный процесс подготовки тестовых версий должен передавать сборку, которая лучше предыдущей. Это означает, что то, что работало раньше, по-прежнему работает, и что часть сборки, которая не работала раньше, работает лучше, чем в предыдущих тестовых версиях. В главе 15 будут рассмотрены два индикатора того, что проект подходит к успешному завершению. Это устойчивое сокращение общего числа открытых дефектов и постепенное уменьшение числа новых дефектов, обнаруживаемых в очередной версии. Таким образом, общая тенденция должна быть такой, что происходит улучшение состояния системы от версии к версии.

Общая тенденция улучшения версий частично является результатом тщательной разработки и исправления ошибок программистами. В одном из проектов примерно 20% отчетов о дефектах были повторно открыты хотя бы однажды, т.е. ошибки, о которых разработчики говорили, что они исправлены, либо не были на самом деле исправлены в тестовой версии, либо в одной из версий были исправлены, а затем возникли снова. Тестировщикам пришлось заново открывать один из отчетов о дефектах десять раз! Основные процессы, программирование и отладка, должны сопровождаться адекватным модульным, компонентным и, если необходимо, комплексным тестированием.

Лучший способ, который я когда-либо встречал для таких видов тестирования, предполагает участие разработчиков в подготовке тестовых сценариев для кода по мере его написания, а затем помещение тестовых сценариев в автоматизированные тестовые драйверы для проверки кода. Это непрерывно растущее множество тестов выполняется для каждой сборки в качестве элемента процесса сборки с целью проверки того, что отдельные части сборки работают верно. (Вообще говоря, качественное тестирование силами разработчиков, особенно на фазах модульного и компонентного тестирования, — это один из признаков качественного

процесса тестирования, как будет показано в главе 17.) Этот метод, являющийся общепринятой практикой уже в течение десяти лет, в настоящее время получает признание в связи с приходом различных методологий быстрой разработки программ, которые усилили идеи разработки с первоначальным тестированием и применения автоматизированных тестовых драйверов[1].

Другое усовершенствование процесса — это тщательный контроль изменений и управление конфигурациями программного продукта. Если два сотрудника работают с одним и тем же модулем одновременно, без библиотеки, управляющей исходных кодом, используя многочисленные копии дерева кода в различных папках и на разных серверах и выполняя прочие хаотические действия над составляющими систему компонентами, это естественным образом ведет к сборке неудачных версий. Нет конца бедствиям, которых можно избежать за счет простого, облегченного режима конфигурационного управления. Наряду с организацией модульного, компонентного и комплексного тестирования при помощи тестовых драйверов, это еще один лучший метод организации работ, который завоевал популярность в последние десятилетия[2].

Сейчас даже при наличии инсталлируемых сборок, проведении качественного модульного, компонентного и комплексного тестирования и эффективном управлении конфигурацией программного обеспечения существует вероятность передачи группе тестирования неудачной сборки. Иногда фрагменты работают по отдельности, но как только их объединяют вместе, возникают проблемы, которые проявляются только в определенных частях продукта и при определенных условиях. Многие из нас слышали смущенный возглас программиста: "В моей системе это работает отлично".

Конечно, до передачи системы на тестирование невозможно гарантировать, что она пройдет успешно все тесты. Если бы мы могли это гарантировать, нам не нужно было бы ее тестировать. Однако правило 80/20 поможет избежать напрасной траты времени[3]. Большая часть серьезных проблем — ситуации, блокирующие дальнейшее выполнение тестов, крах системы, разрушение целостности данных — обнаруживаются в первые часы тестирования. Это особенно верно, если вы начинаете с широкого покрытия. Это правило является движущей силой, в дополнение к идее

---

[1] И Гленфорд Майерс в книге "*Искусство тестирования программ*", впервые опубликованной в 1979 г., и Фред Брукс в книге "*Мифический человеко-месяц*", опубликованной впервые в 1975 г., рассматривают тестовые драйверы. Современные тестовые драйверы исследуются в статье "Mission Made Possible", упомянутой ранее.

[2] В моем первом проекте разработки для UNIX в 1983 г. применялся строгий процесс конфигурационного управления под контролем библиотекаря, использовавшего утилиту SCCS. *Даже тогда* это был проверенный десятилетиями опыт организации работ. Фред Брукс описывает систему "библиотеки программ и учета", которая применялась в проекте IBM OS/360, в главе "Острый инструмент" книги "*Мифический человеко-месяц*".

[3] Правило 80/20 существует в разных вариациях. Я использую версию для менеджмента, суть которой состоит в том, что 80% прибыли от любой деятельности обычно приносят первые 20% затрат на нее. Правило 80/20 более формально называется принципом Парето, который в нынешнем виде был впервые описан Дж. М. Джураном (J. M. Juran). Интересное обсуждение этого правила и его приложений к тестированию см. в статье Эрика Петерсена (Erik Petersen) "Smarter Testing Using the 80/20 Rule" в "*Proceedings of the STAR West 2002 Conference*", доступной на www.stickyminds.com.

приемочного тестирования. Еще раз повторю, что это не новая идея, поскольку в конце 80-х годов я работал в проекте, в котором активно применялось приемочное тестирование, встроенное в процесс сборки версий.

Подготовка предсказуемых, своевременных и регулярных тестовых версий

Даже при условии, что новая версия является инсталлируемой, улучшенной и стабильной, приемка ее на тестирование означает необходимость в проведении ряда действий. Тест-менеджер должен обсудить с группой тестирования, какие тесты они должны выполнить для этой сборки. Кто-то должен установить версию, загрузить все необходимые файлы с данными, возможно, обновить некоторые базы данных или модифицировать зависимые системные конфигурации. В случае установки клиент-серверных и других многоплатформенных систем инсталляция новой версии может означать ее установку на нескольких машинах. Иногда имеет значение порядок установки. Группа тестирования должна убедиться, что тестовая версия готова к тестированию. Если это не так, группа тестирования должна подготовить ее, например, установив заплатку на продукт на лету, исправив ранее необнаруженную проблему конфигурации тестовой среды и т.п. Если мы не можем подготовить тестовую версию, то группа тестирования должна удалить ее из тестовой среды. Как только готовы правильно установленная тестовая версия и тестовая среда, группа тестирования обычно получает полномочия на проведение проверочного тестирования, цель которого удостовериться в том, что ошибки, о которых было заявлено, исправлены, действительно устранены. Пройдя через все эти этапы, группа тестирования возобновляет плановое тестирование. В некоторых случаях сначала мы хотим провести полное регрессионное тестирование (заново выполнить все ранее проведенные тесты или перетестировать все ранее проверенные условия). Если группа тестирования отвечает за выполнение нескольких проектов тестирования одновременно, поступление на тестирование сборки наиболее важного проекта может приостановить тестирование в других проектах, что требует существенного и рискованного переключения всех или части сотрудников группы тестирования на другой контекст.

Многие группы тестирования работают именно так. В этих обстоятельствах установка новой версии не является легким решением. Поскольку она прерывает и изменяет плановое тестирование, я и многие другие менеджеры, с которыми я общался, считают нежелательным появление тестовых версий в незапланированные сроки. Также из-за накладных расходов, связанных с установкой новой сборки, их частая архивация может серьезно повлиять на скорость работ по тестированию. Чем чаще поступают тестовые версии, которые полны ошибок, нестабильны или неинсталлируемы, тем более это замедляет процесс подготовки тестовых версий. (В таких случаях требуется проведение приемочного тестирования для тестовых версий до их передачи на тестирование.)

Однако, поскольку тестирование несвежей (т.е. устаревшей) тестовой версии может привести к формированию отчета об уже исправленных дефектах, существует также вероятность получения тестовых версий слиш-

ком редко. Если для выполнения всех тестов требуется много времени, вы, возможно, захотите принять очередную тестовую версию до завершения выполнения всех тестовых сценариев.

Вернемся к теме тестовых версий, раундов и циклов (см. главу 3). Поскольку передача тестовой версии является началом последующего тестового цикла, наличие ожидаемых и регулярных тестовых версий позволяет группе тестирования спланировать работу с версиями. Обратите внимание, как Джамал спланировал периоды проверочного, планового и исследовательского тестирования в рамках каждого тестового цикла (см. рис. 12.1). Регулярность действий позволяет ему сделать прогноз, сколько времени потребуется для проведения тестового раунда, поскольку он может добавить в план известный объем накладных расходов на сборку тестовых версий. Он может спланировать менее насущные, но не менее важные действия, такие как обновление тестовых сценариев и данных, настройка автоматизированных тестовых скриптов. Регулярное и плановое поступление тестовых версий позволяет тест-менеджеру спланировать заранее рабочую неделю и выделить время для различных работ.

И наоборот, проекты, в которых отсутствуют предсказуемость и регулярность передачи тестовых версий, часто платят большую цену за то, что кажется гибкостью. Эта цена включает в себя неэффективное и нерациональное тестирование, выполняемое спешащими людьми при помощи некачественных средств тестирования в ненадежной тестовой среде

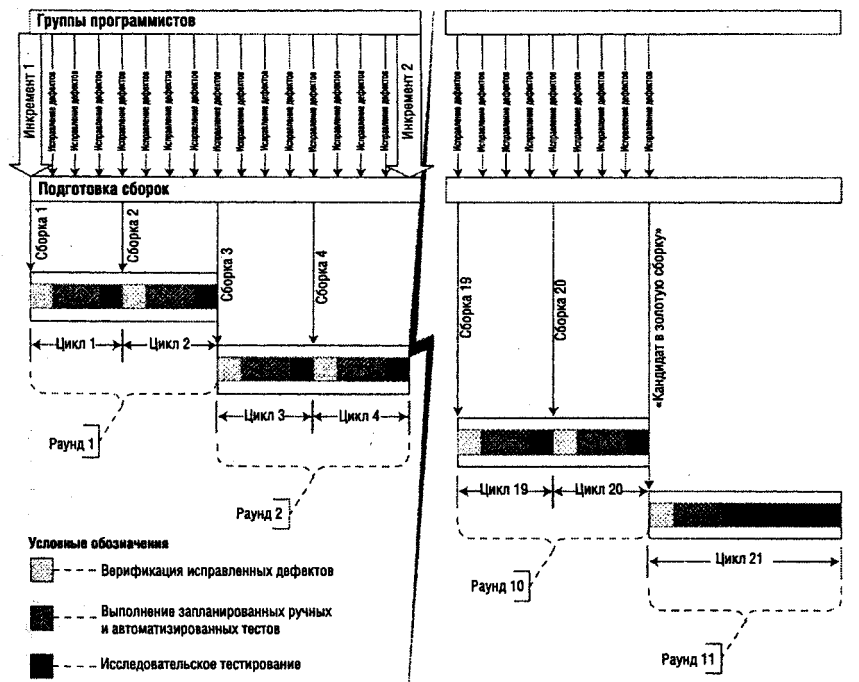


Рис. 12.1. Тестовые раунды, циклы и версии

с попытками учесть последние изменения в плане. Хаотический и непредсказуемый процесс подготовки тестовых версий разъедает все другие процессы, описанные в этой книге. В компаниях часто не понимают, что процессы, снижающие общую эффективность команды, не могут повысить гибкость, они уменьшают ее, забирая энергию, концентрацию и наиболее ценный проектный ресурс — время.

В одном из наиболее успешно управляемых проектов, в которых я участвовал, группа разработки приступала к заключительной стадии разработки согласованного содержимого версии в пятницу, и к субботнему вечеру код был готов. Менеджер сборки приходил в воскресенье утром, и мы получали тестовую версию в воскресенье днем. Ко времени завершения смены в воскресенье вечером у нас была установленная тестовая версия, прошедшая приемочное тестирование и готовая к тестированию в понедельник утром.

Когда различные проекты борются за единственное множество ресурсов тестирования, координация усилий становится более критической и более сложной. Только один хаотический проект, использующий непредсказуемые тестовые версии, может разрушить всю организацию тестирования. Один тест-менеджер вспоминал об этом, как "о наезде поезда, груженного плохим проектом".

Заметим, что в нашем примере Джамал делит свою группу и различные конфигурации тестовой среды на подгруппы и конкретные конфигурации, нацеленные на реализацию отдельных проектов, включая новую версию SpeedyWriter и тестирование версий сопровождения для предыдущих версий пакета OfficeArrow. Считаю такое деление на базе проектов полезным в том случае, когда есть проекты, которые могут внести в последний момент много изменений в расписание тестовых версий. Любое предложение о перераспределении ресурсов из одного проекта в другой является предметом обсуждения, к которому я могу привлечь конкурирующих за услуги группы тестирования менеджеров проектов. Называйте меня Макиавелли, но я считаю, что сохранить мой политический капитал позволяет предоставление права другим сказать мне "нет" всякий раз, когда это возможно. При наличии ограниченных ресурсов менеджер проекта, который настаивает на том, что ему нужна свобода в передаче тестовых версий на тестирование тогда и так часто, как он хочет, может создавать отрицательные последствия только для его проекта, а не для проектов других менеджеров.

Все это не означает, что у вас может быть единый, универсальный процесс сборки тестовых версий для всего, что поступает в тестовую лабораторию. Процесс, показанный на рис. 12.1, подходит для проекта разработки в рамках итерационного жизненного цикла, комплекта тестов, который может быть выполнен от начала до конца за пару недель, и для стабильного продукта, который не требует большого объема регрессионного тестирования для каждой тестовой версии. В проектах с каскадным жизненным циклом или с V-моделью инкременты не подойдут. Наилучший опыт тестирования версий сопровождения я получил, когда группе тестирования была передана первая версия с полностью реализованными свойствами, после чего в течение недели было проведено тестирование и исправление обнаруженных ошибок. Затем поступила вторая вер-

сия и прошла еще неделя тестирования, и наконец была выделена неделя тестирования версии для заказчика, после того как она прошла процесс сборки версий.

Иногда требуется тестировать аварийные патчи, которые содержат исправление одной-двух ошибок и должны быть разработаны в течение недели, а иногда и одного дня. Это очень рискованные предприятия, и ключом к ним является выбор разумной стратегии тестирования, которая уравновешивает все параллельные риски. Возможно, вы передаете аварийные патчи только тем заказчикам и пользователям, которые сталкиваются с проблемой, а затем включаете эти патчи в следующую версию сопровождения после тщательного регрессионного тестирования.

Как видите, различные обстоятельства, в том числе различные фазы системного жизненного цикла, могут оказывать значительное воздействие на процесс подготовки тестовых версий. Здесь может помочь специально разработанное множество стандартизованных процессов в соответствии с рекомендациями этой книги. Можно один раз придумать качественный процесс, а затем многократно использовать знания вместо того, чтобы делать это каждый раз заново.

Использование четко определенных процессов инсталляции и деинсталляции, похожих на те, что применяет пользователь

Ранее я упоминал о том, что версия должна быть как минимум инсталлируемой. Но качественный процесс подготовки тестовых версий включает в себя процедуру установки, которая четко определена. Кроме того, в ходе фазы системного тестирования и на последующих фазах тестирования я предпочитаю использовать процесс подготовки тестовых версий, который использует ту же процедуру инсталляции, что и заказчики и пользователи в процессе эксплуатации системы. (Процедура предполагает тестирование с теми же видами прав и учетных записей.) Я также предпочитаю наблюдать и проводить тестирование процесса деинсталляции тестовой версии, потому что заказчики или группа поддержки системы, вероятно, должны это делать и потому что может возникнуть необходимость откатить плохую тестовую версию.

Сложность процесса инсталляции часто является функцией системы, предназначенной для тестирования (см. ниже). Приложение Windows, как правило, имеет прямой процесс установки, тогда как сложные варианты конфигурации сред с многочисленными серверами, иногда распределенными по глобальной сети, часто требуют для установки больших затрат человеко- часов. Тем не менее во всех случаях процесс инсталляции должен быть как можно более простым. Простые процессы менее подвержены ошибкам и, следовательно, с большей вероятностью быстрее приводят к созданию тестопригодной конфигурации. Также простые процессы обычно требуют привлечения меньшего числа людей. Возможность неправильного взаимодействия, несогласия и возникновения политических вопросов возрастает с ростом числа людей и групп, вовлеченных в процесс, что увеличивает сложность, продолжительность и вероятность сбоев в любой деятельности.

Если процесс достаточно прост, группа тестирования может выполнить работу самостоятельно. Это позволяет группе тестирование проверить сами процессы инсталляции, деинсталляции, обновления и установки патчей. В случае использования архивированной программы часто необходимо провести тестирование инсталляции на разнообразных системных конфигурациях. Для IT- программ среда не может изменяться, но варианты, связанные с порядком исполняемых инсталляций или изменений баз данных, использования процессов установки патчей, обновления и первоначальной инсталляции и т.п., приводят к созданию разнообразных тестовых условий. Процесс инсталляции должен покрывать все эти тесты, таким образом избавляя группу тестирования от большой и ненужной работы по тестированию инсталляции, которая должна проводиться в ходе тестового цикла. Все-таки вам придется когда-то инсталлировать версии.

Процесс деинсталляции — это в лучшем случае вариант процесса инсталляции. Для приложений Windows, например, панель управления обеспечивает доступ к обеим функциям, а пакеты в формате InstallShield поддерживают удаление файлов и изменений реестра в качестве пунктов меню. Для Solaris аналогичные функции предоставляют утилиты управляющего пакета *pkgadd* и *pkgrm*. В некоторых случаях применение утилиты построения образов диска, например Ghost, дает возможность группе тестирования сохранить чистые образы систем, в которые они могут устанавливать любые версии. Однако в некоторых очень сложных средах откат изменений может быть гораздо более сложным, чем установка, если не существует инструментов управления конфигурацией, которые могут надежно откатывать любую последовательность изменений системы. Тем не менее я однозначно выступаю за средства деинсталляции. Я сталкивался с проектами, в которых не было такой возможности, и в результате версии с ошибками разрушали тестовую среду, что приводило, как минимум, к неделе простоя, пока шло восстановление всей тестовой среды, включая операционную систему, сопутствующие программы и предыдущую версию системы, предназначенную для тестирования. Я слышал аналогичные истории и от других тестировщиков, так что нельзя назвать это моим личным невезением.

Довольно давно в проекте сетевой операционной системы UNIX я наблюдал, как хорошо выполнялся процесс инсталляции и деинсталляции. В качестве аппаратной среды использовались разнородная сеть мэйнфрейм/PS-2 с инфраструктурой Ethernet и кольцо с маркерным доступом (token ring). Интегрированная программа, запускавшаяся с одного серве-

Номер сборки, имя сборки, версия, номер версии (Build ID, Release Name, Revision, Revision Number, Version, Version Number)

Уникальная последовательность цифр и/или букв, однозначно определяющая сборку.

ра, могла 1) установить свежую версию операционной системы на форматированные жесткие диски на одном и более серверах или рабочих станциях; 2) обновить существующие серверы или рабочие станции новыми версиями или патчами; 3) откатить обновления и патчи. В ходе системного тестирования мы варьировали процесс каждую неделю, чтобы охватить все возможные комбинации, сначала устанавливая версию для тестирования на чистое оборудование, а по завершении фазы системного тестирования проводя систематическое тестирование средств инсталляции и деинсталляции в операционной системе.

Присвоение имен тестовым версиям

Тестовая версия, поступающая из большой системы управления конфигурациями, как правило, имеет идентификатор или номер версии. В идеале тестировщику должна быть доступна простая команда, позволяющая получить номер версии из самой системы, предназначенной для тестирования. Например, в большинстве приложений Windows выбор меню Помощь/О программе открывает экран, показывающий номер версии.

Это важно, потому что, за возможным исключением приемочных тестов, тестировщики выполняют тесты на многих версиях и находят ошибки в каждой из версий. Они вводят ошибки в систему управления дефектами (см. главу 14). В отчете о дефекте тестировщик должен указать версию, содержащую ошибку. Без этой информации разработчикам трудно локализовать корневую причину среди многих изменений в коде, особенно если симптом отказа некоторым образом изменяется от одной версии к другой. Кроме того, менеджеры, вероятно, будут иметь проблемы при получении измерений числа ошибок, найденных в версии.

По этим причинам для любого продукта, передаваемого на тестирование, необходимо соглашение об именовании версий. Соглашение об именовании не обязательно должно иметь смысловую нагрузку. Можно, вероятно, назвать версии Фред, Зедедиа, Эмамалини. Однако я предпочитаю видеть упорядоченность и простоту в способах именования версий. Чтобы начать последовательность, мы можем использовать такие шаблоны, как A, B, C и т.д. или 0001, 0002, 0003 и т.д. Первый подход довольно слаб, поскольку перечень букв может иссякнуть либо по достижении Z превратиться в AA, AB, AC и т.д. Я наблюдал, как в одном успешном подходе к управлению версиями мой клиент использовал для обозначения версий четыре цифры. Тестовые версии появлялись каждый вечер, подвергались приемочному тестированию и использовались разработчиками в качестве основы для дальнейшей работы. Один раз в неделю менеджер сборок передавал проверенную ночью версию группе тестирования. Таким образом, мы получали версии с номерами, отличавшимися на 7 единиц, например, 0103 для одного тестового цикла, 0110 — для второго, 0117 — для третьего и т.д. Иногда номера версий отличались более чем на 7 единиц, поскольку разработчики повреждали сборку и в этот день появлялось больше одной версии.

Большое разнообразие последовательностей имен является работоспособным, если любое изменение добавляет единичку к номеру версии. Всякий раз, когда состав библиотеки исходного кода изменяется, а номер

версии нет, две тестовые версии с одним и тем же номером не являются одинаковыми и могут вести себя по-разному. Еще хуже, когда исходный код в репозитории не меняется, а тестовая версия меняется, поскольку кто-то не положил измененный модуль в библиотеку. Это путь к хаосу в отчетах о дефектах и к невоспроизводимым ошибкам, что, безусловно, означает неэффективное и нерациональное проведение тестирования.

Заметим, что тестировщик не должен знать с точностью до отдельного файла, что означают номера версий множества компонентов. Важно, чтобы разработчики могли разобраться в этом, обычно при помощи системы управления исходным кодом. Я отдаю предпочтение единственному номеру версии. Инструмент, работающий с репозиторием, может сопоставить этот номер номерам версий отдельных компонентов и, если необходимо, конкретным изменениям исходных файлов.

Помимо схем логического именования, тестировщик должен уметь устанавливать номер тестовой версии при помощи самой версии. Я предпочитаю методы, позволяющие делать запрос к системе. В качестве примера вновь можно привести стандарт Windows для экранов Помощь/О программе. Если совокупность символов не извлекается из программы, тестировщик может найти наименование при помощи утилиты *strings*. При любом подходе, конечно, процесс подготовки тестовых версий должен включать в себя автоматизированный или ручной способ изменения этого именования при переходе от одной версии к другой.

В некоторых случаях можно запросить не саму систему, а систему, которая ее обновляет. Рассмотрим следующий пример. Однажды я тестировал клиент-серверную систему, в которой программы, выполнявшиеся на клиентах, поступали с одного сервера. Этот сервер имел базу данных, которая содержала идентификатор клиентского устройства и номер последней тестовой версии, установленной на устройстве. Запрашивая сервер, тестировщики знали, какая версия программы выполнялась на каждом из данных клиентов.

Возвращаясь на минуту к теме инсталляции и деинсталляции, отмечу, что эта система также обеспечивала простой интерфейс в виде командной строки, позволявший изменять версию, которая направлялась на клиентские машины при их очередном соединении с сервером, как на предыдущую (деинсталляция), так и на последующую (инсталляция). Вот отличный, высококачественный и непротиворечивый интерфейс для инсталляции, деинсталляции и запросов. Данный пример показывает, что при некоторой предусмотрительности простая, но мощная утилита может решить много различных проблем, связанных с управлением версиями, даже в сложной среде.

Подготовка сборки из тщательно отобранных, скоординированных и документированных компонентов

Для иллюстрации проблем, связанных с управлением содержимым тестовой версии, рассмотрим следующую гипотетическую ситуацию. Цикл системного тестирования начинается с тестирования данной версии. В течение недели группа тестирования находит и информирует о 30 проблемах в версии, каждая в отдельном отчете о дефекте. Кроме того, представите-

ли группы продаж, маркетинга или бизнеса поместили шесть запросов на усовершенствование, т.е. на новые свойства для настройки продукта для определенных сегментов заказчиков и пользователей. Вместе эти документы дают возможность группе проекта улучшить качество продукта. Чтобы обеспечить производительность и эффективность, проектная группа должна тщательно управлять процессом. Повышение качества продукта на основе 36 запросов нуждается в процессе, который последовательно рассматривает три вопроса.

1. Как группа управления проектом решает, какие из 36 потенциальных изменений являются кандидатами на включение в очередную тестовую версию?
2. Как группа разработки уведомляет группу тестирования о том, какие изменения в поведении системы, по их мнению, будут реализованы посредством изменения кода?
3. Как группа тестирования информирует группу разработки о состоянии успешного или неудачного изменения кода: изменение реализовано без связанной регрессии, изменение реализовано, но со связанной регрессией, изменение не реализовано верно?

На рис. 12.2 показан процесс управления этими проблемами. В этом процессе отчеты о дефектах и запросы на усовершенствование рассматриваются на совещании с участием специалистов разных направлений. На этом совещании группа управления проектом, включая разработчиков, тестировщиков, специалистов по продажам и маркетингу, спонсоров, специалистов по технической поддержке или поддержке пользователей и, возможно, операций, оценивает затраты и преимущества каждого предлагаемого исправления ошибки и потенциального усовершенствования. Эту группу специалистов разного профиля обычно называют группой по управлению изменениями. (Процесс управления изменениями рассматривается в главе 16.)

Выбор содержимого версии оказывает влияние как на систему, предназначенную для тестирования, так и на процесс тестирования в данном цикле. Например, некоторые пропущенные свойства или исправления дефектов могут иметь особое значение, поскольку они блокируют ключевые тесты, как отмечал Джамал в рассматриваемом нами примере. Эти тесты в случае задержки их выполнения до конца процесса тестирования могут привести к обнаружению в последний момент критических ошибок. Кроме того, в некоторых моделях жизненного цикла разработки, например в эволюционной модели, используемой в нашем примере, проектная группа собирает состав версий таким образом, чтобы в конце каждого тестового цикла она могла получить систему, которую можно поставить или развернуть.

В конце совещания достигается предварительное соглашение о том, какие изменения будут и не должны быть реализованы. Я использую слово "предварительное" по двум причинам. Во-первых, поскольку старшие и топ- менеджеры, вообще говоря, считаются с мнением группы управления проектом по поводу управления дефектами и процесса выбора усовершенствований, иногда у них также бывают любимые свойства или любимые жалобы, которые, по политическим причинам за отсутствием других, проектной группе приходится учитывать. Во-вторых, группа раз-

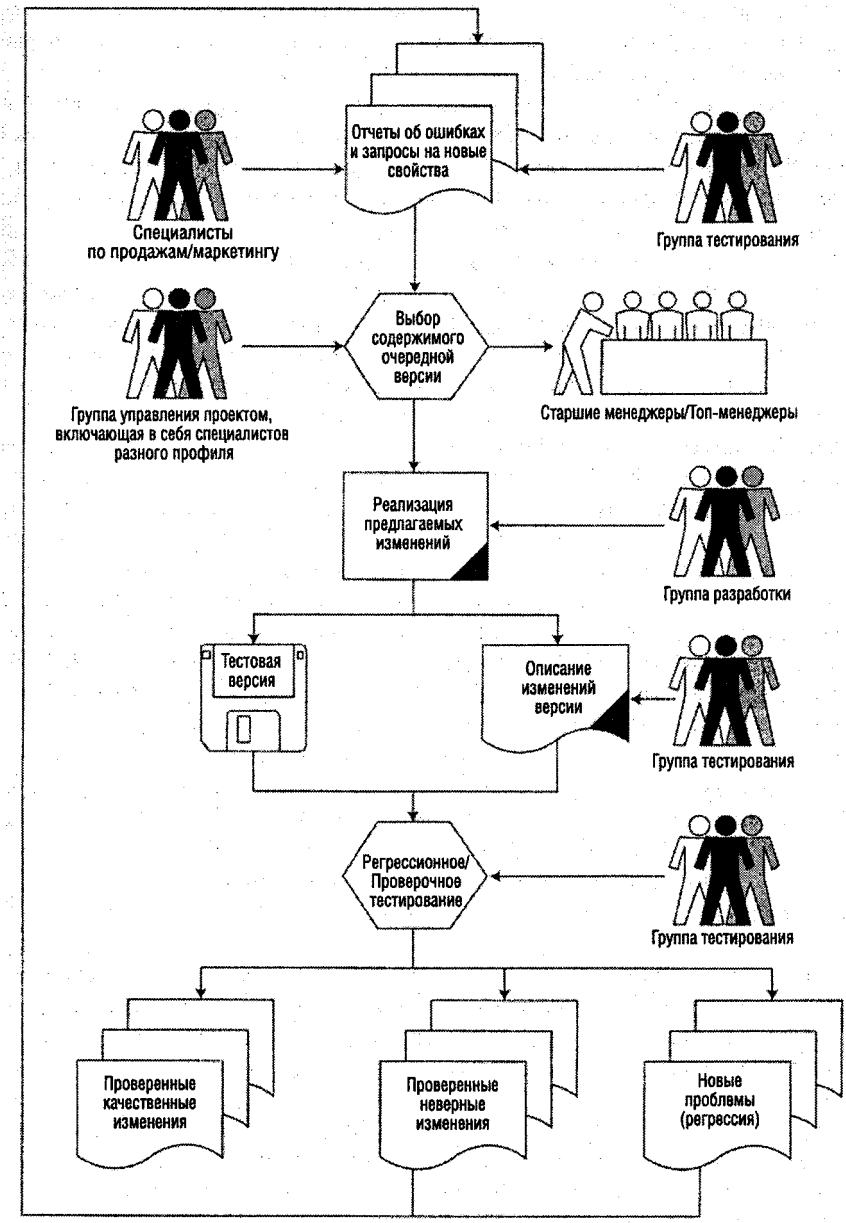


Рис. 12.2. Процесс управления изменениями и тестовыми версиями

работки не может обычно завершить реализацию всех изменений вовремя к выпуску следующей версии. Некоторые изменения требуют больше времени, чем другие. Некоторые оказываются при ближайшем рассмотрении более сложными, чем считалось сначала.

Группа разработки переходит к реализации предложенных изменений и передаче тестовой версии. Кроме того, группа тестирования должна получить информацию об изменениях, содержащихся в версии. Информация об изменениях показывает, какие из предложенных изменений реализовала группа разработки. Я встречал разные способы неплохой работы по подготовке информации об изменениях. В тех случаях, когда проектная группа работала с надежным инструментом управления дефектами и усовершенствованиями, информация об изменениях в версиях предоставлялась при помощи этого инструмента. В некоторых проектах по разработке приложений для UNIX и web-приложений группа проекта передавала информацию об изменениях в виде текстового файла как часть пакета, содержащего тестовую версию. Эти документы, часто называемые файлами *readme*, могут содержать ссылки на идентификаторы дефектов или просто перечислять изменения. Автоматизация и возможность отслеживания упрощают сбор и подготовку информации о версии, но отсутствие таких инструментов не устраняет необходимости в информации об изменениях в версии. Я встречал подготовленные без всякой автоматизации, вручную файлы с информацией об изменениях в версии, которая была критически важна для группы тестирования.

Завершение цикла изменений – это последний этап итерационного процесса. При использовании автоматизированной системы управления дефектами и усовершенствованиями это обычно означает закрытие отчетов о дефектах, для которых успешно завершилась проверка их устранения, повторное открытие и добавление комментариев в отчеты об ошибках, которые оказались неисправленными, и подготовку новых отчетов о дефектах, обнаруженных при регрессионном тестировании, или об ошибках в новых функциях. (Более подробно о проверочном и регрессионном тестировании и подготовке отчетов о дефектах рассказывается в главах 13 и 14.) Отчеты о повторно открытых ошибках и об ошибках регрессии, а также о новых ошибках, найденных в период планового тестирования, передаются на следующий раунд управления содержимым версий, исправления дефектов и документирования (см. рис. 12.2).

До настоящего времени могло казаться, что в центре обсуждение стоит исходный код системы. Тем не менее я говорю обо всей системе, обо всех ее элементах. В ситуациях, когда данные и/или метаданные оказывают существенное влияние на работу системы, проектная группа должна управлять этими элементами и документировать их так же тщательно, как и исполняемое содержимое. Например, как-то я работал в проекте, в котором одна из групп разработки утверждала, что ее компонент системы содержит все свойства, но они перенесли изменения схемы базы данных для мастер-таблицы, которая содержала информацию о пользователях, из одной версии в следующую. Это вызвало столько же проблем при тестировании, как и добавление или удаление свойств, поскольку каждое поле было связано с изменениями пользовательского интерфейса и модификациями сценариев использования системы и транзакций, которые оказывали влияние на агентов центра обработки вызовов.

Решение проблем

Успешный процесс управления тестовыми версиями базируется на прочном фундаменте проверенных в течение десятилетий лучших способов организации работ по созданию систем. Тем не менее усложнение технологий и систем может действительно породить новые проблемы. Кроме того, возникают вопросы, связанные с управлением. Посмотрим на некоторые из этих проблем.

Кто отвечает за процессы подготовки тестовых версий?

В таблице 12.1 представлены этапы процесса подготовки тестовых версий, описанного выше, и ответственные за каждый этап. В некоторых компаниях этапы 3 — 6 живут где-то на необитаемом острове. Мне известны проекты, в которых ключевые участники, как мне кажется, полагали, что программы волшебным образом переходят от группы разработки к группе тестирования без каких-либо затрат. Поскольку группа тестирования ожидает передачи программ, чтобы начать тестирование, часто эти этапы выполняются самой группой тестирования, которая не имеет ни сотрудников нужной квалификации, ни свободного времени на осуществление этого процесса без оказания влияния на эффективность и производительность группы.

Таблица 12.1. Задачи процесса подготовки тестовых версий и ответственные за их выполнение

Этап	Задачи	Ответственный (Как правило/Предлагается)
1.	Выбор содержимого новой версии	Группа управления проектом, включающая в себя специалистов разных направлений
2.	Реализация изменений	Группа разработки
3.	Сборка тестовой версии	Неясно/Инженер сборок
4.	Приемочное тестирование тестовой версии в среде управления конфигурацией/сборкой	Неясно/Инженер сборок
5.	Упаковка и передача тестовой версии	Неясно/Инженер сборок
6.	Установка тестовой версии в тестовой лаборатории	Неясно/Системные администраторы (сложные системы) или группа тестирования (простые системы)
7.	Приемочное тестирование тестовой версии в тестовой лаборатории	Группа тестирования

Когда процесс подготовки тестовых версий проходил гладко, за выполнение этапов 3 — 6 отвечали сотрудники в соответствии с предложениями, представленными в таблице 12.1[1]. Создание сборки, ее первое приемочное тестирование и передача тестовой версии находятся в сфере ответственности специально выделенного сотрудника рабочей группы проекта. Его называют по-разному, я буду использовать термин "*инженер сборок*" (release engineer). Кому он непосредственно подчиняется? Не ду-

маю, что это имеет значение. Я наблюдал, как группа тестирования подчинялась группе обслуживания разработки, которая также отвечала за управление версиями (и пользовательскую документацию). Были случаи, когда инженер сборок являлся самостоятельной единицей группы разработки. Оба подхода вполне работоспособны.

Если не разрешается иметь в проектной группе специального инженера сборок, выполнять такие обязанности в проектной группе должна группа разработки. Во всяком случае, при передаче каждой тестовой версии менеджер разработки предлагает группе тестирования принять работу его группы, поскольку она готова к тестированию. Разве не согласится менеджер разработки отвечать за задачи, связанные с передачей *тестовой версии*, если она готова к тестированию.

Сложность системы, предназначенной для тестирования, и другие вопросы

Как отмечалось выше, группе тестирования следует проверить процессы инсталляции, деинсталляции, установки обновлений и патчей и т.п. Помимо этого, предметом анализа является сложность системы. Если вы тестируете приложение для ПК, его установка, скорее всего, является тривиальным процессом. Даже многие варианты ОС UNIX сейчас содержат дружественные, графические утилиты инсталляции.

Предположим, однако, что мы начали добавлять дополнительные объекты, например базу данных, которая может быть как локальной, так и сетевой. Что вы думаете по поводу трехзвенной архитектуры и о приложениях на основе браузера, которые общаются с web-сервером, сервером приложений и сервером баз данных? Поскольку определение "системы, предназначенной для тестирования", может быть расширено таким образом, что оно начинает включать в себя разнообразные приложения, иногда выполняемые на различных аппаратных платформах и, возможно, взаимодействующие в различных сетевых архитектурах, сложность, присущая "установке системы", будет увеличена от некоторого вполне охватываемого действия, выполняемого одним младшим тестировщиком, до задачи, требующей участия группы специалистов. Такая группа может быть очень специализированной и недешевой, чтобы тратить большую часть своего времени на тестирование; эти специалисты, по всей вероятности, работают в какой-то другой группе, например в группе сетевых операций или системного администрирования.

Если учитывать всю совокупность проблем, которые встают перед группой тестирования, такая сложность делает понятие состава "тестовой версии" очень размытым, аморфным. Я работал в одном проекте, в котором тестовые версии содержали особые совокупности из одного или (обычно) многих приложений, которые были частями девяти отдельных подсистем. Над каждой из девяти подсистем трудилась отдельная группа разработки,

---

[1] Один из рецензентов, Маркус Манляйтнер (Markus Manleitner), отметил, что в случае сложного продукта приемочное тестирование версии в среде управления конфигурацией и сборками выполняется специальной комплексной группой тестирования. Он сообщил о неплохих результатах при этом подходе.

Менеджер сборки, инженер сборки (Release Engineer, Build Manager, Source Code Control Engineer, Release Manager)

Специалист, отвечающий за создание тестовой версии из исходного кода, данных, метаданных, файлов помощи, web-экранов и прочих объектов, необходимых для сборки системы.

и каждая подсистема имела номер версии, состоящий из пяти частей, например 101.252.17.19.2003. Так что нужно было взять последовательность из 45 чисел, чтобы найти определенную тестовую версию.

Более того, в этом проекте не существовало способа, при помощи которого подсистема могла бы запросить номер другой подсистемы или передать свой номер другой подсистеме, если та его запрашивала. Это существенная проблема, поскольку подсистемы тесно связаны и по данным, и по потокам управления, проходящим через подсистемы и представленным во многих тестовых сценариях. Тестировщик, вошедший в одну из подсистем, мог вовлекать в тестирование 3-4 другие подсистемы, но в лучшем случае он мог запросить номер версии только у базовой подсистемы. В большинстве случаев подсистемы состояли из всевозможных приложений, утилит, схем баз данных и других меняющихся компонентов. Однако приложение, которое видел тестировщик, не могло получить информацию о версиях других компонентов.

Да, скажете вы, совершенно понятно, что иногда по-настоящему большие, мультисистемные тестовые версии, подготовленные при участии нескольких групп, трудно поддаются управлению. Более мелкие и простые проекты не приводят к такого рода проблемам.

Это не всегда верно. Я работал в проекте разработки Интернет-приложения, в котором у моего клиента было три различных программных компонента: клиентское приложение, которое обеспечивало функции браузера и почтового клиента, (встроенный) BIOS для Интернет-приложения, версия встроенного программного обеспечения для модема. У нас были устройства в тестовой лаборатории, причем когда мы их опрашивали, они показывали, что выполняют одни и те же программы, но это происходило потому, что мы не могли опросить модемы. Вышло так, что встроенные в модем программы менялись в процессе тестирования из-за внешних факторов, связанных с аппаратурой производителя модема. Эти изменения приводили к большому количеству проблем соединения в определенных версиях встроенных программ. Можете представить, насколько нас сбивали с толку попытки локализовать проблемы надежности в коммутируемых сетях в очевидно одинаковых конфигурациях системы, которые на самом деле не были одинаковыми.

Мне неизвестны технические решения, которые могут работать с несравнимыми, произвольно большими и сложными системами тестов. Однако проблемы, упомянутые выше, являются не только техническими. Это вопросы дисциплины управления изменениями. Почему производитель модема изменял встроенные программы, не уведомляя нас? Потому

что группа управления проектом не сообщила ему, что он должен это делать. Почему программисты, создавая различные подсистемы, не создали простые утилиты для опроса центральной базы данных, чтобы определять версию системы, всех ее 45 сегментов? Потому что руководство не сообщило им, что они должны это делать. В обоих случаях группа управления изменениями, функционирующая надлежащим образом, могла бы решить эти проблемы с привлечением простых, но надежных методов создания и запроса версий. Как и в случае с информацией об изменениях в версии, отсутствие готовых программных средств, позволяющих решать проблему автоматически, не отменяет ответственности рабочей группы проекта за решение проблемы.

Проблемы тестовой версии вытекают и из сложности тестовой среды. Повторю, важно знать, что мы тестируем, и это касается программных и аппаратных средств, в окружении которых работает система, предназначенная для тестирования. Однако в отличие от тестируемой системы тестовая среда либо меняется незначительно, либо вообще не меняется в ходе проекта тестирования. Если в тестируемой системе мы пытаемся добиться упорядоченных и управляемых изменений, то в тестовой среде мы хотим избежать изменений настолько, насколько это возможно. Единственным исключением, я думаю, являются необходимость изменения определения развернутой среды и необходимость тестирования различных конфигураций в случае, когда система будет работать в большом числе вариантов среды поставки или среды пользователей.

Доступность инструментов управления конфигурацией системы, предназначенной для тестирования, и конфигурацией тестовой среды существенно зависит от целевой платформы. Windows и UNIX, включая Linux, обладают довольно широким ассортиментом таких инструментов, в то время как другие, менее применяемые платформы, возможно, не имеют таких средств. Заметим также, что подходящий набор инструментов для этой работы зависит не только от целевой платформы, но и от предполагаемого процесса установки. В некоторый момент группе проекта, вероятно, потребуется протестировать процессы инсталляции, деинсталляции, установки обновлений и патчей. Некоторые простые методы управления конфигурацией системы тестов могут подорвать близость тестовой среды к реальной и сделать недействительными результаты тестирования.

Неполные и нереалистичные тестовые версии для системного тестирования и приемо-сдаточных испытаний

В ходе ранних фаз тестирования (модульного, компонентного и комплексного тестирования) тестировщики (программисты, независимые тестировщики или кто-то другой) обычно получают тестовые версии, которые не являются полными, содержат заглушки, драйверы или еще какие-либо отличия от той версии, которая будет передана заказчику. Согласно эволюционной и итерационной методологиям каждая из версий, предоставляемая на системное тестирование, также не будет содержать все свойства версии, передаваемой заказчику. Даже в рамках жизненного цикла каскадной и V-модели существуют формальные и неформальные основания для изменения определения системы, передаваемой на сис-

темное тестирование. Формальными основаниями могут быть решения группы управления изменениями, неформальными основаниями могут быть просьбы или угрозы со стороны программистов или руководства отсрочить реализацию любимых свойств.

Однако, как отмечала Лин Цу в нашем примере, тестирование неполных и нереалистичных тестовых версий в ходе последних циклов тестирования неизбежно является рискованным. Такая ситуация создает вероятность пропуска проблем с регрессией и скрытых дефектов.

Кроме того, что такое исправление дефекта, как не небольшое изменение в определении системы? Следовательно, если мы исправляем ошибки, мы меняем систему точно так же, как если бы мы добавили новые свойства. До определенной степени изменение кода может стать объектом риска независимо от того, добавляем ли мы новые свойства или просто исправляем ошибки. Это вторжение в код порождает риск постольку, поскольку такова природа изменения[1].

Другая проблема — отсутствие реализма. Отсутствие реализма может иметь следующее проявление. Передается и инсталлируется тестовая версия, которая подрывает наши возможности протестировать процессы лицензирования, выпуска, передачи и установки так, как они происходят в реальном мире. Сама система также может оказаться не совсем реальной. Например, мы применяем средство покрытия кода для инструментирования исходного кода, а затем создаем и выполняем системные тестовые сценарии на инструментированной системе. Такое тестирование может предоставить полезную информацию, касающуюся структурного тестирования, но какова вероятность того, что инструментирование оказало влияние на результаты тестирования? Это может произойти, если мы используем инструментированные системы тестов для тестирования мощности, объема, перегрузок и производительности, поскольку инструментирование увеличивает размер и уменьшает производительность системы, за исключением некоторого небольшого процента случаев, когда мы используем хорошие средства инструментирования.

Основной прием решения этих проблем — поиск компромисса между преимуществами и риском. В ходе начальных фаз тестирования и даже во время системного тестирования вы не ожидаете, что группа проекта прекратит внесение изменений в код, поскольку при помощи изменений добавляются свойства и улучшается качество. На рынке с высокой конкуренцией удовлетворение спроса на новейшие свойства и технологии может стать стратегической целью, и будет нелогично, если наше желание иметь надежную тестовую среду помешает целям бизнеса. Однако иногда риск изменений угрожает другой цели бизнеса: передаче или поставке качественной системы.

---

[1] Некоторые используют процент нового и измененного кода в репозитории кода в качестве метрики для завершения системного тестирования. Эта метрика может дать вам представление о вероятности регрессии или новых дефектов. Поскольку хочется, чтобы к концу проекта ежедневное число дефектов стремилось к нулю, соответственно есть желание, чтобы процент изменений кода также стремился к нулю. Кроме того, чем меньше изменяется код, тем более значимыми со временем становятся результаты тестирования. Если 10% кода меняется каждую неделю, о чем скажут результаты тестирования 5-6 недельной давности?

Прозорливый тест-менеджер может оказать помощь группе проекта в уравновешивании этих рисков. Рассмотрим пример. В проекте разработки Интернет-приложения, следовавшем жизненному циклу разработки V-модели, один из критериев начала системного тестирования гласил, что система должна иметь все запланированные свойства перед началом системного тестирования. На совещании, где принималось решение о начале системного тестирования, мы отказались от этого критерия в отношении двух конкретных свойств: печати и поддержки свойств аудио. Программисты все еще продолжали над ними работать, поскольку эти свойства считались важными для первой версии. Тем не менее имело смысл начать системное тестирование. С одной стороны, большая часть системы была завершена, и мы действительно получали на тестирования систему, хотя и не *финальную* ее версию. С другой стороны, существовала вероятность, что технические проблемы могут воспрепятствовать завершению работы над этими свойствами, поэтому задержка системного тестирования в ожидании реализации этих свойств могла бы поставить под угрозу дату поставки системы, причем указанные свойства могли все равно не войти в первоначальную поставку. Мы были связаны только критерием завершения тестирования, который предполагал трехнедельное тестирование без изменений, за исключением исправления дефектов, имеющих незначительный риск.

До некоторой степени мы можем снизить риск появления нереалистичных версий на системном тестировании, используя стратегию регрессионного тестирования. Если мы повторяем тесты в соответствии со стратегией регрессионного тестирования (см. главу 13), мы можем допустить инструментированные тестовые версии и значительные изменения, пока не начнем заключительный полный прогон всех тестов. (Конечно, здесь существует риск, что мы обнаружим регрессию или скрытые проблемы при последнем прогоне, что может угрожать срокам, но это меньший риск, чем не обнаружить проблем вовсе.) Если вы совсем не повторяете тесты, то любой прогон тестов для инструментированной или устаревшей тестовой версии повышает риск обнаружения регрессии или скрытых ошибок в заключительной версии.

Внедрение усовершенствований

Процесс подготовки тестовых версий — это один из наиболее чувствительных к контексту процессов. Общий процесс, описанный выше, дает хорошие результаты во многих случаях, но не во всех. Особые технологии и инструменты, на которых базируется процесс, оказывают большое влияние на то, как он осуществляется. Они также определяют точки приложения усилий, которые позволяют получить возврат вложений в изменение процесса. Однако я обнаружил несколько общих правил, заслуживающих внимания.

1. Подумайте о преимуществах инсталлируемых средств, надежного автоматизированного модульного, компонентного и комплексного тестирования каждой тестовой версии и приемочного тестирования для эффективности проекта тестирования.

2. Рассмотрите увеличение эффективности проекта, которое может быть получено за счет наличия средств именования версий и их запроса. Неспособность понять, в какой версии есть ошибка, расстраивает не только разработчиков, но и тестировщиков, поскольку программисты отмечают ошибки как невоспроизводимые, когда проблема состоит в том, что никто не знает, что происходило в тестовой лаборатории во время тестирования. На уровне управления тестированием может произойти существенная потеря доверия группе тестирования, если разработчики не могут найти проблемы, о которых сообщают тестировщики. Позор, если причиной этого являются коммуникативные проблемы, когда группа тестирования не в состоянии правильно назвать объект тестирования.
3. Посмотрите на источники снижения эффективности в процессах выполнения тестов и подготовки отчетов о дефектах. Я видел много проектов, которые попусту теряли две и более недели в начале тестирования, поскольку никто не понимал, как установить тестовую версию так, чтобы она работала. Я также сталкивался с проектами, которые превращались в бесконечную цепь ошибок, блокированного тестирования и несовместимых компонентов, сталкивающихся в тестовой лаборатории.

С передачей хороших тестовых версий не только улучшается процесс тестирования, но и вы становитесь более производительными. Я отстаиваю идею простых процессов именования и запроса номеров версий, аргументируя это улучшением предоставления услуг группе разработки. Иногда я говорю: "Профессиональная группа тестирования должна писать хорошие отчеты об ошибках; мы хотим точно указать, в каких версиях есть конкретные ошибки. Чтобы это можно было сделать достаточно надежно, мы обращаемся к вам за помощью в разработке качественных процессов именования и запроса имен версий". Что касается проверочного тестирования, я обычно говорю, что мы можем лучше и быстрее проверить, можно ли закрывать ошибки, если мы точно знаем, в какой версии они возникли.

Хорошие тестовые версии указывают на зрелость процесса и позволяют сэкономить время группы тестирования, что идет на пользу проекту. Во всех проектах, в которых я участвовал, а их было около десятка, и которые, по моему мнению, действительно отменно управлялись, был хорошо поставлен процесс подготовки тестовых версий. Три проекта, которые я вспоминаю как наиболее проблемные, испытывали трудности именно с управлением процессом подготовки тестовых версий. Невозможно успешно проводить тестирование без качественной подготовки тестовых версий, а невозможность успешного проведения тестирования обычно рассматривается как ключевой фактор неудачи проектов.

Помимо эффективности и успеха проекта, другим фактором, способствующим качественной подготовке тестовых версий, является неотвратимость передачи результатов проекта. Рано или поздно проектная группа должна передать инсталлируемую и стабильную версию пользователям или заказчикам. Рано или поздно проектная группа должна подготовить работоспособные процессы инсталляции, деинсталляции, установки обновлений и патчей для версий заказчиков и пользователей. Рано или позд-

но система получит номер версии, а заказчики и пользователи должны будут иметь возможность сообщать номер версии, если они обнаружат проблемы. Рано или поздно нужно будет согласовать все компоненты системы и установить все взаимодействующие компоненты в рабочую среду. Поскольку рано или поздно должны быть решены все вопросы, рассмотренные в этой главе, почему не начать получать эти решения и проверять их в ходе тестирования?

Поскольку изменения неизбежны, их появление — это только вопрос времени. Если кто-то настаивает на том, чтобы отложить определение и формирование качественных процессов подготовки версии до конца проекта, воспользуйтесь вашими контактами с группами обслуживания пользователей, технической поддержки, системы сетевой поддержки и с бизнес-аналитиками из проектной группы, чтобы они помогли убедить руководство в необходимости изменений. Могут также помочь и сотрудники отделов продаж и маркетинга. Если до людей, которые тесно общаются с заказчиком и знают пользователей, удастся донести значение процессов подготовки версий, то их можно убедить поддержать ваши призывы следовать процессу подготовки тестовых версий, который раньше вскроет потенциальные проблемы.

Как только мне удается убедить людей в необходимости изменений, обычно я даю возможность разработчикам и инженерам сборки руководить определением деталей процесса подготовки версий. Меня не сильно волнуют все эти подробности, поскольку сам процесс отвечает критериям и решает проблемы, описанные в этой главе. Я рад работать с людьми и помогать им в формировании качественного процесса. Суть процесса все-таки в передаче результатов. В эстафете бегуны должны согласовывать момент передачи эстафетной палочки, поскольку неудачная передача означает ее падение и поражение в соревнованиях. Придерживайтесь этого сравнения при передаче результатов, и вы сможете добиться многого, когда наступит время внедрять изменения в процессы подготовки тестовых версий.

ГЛАВА 13

Оценка качества: выполнение тестовых сценариев

Наконец, пришло время тестирования. В ходе выполнения тестов тестировщики используют ту систему тестов, которую мы подготовили для оценки качества тестовых версий, по мере того, как мы их получаем. Это, с одной стороны, волнующий, а с другой стороны, страшащий момент проекта, поскольку качество системы неизвестно, пока тестировщики не начали работать с ней. Более того, группа тестирования обычно имеет здесь уникальную возможность поиска способов заставить систему, предназначенную для тестирования, выйти, насколько это возможно, *из-под контроля* и *за рамки ожидаемого поведения*.

Кроме того, хотя от всех руководителей ожидают управления непредсказуемыми событиями в рамках их сфер ответственности, тест-менеджер — это единственный руководитель в компании, чьей сферой ответственности является поиск непредсказуемых событий. Этот поиск принимает форму выполнения комплекта тестов. Каким образом группа тестирования выбирает тесты, выполняет их в установленной системе и собирает информацию о системе, предназначенной для тестирования, о тестовых сценариях и о самом процессе выполнения тестов?

Процесс выполнения тестов

Начнем с общего процесса выполнения тестов — Процесса 9. Этот процесс пересекается с другими процессами. Этап 3.D, подготовка отчетов о проблемах качества тестируемой системы, следует процессу, описанному в главе 14. Этап 3.E, устранение проблем в системе тестов, следует процессу, представленному в главах 10 и 11. Наконец, этап 7, отчет о результатах тестового цикла и ходе тестирования, подчиняется процессу, обсуждаемому в главе 15[1].

---

[1] Я благодарен Тиму Коомену (Tim Koomen) за то, что он напомнил мне о некоторых элементах управления конфигурацией тестирования в этом процессе.

Процесс 9. Процесс выполнения тестов

№ этапа	Этап	Выполнено?
1.	На основе приоритетов рисков, ограничений проекта и других соображений выбрать комплекты тестов (из всего множества тестов), которые должны быть выполнены в данном тестовом цикле.	☐
2.	Назначить тестировщиков, ответственных за выполнение тестовых сценариев из каждого комплекта тестов.	☐
3.	Выполнить тестовые сценарии, подготовить отчеты о дефектах и непрерывно собирать информацию о тестах, принимая во внимание при выполнении очередного теста предыдущие результаты тестирования.	☐
3.A	Привести систему, предназначенную для тестирования, и систему тестов в соответствующие исходные состояния. Если данные исходные состояния понадобятся для выполнения различных тестов или нескольких итераций данного теста, сохранить исходные состояния для последующего применения.	☐
3.B	При помощи ввода данных и других действий заставить систему, предназначенную для тестирования, реагировать на желаемые тестовые условия.	☐
3.C	Наблюдать и оценивать выходные данные, поведение системы и ее состояния. Исследовать любые отклонения от ожидаемых результатов.	☐
3.D	Если необходимо, подготовить отчеты о проблемах в системе, предназначенной для тестирования.	☐
3.E	Если необходимо, подготовить отчеты о проблемах в системе тестов и/или исправить их.	☐
3.F	Собрать информацию и подготовить отчет о выполненных тестах.	☐
4.	Решать проблемы, препятствующие тестированию, при их возникновении.	☐
5.	Ежедневно готовить отчет о ходе тестирования, уточнять ответственных за выполнение тестов и пересматривать планы и приоритеты.	☐
6.	Если необходимо, удалить невыполнимые или устаревшие тесты в порядке, обратном их приоритетности (сначала удалять тесты низших приоритетов, в последнюю очередь – высших).	☐
7.	Периодически отчитываться о ходе и результатах тестового цикла.	☐
8.	Поместить все документы о ходе работ, начальные состояния, обновленные средства тестирования, другие элементы системы тестов и прочую информацию, получаемую на постоянной основе, в проектную библиотеку или систему управления конфигурациями. Вести контроль изменений всех элементов.	☐

Комплект тестов (Test Set)

Все наборы тестов, которые группа тестирования планирует выполнить в течение данной фазы (или уровня) тестирования.

Таблицы отслеживания тестов

В этой главе рассказывается о планировании и отслеживании выполнения тестов при помощи таблицы, которую я называю таблицей отслеживания тестов[1]. Основными частями этой таблицы являются сводная таблица тестовых сценариев и сводная таблица комплектов тестов.

На рис. 13.1 показана сводная таблица тестовых сценариев в том виде, в каком она появилась в начале раунда 5, после того как тест-менеджер Джамал Браун завершил отбор комплектов тестов и назначил тестировщиков, ответственных за выполнение каждого сценария. В столбце "Ответственный" указывается отвечающий за выполнение тестировщик. Столбец "Номер теста" содержит уникальный четырехзначный номер каждого теста. Столбец "Комплект тестов/Тестовый сценарий" содержит краткое наименование каждого комплекта тестов и каждого тестового сценария. Столбец "Плановая дата" содержит информацию о дате планового завершения теста согласно ожиданиям Джамала. Столбец "Плановые затраты" указывает число человеко-часов, необходимых, по оценке Джамала, на выполнение каждого тестового сценария.

По мере того как тестировщики будут выполнять каждый тест, они будут собирать информацию о тестах. Один из наиболее существенных элементов информации — состояние теста.

- *На очереди.* Тестовый сценарий готов к выполнению, назначен тестировщик, отвечающий за его выполнение в этом раунде.
- *В работе.* Тест выполняется в настоящее время.
- *Заблокирован.* Что-то мешает тестировщику выполнить тест.

	A	B	C	E	F	G	H	I	J	K	M	N	O	P
1			Sumatra Test Case Summary											
2			Integration and System Test Pass 5 (12/16-12/27)											
3														
4	Assigned To	Test ID	Test Suite/Case	Test Status	Test Environment	Bug ID	Bug RPN	Run By	Plan Date	Act Date	Plan Effort	Actual Effort	Test Duration	Comment
5														
7		1.000	*Functionality*											
8	DLK	1.001	File						12/16		2.0			
9	JW	1.002	Edit						12/16		0.5			
10	BEZ	1.003	Font						12/16		0.5			
11	DLK	1.004	Tables						12/17		3.0			
12	BEZ	1.005	Printing						12/16		2.0			
13	LJ	1.006	Managed Files						12/16		2.5			
14	LJ	1.007	Non-Managed Files						12/17		4.0			
15	BEZ	1.008	DMS Check-In New						12/17		2.5			
16	BEZ	1.009	DMS Check-In Existing						12/17		3.0			
17	BEZ	1.010	DMS Check-Out						12/18		3.5			
18	LTW	1.011	DMS Load/Migrate Online						12/16		3.5			
19	HS	1.012	DMS Load/Migrate NOL						12/16		3.0			
20	HS	1.013	DMS Load/Migrate Offline						12/17		3.5			
21	HS	1.014	DMS File Information						12/17		2.5			
22			Suite Summary						12/18	1/0	36.0	0.0	0.0	

Рис. 13.1. Таблица тестовых сценариев на момент начала раунда 5

[1] Подробное описание этого инструмента приведено в главе 5 моей предыдущей книги "*Managing the Testing Process, Second Edition*".

- *Пропущен.* По некоторой причине тестировщик решил не выполнять тест.
- *Прошел.* Тестировщик выполнил тест целиком и наблюдал только ожидаемые результаты, состояния и поведение.
- *Не прошел.* Тестировщик выполнил тест целиком и наблюдал один или более непредусмотренных результатов, состояний или поведение, серьезно подвергающее риску качество системы в отношении цели теста.
- *Предупреждение.* Тестировщик выполнил тест целиком и наблюдал один или более непредусмотренных результатов, состояний или поведение, однако качество системы в отношении цели теста не подвергается серьезному риску.
- *Закрыт.* После того как в предыдущем раунде тестового цикла тест был помечен как "Не прошел" или "Предупреждение", в следующей тестовой версии содержится исправление ошибки, которое реша

В ходе выполнения всех запланированных тестовых сценариев для тестового раунда тестировщики проходят по жизненному циклу тестов, показанному на рис. 13.2[1].

Помимо состояния тестового сценария, тестировщики собирают другую важную информацию. В столбце "Тестовая среда" (см. рис. 13.1) они указывают, какое аппаратное и программное обеспечение и другие элементы используются для выполнения теста. В столбцах "Номер дефекта" и "Приоритет дефекта" размещается информация об обнаруженных дефектах (см. главу 14). В столбце "Кем выполнен" каждый тестировщик

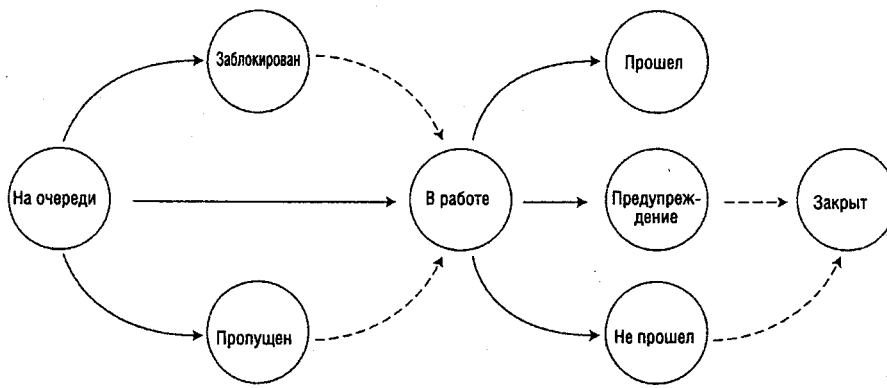


Рис. 13.2. Состояния и жизненный цикл тестовых сценариев

[1] Некоторые применяют более простые множества состояний тестов. Например, Джоанна Ротман рассказала мне, что она не применяет статусы "Заблокирован", "Предупреждение" и "Закрыт", а использует "Не прошел", "В работе" и "Прошел" соответственно.

Таблицы отслеживания тестов

В этой главе рассказывается о планировании и отслеживании выполнения тестов при помощи таблицы, которую я называю таблицей отслеживания тестов[1]. Основными частями этой таблицы являются сводная таблица тестовых сценариев и сводная таблица комплектов тестов.

На рис. 13.1 показана сводная таблица тестовых сценариев в том виде, в каком она появилась в начале раунда 5, после того как тест-менеджер Джамал Браун завершил отбор комплектов тестов и назначил тестировщиков, ответственных за выполнение каждого сценария. В столбце "Ответственный" указывается отвечающий за выполнение тестировщик. Столбец "Номер теста" содержит уникальный четырехзначный номер каждого теста. Столбец "Комплект тестов/Тестовый сценарий" содержит краткое наименование каждого комплекта тестов и каждого тестового сценария. Столбец "Плановая дата" содержит информацию о дате планового завершения теста согласно ожиданиям Джамала. Столбец "Плановые затраты" указывает число человеко-часов, необходимых, по оценке Джамала, на выполнение каждого тестового сценария.

По мере того как тестировщики будут выполнять каждый тест, они будут собирать информацию о тестах. Один из наиболее существенных элементов информации — состояние теста.

- *На очереди.* Тестовый сценарий готов к выполнению, назначен тестировщик, отвечающий за его выполнение в этом раунде.
- *В работе.* Тест выполняется в настоящее время.
- *Заблокирован.* Что-то мешает тестировщику выполнить тест.

	A	B	C	E	F	G	H	I	J	K	M	N	O	P
1			Sumatra Test Case Summary											
2			Integration and System Test Pass 5 (12/16-12/27)											
3														
4	Assigned	Test			Test	Bug	Bug	Run	Plan	Act	Plan	Actual	Test	
5	To	ID	Test Suite/Case	Status	Environment	ID	RPN	By	Date	Date	Effort	Effort	Duration	Comment
6														
7		1.000	*Functionality*											
8	DLK	1.001	File						12/16		2.0			
9	JW	1.002	Edit						12/16		0.5			
10	BEZ	1.003	Font						12/16		0.5			
11	DLK	1.004	Tables						12/17		3.0			
12	BEZ	1.005	Printing						12/16		2.0			
13	LJ	1.006	Managed Files						12/16		2.5			
14	LJ	1.007	Non-Managed Files						12/17		4.0			
15	BEZ	1.008	DMS Check-In New						12/17		2.5			
16	BEZ	1.009	DMS Check-In Existing						12/17		3.0			
17	BEZ	1.010	DMS Check-Out						12/18		3.5			
18	LTW	1.011	DMS Load/Migrate Online						12/16		3.5			
19	HS	1.012	DMS Load/Migrate NOL						12/16		3.0			
20	HS	1.013	DMS Load/Migrate Offline						12/17		3.5			
21	HS	1.014	DMS File Information						12/17		2.5			
22			**Suite Summary**						**12/18**	**1/0**	**36.0**	**0.0**	**0.0**	

Рис. 13.1. Таблица тестовых сценариев на момент начала раунда 5

[1] Подробное описание этого инструмента приведено в главе 5 моей предыдущей книги "*Managing the Testing Process, Second Edition*".

- *Пропущен.* По некоторой причине тестировщик решил не выполнять тест.
- *Прошел.* Тестировщик выполнил тест целиком и наблюдал только ожидаемые результаты, состояния и поведение.
- *Не прошел.* Тестировщик выполнил тест целиком и наблюдал один или более непредусмотренных результатов, состояний или поведение, серьезно подвергающее риску качество системы в отношении цели теста.
- *Предупреждение.* Тестировщик выполнил тест целиком и наблюдал один или более непредусмотренных результатов, состояний или поведение, однако качество системы в отношении цели теста не подвергается серьезному риску.
- *Закрыт.* После того как в предыдущем раунде тестового цикла тест был помечен как "Не прошел" или "Предупреждение", в следующей тестовой версии содержится исправление ошибки, которое реша

В ходе выполнения всех запланированных тестовых сценариев для тестового раунда тестировщики проходят по жизненному циклу тестов, показанному на рис. 13.2[1].

Помимо состояния тестового сценария, тестировщики собирают другую важную информацию. В столбце "Тестовая среда" (см. рис. 13.1) они указывают, какое аппаратное и программное обеспечение и другие элементы используются для выполнения теста. В столбцах "Номер дефекта" и "Приоритет дефекта" размещается информация об обнаруженных дефектах (см. главу 14). В столбце "Кем выполнен" каждый тестировщик

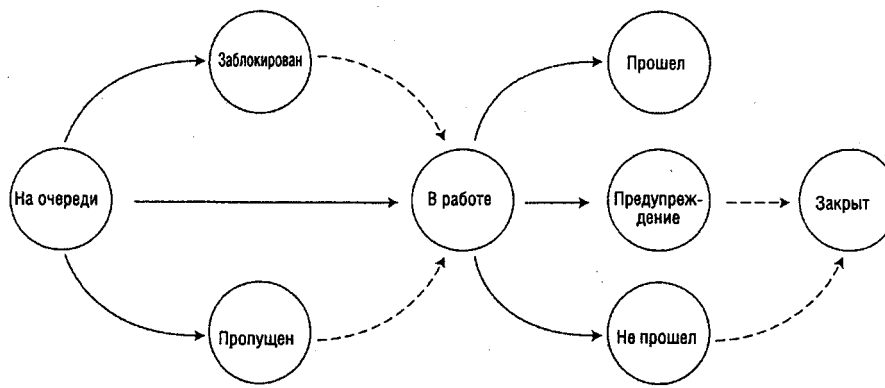


Рис. 13.2. Состояния и жизненный цикл тестовых сценариев

[1] Некоторые применяют более простые множества состояний тестов. Например, Джоанна Ротман рассказала мне, что она не применяет статусы "Заблокирован", "Предупреждение" и "Закрыт", а использует "Не прошел", "В работе" и "Прошел" соответственно.

вводит свои инициалы. "Дата выполнения" указывает, когда тестировщик выполнил тестовый сценарий, "Фактические затраты" показывают, сколько человеко-часов было потрачено тестировщиком, а "Продолжительность выполнения теста" показывает астрономическое время выполнения теста. (Продолжительность выполнения теста и затраты на выполнение могут отличаться для автоматизированных и ручных тестов, которые выполняются несколькими тестировщиками.)

Чтобы лучше понимать, что происходит со всеми комплектами тестов, и собирать полезную информацию для измерений (см. главу 15), я провожу анализ информации о состоянии теста и затратах на его выполнение в сводной таблице комплектов тестов, как показано на рис. 13.3. Это таблица содержит информацию о каждом комплекте тестов и по всему множеству тестов. К классу *выполненных* тестов я отношу те тесты, которые проведены или пропущены. (Для простоты тесты в состояниях "Предупреждение" и "Закрыт" я регистрирую в столбце "Прошел".) Я отношу к классу *невыполненных* те тесты, которые заблокированы или еще не завершены. Эта классификация позволяет мне проанализировать, в какой точке процесса проведения тестов я нахожусь.

Раздел "Выполненная работа" с правой стороны позволяет проанализировать, прошли ли мы нужное количество тестов в каждом комплекте, с учетом фактических затрат. Например, если мы прогнали 7 из 14 тестовых сценариев, мы выполнили 50% тестов (эта цифра появляется в столбце "Процент выполнения"). Предположим, что мы израсходовали 40 человеко-часов из 50 запланированных. В этом случае мы потратили 80% запланированных затрат (указывается в столбце "Процент затрат"). Это означает, что мы вряд ли завершим оставшиеся 7 тестов в запланированные сроки (считая, что все тестовые сценарии примерно одного размера, и не принимая в расчет вопросы приоритетности). Если это происходит со всеми комплектами тестов, то мы выйдем за рамки времени, отведенного на выполнение тестов, до того, как завершим все тесты, запланированные для этого тестового раунда. Поэтому выполненная стои-

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1						Sumatra Test Suite Summary									
2						Integration and System Test Pass 5 (12/16-12/27)									
3															
4		Total	Planned Tests Fulfilled				Weighted	Planned Tests Unfulfilled				Earned Value			
5	Suite	Cases	Count	Skip	Pass	Fail	Failure	Count	Queued	IP	Block	Pln Hrs	Act Hrs	% Effort	% Exec
6															
7	Functionality	14	0	0	0	0	0.00	14	14	0	0	36.00	0.00	0%	0%
8	Performance, Load, Capacity, and Volume	5	0	0	0	0	0.00	5	5	0	0	7.00	0.00	0%	0%
9	Reliability/Stability	2	2	2	0	0	0.00	0	0	0	0	0.00	0.00	0%	0%
10	Error Handling and Recovery	3	0	0	0	0	0.00	3	3	0	0	12.50	0.00	0%	0%
11	Installation, Maintenance, and Operations	4	0	0	0	0	0.00	4	4	0	0	72.00	0.00	0%	0%
12	Localization	8	0	0	0	0	0.00	8	8	0	0	128.00	0.00	0%	0%
13	Security and Privacy	4	0	0	0	0	0.00	4	4	0	0	17.00	0.00	0%	0%
14	Documentation	3	0	0	0	0	0.00	3	3	0	0	28.00	0.00	0%	0%
15	Integration	4	0	0	0	0	0.00	4	4	0	0	8.00	0.00	0%	0%
16	Usability Study	2	0	0	0	0	0.00	2	2	0	0	16.00	0.00	0%	0%
17	Exploratory	6	0	0	0	0	0.00	6	6	0	0	12.00	0.00	0%	0%
18															
19	Total	55	2	2	0	0	0.00	55	55	0	0	336.50	0.00	0%	0%
20	By Pct		4%	4%	0%	0%	N/A	96%	96%	0%	0%				

Рис. 13.3. Таблица комплектов тестов на момент начала раунда 5

мость позволяет тест-менеджеру или ведущему тестировщику вовремя скорректировать план тестового раунда.

Атака на большую сборку

В качестве первоначального действия каждую неделю группа тестирования "Суматры" выполняла проверочное тестирование исправлений дефектов в новых тестовых версиях согласно плану тестирования. В случае с большой сборкой оказалось много ошибок для верификации исправлений. Пока группа тестирования занималась проверочным тестированием, Джамал подробно планировал работу на предстоящую неделю.

Поскольку на тестирование был передан инкремент 3, это означало, что на этой неделе во внешних тестовых лабораториях должен начаться первый раунд тестирования локализации и удобства использования. Джамал еще раньше попросил Лин Цу координировать эти действия с тестовыми лабораториями и сделал пометку о необходимости напомнить ей об этом ближе к делу. Во вторник она связалась с лабораториями и запланировала передачу тестовой версии в обе лаборатории на вторую половину дня понедельника. В результате она получила запас времени для того, чтобы убедиться, что тестовая версия готова к началу тестирования.

Большая сборка означала большую работу. Это понимал Джамал, глядя на таблицу отслеживания тестовых сценариев. В ходе этого раунда он планировал пропустить только один комплект тестов – комплект для проверки надежности и стабильности. Это означало, что Эмма Мурхаус, одна из инженеров по автоматизации тестирования, будет выполнять комплект для тестирования производительности, нагрузки, мощности и объема, что требовало больших затрат.

Определенную проблему создавал и тот факт, что в середине тестового цикла были Рождественские каникулы. Однако Джамал пригласил в группу тестирования людей, относящихся к разным культурам, что позволяло избежать полной потери недельного периода. Он уже договорился о том, что инженер по ручному тестированию Лин Цу Ву и один из младших тестировщиков Джон Ву будут отдыхать в начале следующего года во время празднования китайского нового года по лунному календарю вместо рождества. Инженер по автоматизированному тестированию Динеш Кумар и младший тестировщик Эмамалини Шашидар получат дополнительную неделю к двухнедельному отпуску в апреле, после выпуска версии, чтобы посетить свои семьи в Индии и Бангладеш. Биньямин Цукерман поедет навестить семью в Нью-Йорк во время хануки. Таким образом, каждый тестировщик сможет хорошо провести свои религиозные и культурные праздники, а Джамал сможет распределить влияние этих праздников по нескольким раундам тестирования. Таким образом, на Рождественской неделе будут работать пятеро из семи его тестировщиков.

Джамал взглянул на распределение тестовых сценариев, которые должны быть выполнены в наступающем тестовом цикле на этой неделе. В первую очередь, с точки зрения рисков, нужно выполнить тесты, которые были ранее заблокированы, особенно функциональные. Некоторые

из сделанных изменений означают, что также важно выполнить тесты, проверяющие секретность, безопасность и обработку ошибок.

№ этапа	Этап	Выполнено?
1.	На основе приоритетов рисков, ограничений проекта и других соображений выбрать комплекты тестов (из всего множества тестов), которые должны быть выполнены в данном тестовом цикле.	☑

Джамал просмотрел каждый комплект тестов и, исходя из приоритетности рисков, определил плановые даты завершения и назначил тестировщиков, отвечающих за каждый тестовый сценарий. На основе предыдущих раундов он также пересмотрел плановые часы для каждого тестового сценария. Часть полученной им в качестве плана таблицы отслеживания тестовых сценариев показана на рис. 13.1.

№ этапа	Этап	Выполнено?
2.	Назначить тестировщиков, ответственных за выполнение тестовых сценариев из каждого комплекта тестов.	☑

После распределения ответственных Джамал подготовил семь копий этой таблицы для раунда 5. Чувствуя, что на очереди совещание по синхронизации работы группы, и зная, что ничто так не способствует удачному совещанию, как хорошая пища, он отправил группе тестирования сообщение по электронной почте.

Привет всем!

Пришла большая сборка. Наступило время для наиболее важного и сложного тестового цикла в проекте. Чтобы убедиться, что мы все работаем синхронно, пожалуйста, соберитесь в полдень в тестовой лаборатории для обеда, посвященного началу тестового цикла. Я выдам каждому таблицу отслеживания тестовых сценариев с распределением ответственных за выполнение тестовых сценариев, плановыми часами и плановыми датами завершения этого цикла. Чтобы никто не опаздывал, я закажу обед в нашем любимом индийском ресторане с обычными неострым, вегетарианским и кошерным вариантами. Помните, те, кто обожает ядреную острую куриную виндалу и аачар так же, как Эмма и я, должны не опаздывать, иначе мы съедим все!

С уважением,

Джамал

Совещание прошло успешно. Необходимо было выполнить много работы за короткое время, но благодаря планированию каникул, проведенному загодя Джамалом, загрузка не была чрезмерной, даже с учетом Рождественских праздников. Возможно, более важным было то, что каждый тестировщик горел желанием заняться тестированием большой сборки, тестовой версии с (почти) полным набором свойств. Как сказал один из младших тестировщиков, Луис Хасинто: "Наконец-то полная кружка Суматры, а не малюсенькие чашечки для эспрессо".

"Итак, — сказал Джамал, когда совещание и еда на столах подошли к концу, — отправляйтесь и найдите много ошибок. Я знаю, они прячутся в большой сборке. Встречусь с вами для разбора полетов во время пересменки в 16.00".

Тестировщики отправились выполнять тесты. Луис Хасинто начал свою работу с функционального теста для управляемых файлов. Настройка этого тестового сценария требовала запуска нового экземпляра SpeedyWriter на клиентской тестовой машине, а затем регистрации в системе управления документооборотом. Первый этап теста включал в себя создание нового файла, ввода в него некоторого текста и помещения файла в систему управления документооборотом.

№ этапа	Этап	Выполнено?
3.A	Привести систему, предназначенную для тестирования, и систему тестов в соответствующие исходные состояния. Если данные исходные состояния понадобятся для выполнения различных тестов или нескольких итераций данного теста, сохранить исходные состояния для последующего применения.	☑
3.B	При помощи ввода данных и других действий заставить систему, предназначенную для тестирования, реагировать на желаемые тестовые условия.	☑

Здесь Луис обнаружил, что функция создания файла серьезно нарушена. Из-за этого повреждались все новые файлы, когда производились определенные действия над шрифтами. Он выяснил, что существует единственный обходной путь — поместить файл в систему документооборота сразу после создания, не начиная редактировать его. Помимо выполнения плановых тестов, Луис потратил некоторое время на исследование и подготовку отчета о дефекте. Это заняло немногим больше времени, чем запланированные два с половиной часа, но ему нужно было время на локализацию и документирование столь критической ошибки. Перед тем как начать выполнение следующего тестового сценария, он пометил карандашом информацию о состоянии тестового сценария в таблице отслеживания тестовых сценариев.

№ этапа	Этап	Выполнено?
3.C	Наблюдать и оценивать выходные данные, поведение системы и ее состояния. Исследовать любые отклонения от ожидаемых результатов.	☑
3.D	Если необходимо, подготовить отчеты о проблемах в системе, предназначенной для тестирования.	☑
3.F	Собрать информацию и подготовить отчет о выполненных тестах.	☑

Затем Луис перешел к выполнению теста для неуправляемых файлов. Но в это время системный администратор по неосторожности отключил Тобрук — web- сервер на Linux, которым пользовался Луис. После некоторого недоумения он обнаружил, что сервер перезагружен, однако было потеряно около часа.

№ этапа	Этап	Выполнено?
4.	Решать проблемы, препятствующие тестированию, при их возникновении.	☑

После ежедневного подведения итогов с участием всей группы тестирования Луис провел еще пару часов за тестированием, затем в течение часа читал электронную почту и отвечал на письма, пока не завершился его 10-часовой рабочий день. Еще несколько минут он обсуждал ошибку, связанную со шрифтами, с Бинямином Цукерманом, одним из младших тестировщиков вечерней смены. В соответствии с таблицей отслеживания тестовых сценариев, сегодня вечером Бинямин назначен ответственным за тестирование шрифтов. Они также немного поговорили о скорой поездке Бинямина в Нью-Йорк, в частности о его планах посмотреть на реконструкцию в Нижнем Манхеттене. Наконец Луис отправился на поздний обед со своими друзьями в центр Остина.

Луис обсудил проблемы с Биньямином, но, к сожалению, он не говорил в течение дня с Эммой Мурхаус. Это не было бы проблемой, если бы во время отправки Луисом отчета о дефекте всем тестировщикам Эмма не была занята запуском автоматизированных тестов производительности для Solaris, поэтому она не прочитала письмо. И это также не было бы проблемой, если бы тестовые скрипты Эммы не зависели от корректной работы системы как раз в той части, которую Луис описал в своем отчете о дефекте как серьезно поврежденную.

Эмма ушла в понедельник вечером с работы и не знала, что тесты производительности Solaris дали неверные результаты. (Она не добавила в инструмент средства, которые позволили бы скриптам послать ей сообщение, если что-то в поведении было неверным.) И до утра вторника ей не было известно об этом упущении. Эмма была расстроена и огорчена. Сначала она пожаловалась коллегам, а затем рассказала Джамалу, что произошло. Джамал понимающе пожал плечами и сказал: "Причуды автоматизированного тестирования". Испытав облегчение, но не удивившись понимающей реакции Джамала, Эмма переработала тестовые скрипты для того, чтобы перезапустить их вечером во вторник.

№ этапа	Этап	Выполнено?
3.E	Если необходимо, подготовить отчеты о проблемах в системе тестов и/или исправить их.	☑

В результате происшедшего инцидента Джамал понял, что ему нужно укрепить процесс. Он разослал группе тестирования следующее сообщение с пометкой "Срочно!".

Всем!

С момента начала тестирования в октябре мы использовали неформальный процесс рассылки отчетов о дефектах по электронной почте всей группе с целью просмотра и получения информации. Считаю, что этот процесс способствовал росту профессионализма при подготовке отчетов о дефектах и своевременному информированию о них тестировщиков.

Моя работа как менеджера состоит в том, чтобы делать хорошие идеи частью нашего рабочего процесса. В этом отношении я допустил ошибку, поскольку, возможно, сказал, что чтение сообщений с отчетами об ошибках является низкоприоритетной деятельностью, что мы можем делать это один раз в день или полностью игнорировать в случае сильной занятости. Однако идти в ногу с результатами тестирования других и изучать, как эти результаты могут повлиять на проведение тестирования каждым из вас, является на самом деле ключевым для эффективного и производительного тестирования.

Я допустил ошибку управления и хочу ее исправить. Прошу вас считать чтение отчетов о дефектах, подготовленных другими тестировщиками, центральной и повторяющейся частью вашей повседневной работы. Прошу регулярно до начала выполнения любого тестового сценария проверять почтовый ящик и знакомиться с отчетами о дефектах, подготовленными вашими коллегами. Отправляя такие сообщения, прошу взять за правило в качестве первого слова в строке "тема письма" писать "ОШИБКА:", а затем указывать номер ошибки и столько информации о ней, сколько уместится в строке.

Некоторые из вас уже следуют описанной практике. Однако, поскольку я не внедрил эту практику с самого начала выполнения тестов, есть, по всей вероятности, такие, кто ей не следует. Для этих тестировщиков новая практика означает увеличение загрузки. Если вы уверены в том, что вводимое в процесс выполнения тестов изменение означает существенное увеличение загрузки и помешает завершить ранее порученные задачи в согласованные сроки, пожалуйста, не колеблясь, приходите ко мне, чтобы уточнить сроки выполнения работ и сферы ответственности.

В заключение хочу подчеркнуть, что я отправляю это письмо не для того, чтобы бросить тень на чью-либо работу или подвергнуть сомнению чью-то точку зрения. Я весьма удовлетворен работой каждого из вас в этом непростом проекте. Моя цель - скорректировать мое упущение в руководстве и улучшить процесс. Это увеличение загрузки в долгосрочной перспективе сэкономит наше время, способствуя повышению производительности и эффективности.

С уважением,

Джамал

№ этапа	Этап	Выполнено?
4.	**Решать проблемы, препятствующие тестированию, при их возникновении.**	☑

После того как проблема с тестированием производительности была решена, тестирование продолжилось во вторник с большой скоростью, но тщательно продуманно. Луис продолжил с утра тестирование неуправляемых файлов. Он обнаружил шесть разных проблем и написал шесть отчетов. Также вчерашнее непреднамеренное моделирование краха сервера показало наличие серьезной проблемы, которая в противном случае была бы пропущена. Обсудив проблему с Лин Цу, Луис потратил полчаса на обновление одного из тестовых сценариев для проверки обработки ошибок, чтобы учесть новое условие. Так случилось, что тестирование неуправляемых файлов не было полностью завершено до утра среды.

В 16.00 в среду, как и во все предыдущие дни, Джамал собрал группу тестирования для обсуждения результатов тестирования и передачи смены. Он попросил каждого тестировщика сообщить, какие тесты они выполнили и сколько человеко-часов потратили на каждый из них, а также рассказать обо всех ошибках, обнаруженных при помощи каждого теста. По мере того как тестировщики докладывали о состоянии дел, Джамал обновлял измерения, используя сводную таблицу выполнения тестовых сценариев (см. рис. 13.4). После завершения выступлений Джамал обновил все проектные измерения, используя сводную таблицу комплектов тестов (см. рис. 13.5). Совещание обычно занимало примерно 45 минут, но с учетом объемов тестирования и ярких впечатлений от большой сборки на этой неделе совещание длилось около часа.

После совещания Джамал посмотрел на обновленные таблицы. Он увидел, что для этого тестового цикла тестирование безопасности и секретности является наиболее рискованным. Биньямин увяз в болоте важных функциональных ошибок. Другой младший тестировщик, Джон Ву, работающий в вечернюю смену, обнаружил массу проблем в документации. Лин Цу управляет тестированием локализации и удобства использования. (Оказалось, что ей нужно больше времени, чем запланировано, поскольку были обнаружены ошибки регрессии в процессе инсталляции.) Значит, все, кому можно было бы поручить тестовые сценарии по безопасности и секретности, были заняты.

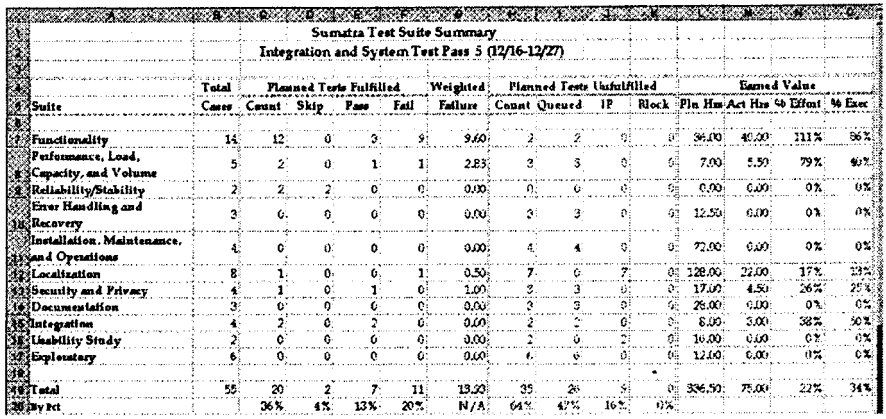
Джамал понял, что у него нет другого выбора, кроме как вызвать резервного тестировщика Джамала Брауна. Он знал, что, как бы ни старался, он не сможет ни потратить 17 часов на тестирование в ближайшие три дня, ни "потушить другие пожары", связанные с тестированием. Поэтому он заново спланировал тестирование, как показано на рис. 13.6.

Sumatra Test Case Summary

Integration and System Test Pass 5 (12/16-12/27)

Assigned To	Test ID	Test Suite/Case	Test Status	Test Environment	Bug ID	Bug RPN	Run By	Plan Date	Act Date	Test Effort	Plan Effort	Actual Effort	Test Duration	Comment
	1.000	*Functionality*												
DLK	1.001	File	Fail	SW/H2/B1	1832	1	DLK	12/16	12/16	2.0	2.0	3.0	4.0	
JW	1.002	Edit	Warn	SW/H2/B1			DLK	12/16	12/16	0.5	0.5	1.0	3.0	
BEZ	1.003	Font	Fail	SS/H2/B7	1832	1	BEZ	12/16	12/16	0.5	0.5	2.0	4.0	
DLK	1.004	Tables	Pass	SS/H2/B5			DLK	12/17	12/17	3.0	3.0	3.0	5.0	
BEZ	1.005	Printing	Pass	SW/H1/B6			BEZ	12/16	12/16	2.0	2.0	2.0	4.0	
LJ	1.006	Managed Files	Fail	SW/H1/B8	1832	1	LJ	12/16	12/16	2.5	2.5	3.0	3.0	
LJ	1.007	Non-Managed Files	Fail	SS/H2/B8	1832	1	LJ	12/17	12/18	4.0	4.0	6.0	6.0	
					1837	6								
					1838	25								
					1840	6								
					1845	3								
					1846	2								
					1847	8								
BEZ	1.008	DMS Check-In New	Fail	SS/H1/B6	1836	1	BEZ	12/17	12/17	2.5	2.5	6.5	5.0	
					1832	1								
					1841	5								
					1844	15								
BEZ	1.009	DMS Check-In Existing						12/17		3.0	3.0			
BEZ	1.010	DMS Check-Out						12/18		3.5	3.5			
LTW	1.011	DMS Load/Migrate Online	Fail	SS/H1/B2	1833	2	LTW	12/16	12/16	3.5	3.5	4.0	4.0	
HS	1.012	DMS Load/Migrate NOL	Fail	SS/H2/B3	1833	2	HS	12/16	12/16	3.0	3.0	3.0	3.0	
HS	1.013	DMS Load/Migrate Offline	Fail	SE/H2/B3	1833	2	HS	12/17	12/17	3.5	3.5	3.5	3.5	
HS	1.014	DMS File Information	Fail	SW/H1/B3	1833	2	HS	12/17	12/17	2.5	2.5	3.0	3.0	
		Suite Summary						12/18	12/18		36.0	40.0	47.5	

Рис. 13.4. Отчет о статусе тестовых сценариев (раунд 5, день 3)



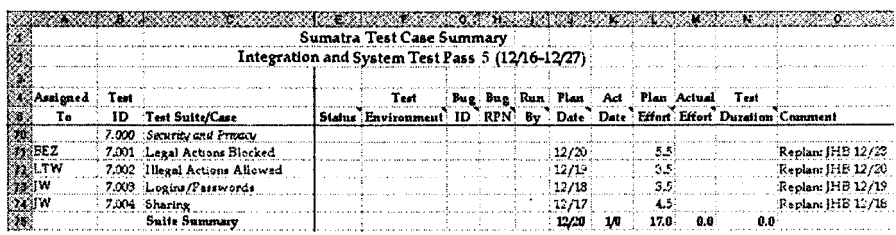
Sumatra Test Suite Summary
Integration and System Test Pass 5 (12/16-12/27)

Suite	Total Cases	Planned Tests Fulfilled: Count	Skip	Pass	Fail	Weighted Failure	Planned Tests Unfulfilled: Count	Queued	IP	Block	Earned Value: Pln Hrs	Act Hrs	% Effort	% Exec
Functionality	14	12	0	3	9	9.60	2	2	0	0	36.00	40.00	111%	86%
Performance, Load, Capacity, and Volume	5	2	0	1	1	2.83	3	3	0	0	7.00	5.50	79%	40%
Reliability/Stability	2	2	2	0	0	0.00	0	0	0	0	0.00	0.00	0%	0%
Error Handling and Recovery	3	0	0	0	0	0.00	3	3	0	0	12.50	0.00	0%	0%
Installation, Maintenance, and Operations	4	0	0	0	0	0.00	4	4	0	0	72.00	0.00	0%	0%
Localization	8	1	0	0	1	0.50	7	0	7	0	128.00	22.00	17%	13%
Security and Privacy	4	1	0	1	0	1.00	3	3	0	0	17.00	4.50	26%	25%
Documentation	3	0	0	0	0	0.00	3	3	0	0	26.00	0.00	0%	0%
Integration	4	2	0	2	0	0.00	2	2	0	0	8.00	3.00	38%	50%
Usability Study	2	0	0	0	0	0.00	2	0	2	0	10.00	0.00	0%	0%
Exploratory	6	0	0	0	0	0.00	6	6	0	0	12.00	0.00	0%	0%
Total	55	20	2	7	11	13.93	35	26	9	0	336.50	75.00	22%	34%
By Pct		36%	4%	13%	20%	N/A	64%	47%	16%	0%				

Рис. 13.5. Отчет о комплектах тестов (раунд 5, день 3)

№ этапа	Этап	Выполнено?
5.	Ежедневно готовить отчет о ходе тестирования, уточнять ответственных за выполнение тестов и пересматривать планы и приоритеты.	☑
6.	Если необходимо, удалить невыполнимые или устаревшие тесты в порядке, обратном их приоритетности (сначала удалять тесты низших приоритетов, в последнюю очередь — высших).	

Четверг и пятница прошли почти по плану лишь с небольшими задержками. Лин Цу и Эмма, чьи несколько тестов должны были завершиться утром в субботу, согласились встретиться с Джамалом в тестовой лаборатории около 10.00, чтобы закончить тестовый цикл. Младший тестировщик Дейл, который обычно проводил приемочное тестирование тестовых версий, также предложил свои услуги, чтобы подготовить тестовую версию для следующего цикла к утру понедельника.



Sumatra Test Case Summary
Integration and System Test Pass 5 (12/16-12/27)

Assigned To	Test ID	Test Suite/Case	Status	Test Environment	Bug ID	Bug RPN	Run By	Plan Date	Act Date	Plan Effort	Actual Effort	Test Duration	Comment
	7.000	*Security and Privacy*											
BEZ	7.001	Legal Actions Blocked						12/20		5.5			Replan: JHB 12/23
LTW	7.002	Illegal Actions Allowed						12/19		3.5			Replan: JHB 12/20
JW	7.003	Logins/Passwords						12/18		3.5			Replan: JHB 12/19
JW	7.004	Sharing						12/17		4.5			Replan: JHB 12/18
		Suite Summary						12/20	1/0	17.0	0.0	0.0	

Рис. 13.6. Таблица тестовых сценариев, показывающая перепланирование выполнения комплекта тестов безопасности и секретности

Джамал посоветовался с инженерами-тестировщиками. Поскольку младшие тестировщики много работали в течение недели, Лин Цу, Эмма и Джамал решили не просить их приходить в субботу. Джамал сказал инженерам: "Нет смысла пытаться пробежать марафон как стометровку. У нас еще впереди три месяца работы, и я полагаю, что нам еще потребуются резервы этих ребят".

Вскоре после этого Джамал переговорил с Дейлом. "Дейл, — сказал он. — Я ценю твое предложение, но я бы хотел, чтобы ты пришел в другой раз, и, возможно, я попрошу тебя потратить выходные на проект позже. Лучше дай мне твой волшебный контрольный список, и я проведу приемочное тестирование сборки сам".

Дейл улыбнулся: "Конечно. У меня в голове были другие идеи, но я полагаю, что могу передать список".

Затем Джамал связался с системным администратором Петрой, которая обычно устанавливала тестовую версию, и попросил придти в выходной. Она согласилась, вероятно, с большим желанием, чем раньше, после того как узнала, что с ней рядом будет работать Джамал, лично выполняя приемочное тестирование.

Наконец в субботу утром тестовый цикл был завершен, и новая версия прошла приемочное тестирование. Завершив первый цикл раунда 5, Джамал подвел итоги измерений и подготовил отчет о раунде. (В главе 15 мы увидим, как Джамал представляет результаты цикла на совещании о состоянии проекта "Суматра".) Затем он положил все документы, которые он обновил и создал заново, в проектный репозиторий.

№ этапа	Этап	Выполнено?
7.	Периодически отчитываться о ходе и результатах тестового цикла.	☑
8.	Поместить все документы о ходе работ, начальные состояния, обновленные средства тестирования, другие элементы системы тестов и прочую информацию, получаемую на постоянной основе, в проектную библиотеку или систему управления конфигурациями. Вести контроль изменений всех элементов.	☑

Построение успешного процесса выполнения тестов

Как и вершина айсберга, процесс выполнения тестов — та часть тестирования, которая видна, но она держится на плаву благодаря качественной работе в ходе предыдущих восьми процессов. Какие же действия и достижения группы тестирования способствуют успешному процессу выполнения тестов?

Выявление опасных мест

В ходе выполнения тестов я использую различные источники информации, чтобы выдвинуть обоснованные гипотезы по поводу того, где нахо-

дятся наиболее неприятные ошибки, и в первую очередь проверяю эти области. Когда я только начинаю выполнение тестов, я применяю результаты анализа рисков качества (см. главу 2), чтобы определить, с чего начать тестирование. Позднее, когда я уже ближе познакомлюсь с системой, у меня появляются более удачные идеи о том, где располагаются места наибольшего риска: там, где я нашел ошибки, и там, где я еще не тестировал. Воспользовавшись этими знаниями, я могу уточнить, что тестировать дальше. (Тем не менее я должен быть осторожным, чтобы не позволить строгим техническим соображениям, например искать там, где мы уже нашли ошибки, скрыть от меня тот факт, что риски, связанные с применением системы заказчиком, и другие соображения бизнеса также должны оказывать влияние на приоритетность тестирования.) В нашем примере Джамал сначала концентрирует внимание на ранее заблокированных функциональных тестах, а также на тестах секретности, безопасности и обработки ошибок, на которые оказывают воздействие новые свойства последней версии.

Меня также волнуют два особых риска качества: исправления дефектов, которые на самом деле не решают проблему, и исправления дефектов, которые порождают новые проблемы. Эти риски являются сферой проверочного и регрессионного тестирования соответственно. Поскольку исправления ошибок, которые не решают проблем, особенно проблематичны, я предпочитаю начинать с этих тестов каждый тестовый цикл.

В главе 6 я подробно рассмотрел стратегии работы с риском регрессии, но здесь хотел бы отметить, что выполнение тестов — это та фаза, когда нужно запускать эти стратегии в действие. Проблема может возникнуть для стратегий риска регрессии, которые предполагают повторение тестов. Этот подход может конфликтовать с нацеленностью на выполнение новых тестов и проверку исправлений дефектов. Я не рекомендую отказываться от тщательно спланированной стратегии регрессии при первых признаках проблем в проекте, но можно пересмотреть подход, который предполагает повторение большого числа тестов, если результаты тестирования показывают небольшой риск регрессии, тогда как другие риски оказываются значительно более серьезными, чем считалось первоначально.

Другая опасность — это обнаружение проблем, которых мы не ожидали. Тщательно спланированное методически верное множество процессов тестирования, аналогичное описываемому в этом книге, помогает предотвратить классические ошибки тестирования[1]. Однако оно может породить опасные темные пятна. Ошибочное предположение в ходе анализа рисков качества может пройти волной через все процессы оценки затрат, планирования тестирования и разработки системы тестов, порождая большие и незаметные пробелы в тех местах, которые действительно важно проверить. Простой взгляд на обнаруженные ошибки, чтобы понять, где есть еще ошибки, не решит эту проблему, поскольку это вам ничего не скажет об ошибках, которых вы не нашли, так как не тестировали эти области совсем. Вот почему сейчас я усиливаю свои планы тестиро-

[1] Другой взгляд на ошибки тестирования приводится в статье Брайана Марика (Brian Marick) "Classic Testing Mistakes» на www.testing.com.

вания за счет исследовательского тестирования как части общей стратегии. В нашем примере Джамал включил исследовательское тестирование в оценку затрат (см. главу 3) и в план тестирования (см. главу 6)[1].

Наконец, тестировщики не должны оставлять без внимания ошибки, которые не относятся к краткосрочным целям тестирования. В случае использования подробных ручных тестовых скриптов или проведения автоматизированного тестирования мы можем попасть в ловушку, если будем игнорировать ошибки, не относящиеся непосредственно к тому, что мы тестируем. Например, если я тестирую функциональность, связанную с открытием файла, и замечаю проблему с утилитой импорта файла, являющуюся целью другого теста, я должен потратить время на исследование проблемы и написание отчета об ошибке. Это особенно важно, если ошибка, которую я заметил, не покрывается другим тестом. Данный вопрос относится к теме тестирования по возможности, которая обсуждалась в главе 2. Часть работы тестировщика — представить ожидания и мнение о системе разумного пользователя. Когда я тестирую, я всегда ищу такое поведение программы, которое, будучи разумным пользователем, я счел бы раздражающим, запутывающим, некорректным или причиняющим беспокойство. Как тест-менеджер я постоянно напоминаю группе тестирования о необходимости делать то же самое.

Генерация, сбор и распространение значимой информации

Точно так же, как фармацевтические компании тестируют лекарства, а инженеры проверяют машиностроительные материалы, мы тестируем программы, чтобы получить новые знания о них. Цель этого процесса — получение информации. Таким образом, имеет смысл спланировать эксперименты — наши тесты — так, чтобы собрать максимум информации и передать ее тем, кто умеет ею пользоваться.

Информация распространяется в основном вне группы тестирования. Эти темы более подробно рассматриваются в главах 14 и 15, сейчас же отметим, что основная цель тестирования — получение информации, которую проектная группа может использовать для принятия разумных решений относительно качества. Комиссия по разбору дефектов или группа управления изменениями может расставить приоритеты ошибок и решить, какие из них устранять (см. главу 16). Разработчики могут принять решение, как исправить эти ошибки. Группа управления проектом может решить, как скорректировать работы, чтобы проект двигался в нужном направлении. Чтобы способствовать принятию разумных и обоснованных решений, процесс выполнения тестов должен обеспечивать сбор результатов тестирования в ясных отчетах и передачу результатов всем заинтересованным сторонам. В нашем примере тестировщики немедленно рассылают отчеты о дефектах, а тест-менеджер готовит еженедельный отчет о результатах тестового цикла для руководства. Как указывалось в

---

[1] Для получения дополнительной информации о методах исследовательского тестирования я рекомендую "*Тестирование программного обеспечения*" Канера, Фолка и Нгуена, "*Уроки, полученные при тестировании программного обеспечения*" Канера, Баха и Петтикорда, и web-сайт Джеймса Баха www.satisfice.com.

главном примере в главе 12, Джамал присутствует на еженедельных совещаниях группы управления изменениями, на которых обсуждают и принимают решения по дефектам.

Другое направление распространения информации – внутри группы тестирования. Результаты предыдущих тестов могут оказывать влияние на последующие тесты. Хорошим примером этого является рассказ Луиса об ошибке со шрифтами, которую он обнаружил. Однако мы видели и отрицательный пример, когда Эмма не прочла отчет об ошибке, подготовленный Луисом.

Это нарушение процесса имело два отрицательных последствия. Первое – выполнение работы заново. Эмма вынуждена была потратить время и силы на повторное выполнение теста производительности для Solaris. Второе, возможно, более серьезное последствие – однодневная задержка в обнаружении ошибок, относящихся к производительности. Вы можете сказать: "Что значит один день?" Но проект – это ничто другое, как последовательность дней, а дни складываются в недели[1]. Исследование моделей устранения дефектов показывает, что более раннее обнаружение и исправление дефектов снижает общее число обнаруженных дефектов, что в итоге экономит деньги и время[2]. Использование перекрестных просмотров отчетов об ошибках и отчетов о ходе тестирования может уменьшить коммуникационные проблемы. Тест-менеджер должен быть уверен, что тестировщики решают эти задачи. Письмо Джамала группе тестирования – это пример процесса, который может хорошо работать, и пример управления, которое может внедрить этот процесс[3].

Как показывает этот пример, внутренняя информация о текущем проекте может способствовать улучшению системы тестов и процесса тестирования. Можно сказать, что в то время как система тестов проверяет систему, предназначенную для тестирования, в процессе выполнения тестов, система, предназначенная для тестирования, также проверяет систему тестов и процесс выполнения тестов. Это означает, что проницательные тестировщики и тест- менеджеры могут собирать информацию о качестве системы тестов и процесса выполнения тестов и находить возможности для их усовершенствования. В рассматриваемом нами примере Луис и Лин Цу внесли изменение в один из тестовых сценариев на основе случайно обнаруженной ошибки. В ходе первоначального отбора тестов и назначения ответственных за их выполнение Джа-

---

[1] Более 25 лет назад Фред Брукс рассматривал опасность небольших отставаний и разъедающий эффект такого отношения к небольшим отставаниям проекта в книге "*Мифический человеко-месяц*». Позднее Том де Марко (Tom DeMarco) писал в книге "*The Deadline*», что "существует бесконечное число способов потерять день для проекта, но ни одного способа вернуть этот потерянный день обратно». Берегите время и энергию вашей группы с самого начала проекта, когда кажется, что еще всего много, позднее вы будете приятно удивлены этой предусмотрительностью, когда время сожмется, а энергии станет меньше.

[2] См., например, книгу Стефана Кана (Stephen Kahn) "*Metrics and Models in Software Quality Engineering. Second Edition*».

[3] Заметим, что Джамал разобрался, что сбой в работе связан с проблемой в процессе, а не с людьми. Я упоминал ранее эксперимент Деминга с красными бусинками в качестве примера того, почему руководство должно брать ответственность за недостатки процессов. Вместо того чтобы подвергнуть наказанию Эмму (она следовала процессу так, как она это понимала), Джамал принял ответственность за инцидент на себя и исправил процесс.

мал уточнил время на выполнение тестов на основе информации о предыдущем тестовом цикле.

Внутренняя информация необходима также для управления задачами, которые должны быть выполнены. Ведущий тестировщик или тест-менеджер должен иметь возможность проверить, что группа тестирования завершает все плановые действия тестирования. Здесь пригодится раздел "Выполненная работа" сводной таблицы комплектов тестов. В нашем примере Джамал посмотрел на ситуацию с выполнением работ и увидел, что многие из комплектов тестов требуют больше времени, чем запланировано. Это заставило его проанализировать сами комплекты тестов, которые еще остались незавершенными. Он понял, что определенные тесты не могут быть завершены или даже выполнены без принятия корректирующих мер. Качественный процесс тестирования позволяет руководителю видеть такие ситуации по мере их развития и предпринимать меры по снижению рисков.

Правильная интерпретация результатов тестирования

Основной характеристикой, которая отличает ценную информацию от бесполезного шума, является точность. Результаты тестирования, которые мы сообщаем, должны быть точными. В процессе выполнения тестов, описанном выше, подэтапы 3.C, 3.D и 3.E требуют от тестировщика тщательной проверки несоответствия в результатах тестирования; нужно различать ошибки в системе, предназначенной для тестирования, и ошибки в системе тестов. Определение трех различных этапов вместо одного "Подготовить отчет об ошибке, если ожидаемые и фактические результаты не совпадают" указывает на необходимость правильной интерпретации результатов тестирования.

Выполнение тестов — это процесс, осуществляемый людьми. Наши возможности наблюдения и оценки иногда дают сбой. Наше ожидание корректного поведения системы иногда неверно. Непостижимые системы, передаваемые на тестирование, иногда скрывают подтверждения своего верного и неверного поведения. Мы можем ошибочно принять проблему в тестовой среде, скриптах, инструментах и других частях системы тестов за проблему в тестируемой системе. По этой и другими причинам мы можем сообщить о сбое, в то время как программа работает верно, или наоборот, что еще хуже (см. таблицу 13.1).

Таблица 13.1. Выходные данные по интерпретации результатов тестирования

	Тестировщик отчитывается ...	
Поведение ...	**о прохождении теста**	**об ошибке**
Корректное	Повышенное доверие	Ненужное беспокойство
Некорректное	Ошибочное доверие	Полезная диагностическая информация

Такие виды просчетов не должны удивлять нас. Для пояснения давайте увеличим графическое изображение этапа 3 процесса выполнения тестов (см. рис. 13.7). С учетом всех меняющихся частей картинки неверная интерпретация может стать проблемой. Добавьте к этому, что все тестировщики и сотрудники проектной группы, вероятно, должны работать с системой тестов и системой, предназначенной для тестирования, пока мы выполняем тесты, и такие недоразумения становятся еще более вероятными[1].

Чтобы решать проблемы наблюдательности тестировщика и его способности к оценке, обычно я прошу старших тестировщиков проверять результаты тестирования младших тестировщиков и назначаю разных тестировщиков ответственными за выполнение одних и тех же тестовых сценариев в ходе следующих друг за другом циклов. Полное решение проблемы распознавания корректного и некорректного поведения системы — проблемы оракула, рассмотренной в главе 10, — зависит от наличия непротиворечивых спецификаций требований и архитектуры и надежного способа прогнозирования корректного ответа на любое обращение к системе. Когда я опускаю эти элементы, я, как правило, делаю ошибки в отчетах о дефектах. (В главе 14 рассматриваются способы решения политических проблем, связанных с информированием о несуществующих ошибках, а в главе 16 обсуждаются некоторые идеи по поводу того, как качественный процесс управления изменениями может разрешить вопросы, связанные с неопределенными отчетами об ошибках.) Проблема с загадочными программами сводится к написанию тестопригодных программ. Только высшее руководство может решить эту проблему, обеспечивая вовлечение тестировщиков в проект во время проектирования и разработки программы.

Что должны делать тестировщики, если они обнаружили проблему в системе тестов во время тестирования? Процесс, описанный выше, требует немедленной корректировки неверного теста. Некоторые предлагают работать с системой тестов так же, как с системой, предназначенной для тестирования: подготовить отчет об ошибке, предоставить комиссии по разбору дефектов право принять решение, стоит ли исправлять ее, собирать измерения, связанные с проблемами качества системы тестов, — в общем, все, что необходимо. Другими словами, поскольку система тестов проверяет систему, предназначенную для тестирования, и наоборот, система, предназначенная для тестирования, проверяет систему тестов, симметрия означает, что мы должны следовать процессу, когда обстоятельства изменяются.

В большинстве проектов, в которых я участвовал, такой процесс не имел смысла. Вернемся к вопросу качества. Если мы признаем, что арбитром качества в последней инстанции является пользователь или заказчик, то система, предназначенная для тестирования, и система тестов имеют принципиально разных заказчиков и пользователей. Система, предназначенная для тестирования, является внешним передаваемым

---

[1] Стив Сплейн (Steve Splaine), один из рецензентов, отметил, что это особенно верно, "если не было предпринято никаких усилий, для того чтобы разбить [или] скоординировать использование... разделяемых тестов и данных". Потенциальные проблемы становятся еще более острыми, чем больше группа и чем сложнее тестовая среда.

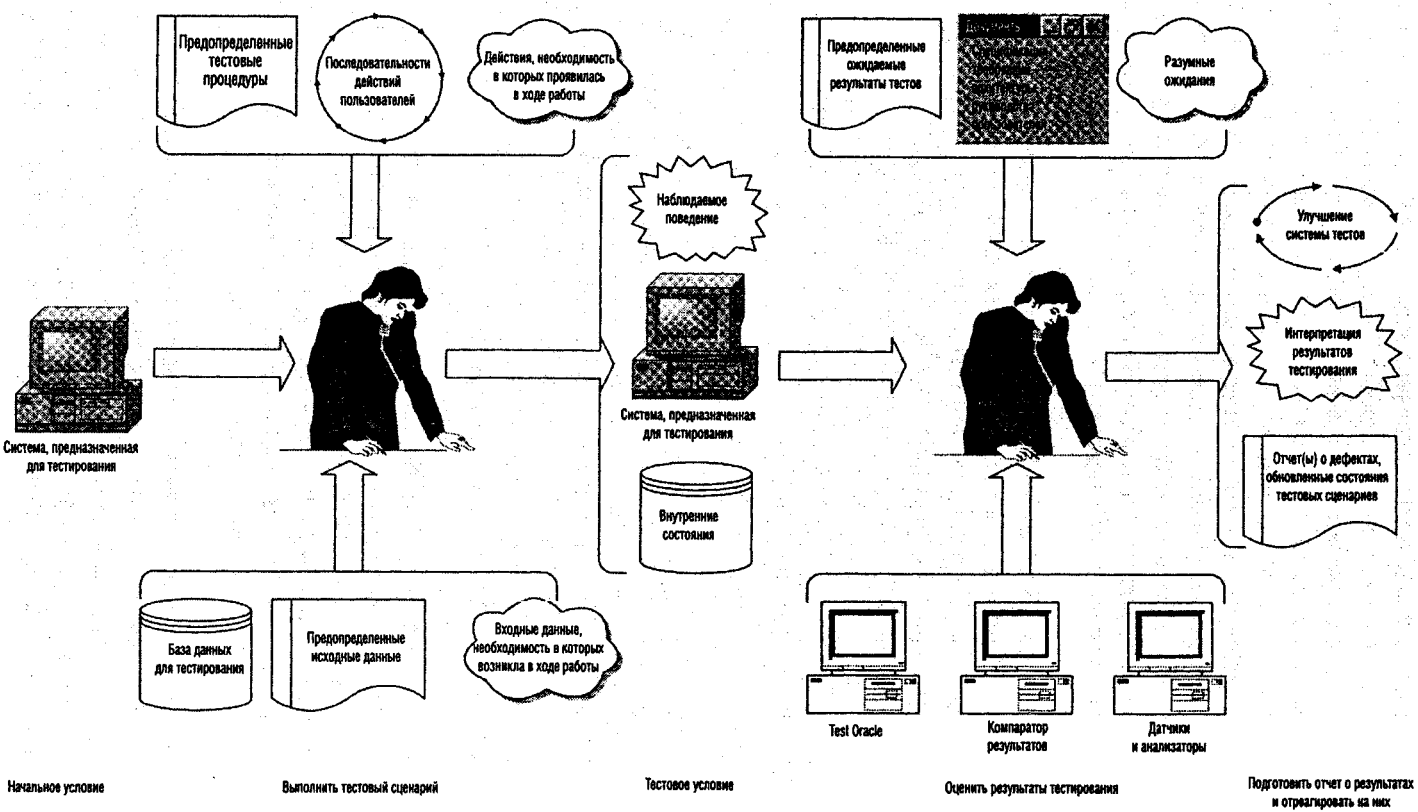


Рис. 13.7. Выполнение тестовых сценариев

продуктом, разработанным для решения реальных проблем для заказчиков и пользователей, которые оплачивают создание системы. Система тестов – это обычно внутренние передаваемые результаты, которые применяется теми же людьми, которые ее разработали, для выполнения своих обязанностей по тестированию. Система, предназначенная для тестирования, является продуктом. Система тестов – это инструмент, помогающий создать продукт.

Итак, нередко сотрудники, создающие тестовые сценарии, и сотрудники, которые их выполняют, – это разные люди. Например, консультанты иногда создают системы тестов и инструменты для компаний. В некоторых случаях проблемы качества системы тестов могут иметь неприемлемые последствия, например, в системах с высокими требованиями к безопасности. Фиксация процесса для создания системы тестов может добавить формализм, который придаст уверенности в том, что передается качественная система тестов.

Наконец, отмечу, что тестировщики не всегда могут знать с полной определенностью, в каком из четырех квадрантов таблицы 13.1 они находятся; на выяснение этого требуется время. Исследование ошибок и поиск других необычных или многообещающих (с точки зрения обнаружения ошибок) вариантов поведения системы требует непрогнозируемых объемов затрат. Поэтому фактическое количество человекочасов, необходимых для выполнения множества тестов, может существенно отличаться от плана. Тестировщики всегда должны внимательно следить, на что расходуется их время. Ведущие тестировщики и менеджеры должны оказывать помощь своим сотрудникам в уравновешивании необходимости двигаться вперед по запланированным тестовым сценариям и необходимости с полной определенностью понимать смысл результатов тестирования.

Не все ошибки заслуживают одинакового исследования. Периодически возникающий дефект, когда при неправильном указании шрифта появляется малопонятное сообщение об ошибке один раз на десять обращений, по всей вероятности, может быть спокойно проигнорирован, чтобы не откладывать выполнение других тестов на часы. Однако периодически возникающий дефект, который время от времени разрушает базу данных разделяемого доступа, может служить основанием для отсрочки выполнения тестового цикла, пока сбой не будет локализован и проблема не станет воспроизводимой.

Демонстрация надлежащего отношения

Тема корректной интерпретации результатов подводит нас к вопросу отношений. Качественный процесс тестирования должен укреплять надлежащее отношение тестировщиков, а надлежащее отношение тестировщиков точно так же повышает возможности процесса тестирования в получении полезных результатов. Как отмечалось в главе 8, надлежащее отношение тестировщика к делу включает в себя профессиональный пессимизм, разумную любознательность и способность концентрироваться. Поскольку тестировщики должны сохранять скепсис и выступать в проекте разработки в роли профессиональных пессимистов, я призываю

к активной позиции в отношении подготовки отчетов об ошибках в ходе тестирования. Такой подход имеет несколько следствий.

- Если тестировщики находятся в сомнении, они принимают за основу, что наблюдаемое поведение является неверным, пока не убедят себя в обратном.
- Тестировщики сообщают о любой ситуации как об ошибке, если экранная справка, руководство пользователя или любая другая документация указывает, что правильное поведение системы отличается от наблюдаемого.
- Тестировщики *всегда* сообщают обо всех событиях, которые ведут к катастрофическим вариантам поведения, независимо от того, насколько часто они происходят или являются невоспроизводимыми. Такие варианты поведения включают в себя любую потерю данных, крах, остановку, зависание системы и любые другие проблемы надежности, качества данных или серьезного снижения производительности, которые затрагивают систему, предназначенную для тестирования, центральную систему, любое сосуществующее программное обеспечение, любую смежную или взаимодействующую систему.
- Наконец, тестировщики сообщают обо всех обстоятельствах, при которых система, предназначенная для тестирования, не соответствует разумным ожиданиям поведения или качества программы или является запутанной, обманчивой или противоречивой.

Такой способ мышления, однако, должен быть сбалансирован с ходом проекта. На ранних стадиях тестирования качество системы обычно невысокое. Тестировщики должны сосредотачиваться на поиске опасных проблем, не тревожась, например, за ошибки в правописании в руководстве пользователя. По мере улучшения системы тестировщики будут наводить глянец на соответствие системы требованиям, хотя в этом случае их часто критикуют за интерпретацию результатов тестирования, считая их отчеты о дефектах придирками. Мы не должны стыдиться, размещая отчеты об ошибках с низким приоритетом в соответствующие моменты проекта. Это одна из форм разумной любознательности.

Способность концентрироваться означает, что квалифицированные тестировщики внимательно следят и сразу информируют о *расхождениях в интерпретации данных* о работе системы. Расхождения в интерпретациях могут возникать не только между тестировщиком и программистом, но и между двумя программистами, которые готовят взаимодействующие компоненты, являющиеся источником большинства ошибок, обнаруживаемых в ходе фазы комплексного тестирования. Классический пример такой ошибки: в ходе полета на Марс, осуществлявшегося НАСА, возникла неразбериха с использованием метрических и английских единиц измерения данных, в особенности касающихся скорости и тормозной тяги. Использование различных единиц разными программистами привело к полному провалу полета и к потере всей полученной информации, когда спускаемый аппарат на большой скорости врезался в марсианскую поверхность и предположительно развалился на куски.

В таких случаях ошибка существует не в одном или в другом компоненте, а во взаимодействии компонентов. До того как появляется строчка с неверным кодом, ошибка одновременно существует в головах двух разных людей. Если бы нужный человек, вероятно тестировщик, в нужное время, например в ходе определения требований, задумался о том, чтобы задать обоим программистам вопрос о единицах измерения: английские или метрические, — можно было бы избежать этого неудачного сценария.

Поэтому тестировщики должны быть бдительными к появлению признаков несогласия или различного понимания внутри групп программистов. Иногда задавая "глупые вопросы", например: "Простите, но я запутался: мы используем метрические или английские единицы в проекте?", можно вызвать обсуждения, которые вскроют непонимание, прежде чем это непонимание превратится в опасные, дорогие и трудно диагностируемые проблемы в ходе системного тестирования или, что еще хуже, позднее. Более 25 лет назад в первом издании книги "*Мифический человеко-месяц*" Фред Брукс писал о проектах, которые осуществлялись в 60-е годы прошлого века: "Задолго до написания кода спецификация должна быть передана сторонней группе тестирования для тщательного рассмотрения полноты и ясности. Как считает [В. А.] Высоцкий [из проекта Safeguard, выполнявшегося в Bell Telephone Laboratories], сами разработчики сделать это не могут: "Они не могут признаться в том, что не видят проблем, они будут счастливо прокладывать свой путь через пропущенные и темные места"[1].

Ошибка — всегда ошибка, независимо от того, есть она в компоненте или в интерфейсе компонентов, в системе, в архитектуре или в требованиях. Наиболее производительные группы тестирования, в которых я работал, имели явное или неявное разрешение сообщать о таких видах ошибок и привлекались к работе над проектом достаточно рано, чтобы сделать что-либо полезное на каждом из этапов.

Эффективная работа

До некоторой степени эффективный процесс тестирования основан на эффективной системе тестов. Это возвращает нас к проблемам разработки системы тестов, которые обсуждались в главе 11. Например, минимизация взаимозависимостей тестовых сценариев и разделяемых множеств тестовых данных предотвращает ненужные переработки сценариев, задержки и, возможно, неверную интерпретацию результатов тестирования.

Однако вопросы эффективности возникают также в ходе выполнения тестов, даже при самой эффективной системе тестов. Как тест-менеджер я хочу добиться минимального перекрытия работ и избежать задержек при проведении тестирования. Часть этой работы требует ясной постановки задач для всех тестировщиков, какие тесты они должны выполнять и когда. В нашем примере Джамал использует столбец "Ответственный" в сводной таблице тестовых сценариев для определения ответственного за выполнение каждого теста. С помощью столбца "Плановая дата" он упорядочивает тесты для достижения максимальной эффективности и для

---

[1] См. "*Мифический человеко-месяц*", с. 142.

получения информации в нужном порядке. По мере осуществления цикла и ответственные, и плановые даты могут меняться для некоторых тестовых сценариев, например, когда Джамал принял на себя ответственность за тесты безопасности и секретности. В таких ситуациях ведущий тестировщик или тест-менеджер должен обсудить изменения ответственности и плановых дат с группой тестирования.

Здесь может возникать путаница с двумя серьезными последствиями. Неверное управление драгоценными ресурсами (тестировщиками и запланированным временем), доверенными тест-менеджеру, является дурным тоном. Часто тест- менеджеры требуют еще больше сверхурочной работы от своей команды, чтобы спасти сроки. А это приводит к потере авторитета и к снижению возможностей руководителя добиться наилучшей работы от своей группы[1].

Это указывает на то, что группа тестирования должна иметь возможность приспосабливать процесс и выполнение запланированных тестов к меняющимся обстоятельствам. Часть тестирования состоит в обнаружении непредсказуемого поведения и в исследовании этих аномалий для того, чтобы подготовить подробные и работающие отчеты об ошибках (см. главу 14). Поскольку поведение является непредсказуемым и поскольку мы не можем со 100%-ной точностью прогнозировать, сколько ошибок мы обнаружим, процесс тестирования порождает свой собственный поток событий. Кроме того, внешние события, например задержка предоставления версий, уточнение состава сборок, изменения в приоритетах руководства, вносят изменения в ход процесса. Тестировщики не могут предотвратить этих изменений, поэтому процесс тестирования должен аккумулировать их. В нашем примере Джамал приспосабливался к изменениям как внутренним — неправильному обмену информацией о результатах тестирования, так и внешним — высокому уровню содержания дефектов в SpeedyWriter и неожиданной остановке тестового сервера системным администратором. Джамал пересмотрел ответственных за выполнение тестовых сценариев, добавил дополнительные ресурсы, изменил плановые даты завершения тестов и перенес выполнение тестовых сценариев низшего приоритета на следующий цикл тестирования.

В качестве отступления рассмотрим бесполезную трату времени и задержки, возникшие в нашем примере, когда системный администратор, не входящий в группу тестирования, остановил сервер, который Луис использовал для выполнения теста. Тему управления конфигурацией среды тестирования я подробно рассматривал в моей первой книге "*Managing the Testing Process*", но я бы хотел отметить, говоря о выполнении тестов, что группа тестирования должна иметь полный контроль над тестовой средой в период проведения тестирования.

Серьезное нарушение процесса возникает тогда, когда ошибка, описанная в главном примере книги, происходит более чем один-два раза в ходе тестирования. В ситуациях, аналогичных сложившейся при тести-

---

[1] Неуважительное отношение ко времени сотрудников группы — смертный грех руководства. Однажды я стал свидетелем ухода ведущего системного архитектора проекта после таких событий с позорной, внезапной и эмоциональной вспышкой гнева в середине совещания. Более глубокое рассмотрение этой и других ошибок руководства можно найти в книге Тома де Марко и Тима Листера "*Peopleware*».

ровании проекта "Суматра», когда внешние по отношению к группе тестирования сотрудники имеют права администратора на тестовую среду, поскольку они занимаются ее поддержкой, трудно гарантировать, что никто не изменяет тестовую среду без санкции тест-менеджера или ведущего тестировщика.

Бывает так, что люди, ответственные за поддержку тестовой среды, создают больше проблем, чем оказывают помощь. В моей практике были случаи, когда администраторы забирали оборудование из тестовой лаборатории, хотя оно приобреталось исключительно для тестирования. При таком отношении лучше попытаться перенести поддержку тестовой среды в группу тестирования, даже если для этого придется уменьшить объем тестирования, и закрыть логический (электронный) и физический доступ к тестовому оборудованию для всех сотрудников, не входящих в группу тестирования. Это могут быть секретные пароли, брандмауэры, замки, отдельные помещения для тестовой лаборатории, карточные ключи и т.п. В противном случае эффективность тестирования может снизиться до нуля.

Наконец, эффективный процесс тестирования должен иметь минимально возможные накладные расходы. Другими словами, мы должны посвятить по возможности большую часть времени тестированию и другим действиям, значимым для компании. Заметим, что ежедневные совещания, посвященные итогам тестирования, перекрестное чтение отчетов об ошибках и другие неформальные действия тестирования, стимулирующие обмен информацией и знаниями, не являются накладными расходами, поскольку способствуют повышению эффективности тестирования. Однако следование неуместно сложному или трудоемкому процессу, использование процессов, отягощенных подготовкой большого количества документов, которые не будут использоваться впоследствии, проведение продолжительных совещаний, когда один человек докладывает руководителю о состоянии дел, в то время как другие просто ожидают своей очереди, – все это примеры накладных расходов.

Решение проблем

Поскольку процесс выполнения тестов подвергается всякого рода непредсказуемым и внешним воздействиям, в нем возникают проблемы. Рассмотрим некоторые из них.

Работа с дополнительными сменами и субподрядчиками, проводящими тестирование

В главе 6 мы видели, как Джамал решил использовать вечернюю смену. Он принял это решение, основываясь на ограничениях тестовой среды. Это обычная причина, по которой используются вечерняя или даже ночная смена и выходные. Я предпочитаю вечерние смены и работу в выходные, если чувствую недостаток в редких прототипах Интернет-приложений и надо выполнить быстро большое количество тестов. Если тестирование должно проводиться в производственной системе, уста-

новленной на универсальной машине, которая просто не может быть полностью отдана на растерзание тестировщикам в обычные рабочие часы, большую часть тестирования приходится проводить в нерабочее время. Другой случай — проведение энергичного тестирования перегрузок 24 часа в день. Это означает, что нам приходится все время быть рядом с тестами. Когда я был менеджером проекта по созданию тестовой лаборатории, мы обычно работали и в вечернюю смену, и в выходные.

Дополнительными сменами не так уж трудно управлять, поскольку вы в состоянии держать в голове несколько объектов. Основной вопрос — это коммуникация. Людям, которые не находятся в одно время в одном месте, труднее общаться, а выполнение тестов (наряду с подготовкой отчетов о дефектах) — это определенно та часть процесса тестирования, для которой взаимодействие сотрудников группы тестирования имеет ключевое значение. В нашем примере Луис и Эмма столкнулись с проблемой коммуникации, работая в одной смене, что показывает, как легко возникают такого рода проблемы. Вечерние смены ограничивают возможности коммуникации, а ночные смены и работа в выходные снижают эти возможности еще больше. Группа тестирования должна предпринять все возможные усилия для улучшения коммуникации, а тест-менеджер обязан внедрить соответствующие процессы. В проекте с Интернет-приложением у нас ежедневно были двухчасовые перекрытия с 15 до 17 часов. В это время вечерняя смена читала электронные сообщения, поступившие за день, и задавала вопросы до того, как дневная смена заканчивала работу.

Дополнительные смены (Extended Shifts)

Любая работа, выполняемая сотрудниками вне основного рабочего времени (примерно с 8.00 до 17.00). Обычно это *вечерняя смена* (примерно с 16.00 до 1.00), *ночная смена* (примерно с 0.00 до 9.00), *работа в выходные* и *праздничные дни* (любая из указанных смен, но не в рабочие дни). Сравните с переработкой (overtime) — работа свыше 40 часов в неделю, может включать в себя работу в любые смены.

Часть общения состоит в сборе информации о состоянии дел. Полезны ежедневные совещания, посвященные ходу тестирования и происходящие в период передачи смен. На этих совещаниях мы говорим о тестировании, которое состоялось днем и предыдущим вечером, а также обновляем таблицы отслеживания тестовых сценариев.

Еще одна проблема — планирование. Все хотят знать, в какой смене и в какие дни они работают. Самый простой случай, когда сотрудники работают в одну и ту же смену в течение всего проекта. Однако это не всегда возможно. Некоторые тестировщики считают работу в дополнительные смены слишком обременительной. И поэтому может быть необходим некоторый способ ротации смен. Мой опыт показывает, что премии, вы-

плачиваемые бригадам, работающим в вечерние смены и в выходные, обычно делают дополнительные смены желательными для определенного круга людей, что уменьшает проблему.

Другой вопрос – поддержка. Если группа тестирования не администрирует сеть тестирования, необходимо подыскать системного администратора или сотрудника поддержки сетевых операций, который обеспечит поддержку во внерабочее время. Этот сотрудник должен предоставить группе тестирования номера пейджера, домашнего и сотового телефонов и т.п. Он должен отвечать на звонки днем и ночью и в целом быть готов к тому, что поздним вечером или даже ранним утром его сон может быть прерван срочным запросом группы тестирования. Я считаю решение этой проблемы более сложным делом, нежели подбор тестировщиков для работы в дополнительные смены. В главном примере книги Джамал очень осторожно решал эту проблему в ходе планирования тестирования. Если вы не решите ее до начала выполнения тестов, то рискуете столкнуться с негодованием, деструктивным формальным согласием и другими формами активного и пассивного сопротивления, включая прямой отказ.

Наконец, рассмотрим вопросы материально-технического обеспечения. Сотрудники, работающие в дополнительные смены, должны иметь соответствующие пропуски. Во время дополнительных смен должны работать средства кондиционирования воздуха. Должны быть предоставлены коды сигналов тревоги. Служба безопасности должна быть оповещена. Внутренняя телефонная сеть должна быть сконфигурирована для использования во внерабочие часы. Администрация зданий должна быть заинтересована в решении возникающих вопросов.

Передача всего тестирования или его части субподрядчику (аутсорсинг) обычно порождает те же проблемы, что и дополнительные рабочие смены, только в большей степени. В этих случаях общение обычно идет по электронной почте и телефону. Время от времени случаются визиты друг к другу, но редко бывает, что все сотрудники тестовой лаборатории приезжают к клиенту и наоборот. Кроме того, визиты друг к другу посвящаются в основном презентациям руководству и обсуждению вопросов заключения и оплаты контракта, а не тактическому взаимодействию между непосредственными исполнителями. Общение по электронной почте и телефону осложняется разными часовыми поясами, что ограничивает периоды одновременной работы и возможности диалога в реальном времени, в интерактивном режиме по мере надобности. Персонал поддержки тестирования, например системные администраторы, обычно не отвечают за обеспечение поддержки тестовой лаборатории субподрядчика.

Из-за широкой клиентской базы большинство компаний, выполняющих тестирование, не берутся за проведение частной экспертизы, необходимой для поддержки системы. В ходе тестирования возникают как небольшие, так и серьезные затруднения при инсталляции, сопровождении или работе с системой, в то же время соответствующий специалист внутри группы тестирования может решить их за минуты. Такие затруднения могут выводить систему из рабочего состояния на часы и даже дни в тестовой лаборатории субподрядчика. Я наблюдал это в целом ряде проектов.

Я не против независимых лабораторий тестирования и консультантов. Во всяком случае, я один из них. Я не против дополнительных смен. Сам пользуюсь ими. Однако важно планировать тестирование с учетом того, что будут обнаружены проблемы. План тестирования, который предполагает, что все тесты пройдут без ошибок и что система будет хорошо работать, обречен на провал. Если вы планируете использование дополнительных смен и аутсорсинг тестирования с пониманием необходимости решать проблемы взаимодействия, поддержки и логистики, которые возникают из-за сложного или полного ошибок продукта, эти варианты будут успешными. Кроме того, группа тестирования, которая увеличивает эффективность и производительность за счет использования дополнительных смен и субподрядных групп тестирования, будет пожинать хороший политический урожай за счет умения проводить более качественное тестирование в более короткие сроки. Доверие к ней будет расти.

Учет праздников, отпусков и культурных традиций

Периоды выполнения тестов — время наибольшего напряжения во многих проектах — часто захватывают праздники. В самом деле, в разнообразном смешении культур, присутствующем во многих компаниях всего мира, с большим трудом можно найти период, который не попадает на праздник для одного или нескольких сотрудников в проекте.

Как продемонстрировал Джамал в нашем примере, культурное различие внутри группы на самом деле является не слабостью, а преимуществом. В однородных проектах все хотят отдыхать в период одних и тех же праздников. В проектах с различием культур найдутся такие сотрудники, которые будут рады обменять, например, рождественские каникулы на их собственные культурные или религиозные праздники.

Отпуска, если они не отгуливаются всеми одновременно, также представляют проблему с точки зрения вынужденного простоя, особенно для ключевых участников. Это может стать проблемой для короткого или очень рискованного проекта, и высшее руководство обычно отменяет все отпуска на этот период времени. Однако обычно кто-то берет отпуск в период длинного проекта. На самом деле, выбор невелик: либо предоставлять отпуска, либо люди перестанут работать и начнут увольняться.

Какой бы ни была причина отсутствия персонала, основным принципом является заблаговременное предупреждение. Я считаю, что это можно спланировать, зная об отпусках, культурных традициях и праздниках. В ходе проведения тестирования важно помнить, в какие недели будет работать сокращенное число сотрудников. Тест-менеджер может перенести часть тестирования на другие недели.

Сбор данных о проекте тестирования на месте

Сбор данных о проекте является особенно критичным, если есть внешние зависимости. Если вы не сможете завершить плановое тестирование, поскольку не работает тестовая среда, в этом, возможно, не ваша вина. Вероятно, группа операций, ответственная за поддержку тестовой среды,

не отвечала на пейджинговые и телефонные запросы о помощи. Возможно, конфликт приоритетов не позволил им отреагировать на запросы так, как оговорено в плане тестирования. В этом случае руководство должно понимать, какое влияние конфликт приоритетов оказывает на тестирование.

Конечно, может быть так, что внешние группы, от которых вы зависите, не заинтересованы в успехе тестирования или проекта в целом. По очевидным политическим причинам хранение точных протоколов в таких ситуациях – вопрос выживания. Обсуждение черной магии корпоративной политики находится за рамками этой книги. Скажу лишь, что для выхода из такого рода ситуаций понадобится адекватная документация.

Внедрение усовершенствований

Хотя проведение тестирования тесно связано с контекстом тестирования и остального проекта, существует несколько идей, которые можно использовать во многих случаях для более успешного проведения тестирования.

1. Оцените, где вы находитесь и какие части процесса выполнения тестов, включая другие процессы, рассмотренные в этой книге, не находятся под вашим контролем. Вероятно, некоторые из источников хаоса являются внешними. Можете ли вы уменьшить влияние внешнего беспорядка на проект тестирования?

2. Убедитесь в том, что вы знаете, как нужно проводить тестирование и сколько времени потребуется для выполнения выбранных тестов, разбитых на небольшие фрагменты. Двух-четырехчасовые задачи – это размер, которым, по моему мнению, удобно управлять. Они не обязательно должны принимать форму подробных тестовых сценариев, записанных в виде скриптов, когда можно оценить с точностью до минуты, сколько требуется в среднем на выполнение каждого детально описанного непротиворечивого шага. Вполне достаточно увязать план тестирования (например, тестирование возможностей манипуляции файлами) с планируемой продолжительностью (например, два часа на проверку возможностей манипуляции файлами).

3. Используйте какой-либо способ отслеживания тестовых сценариев: таблицу тестовых сценариев, таблицу комплектов тестов (см. выше) или какой-то другой механизм, – с учетом как продолжительности и сроков, так и результатов тестирования. Повторюсь, что вы можете начать с чего-то очень простого, и, возможно, этого будет вполне достаточно.

4. Используйте информацию о затратах на тестирование, которую вы собираете в процессе отслеживания тестовых сценариев, а также ваши прогнозы по поводу обнаружения ошибок в различные моменты фазы тестирования, чтобы предсказать, сколько времени нужно выделить на обработку ошибок. Классической ловушкой тестирования считается неспособность предвидеть наличие ошибок

в системе, предназначенной для тестирования. В результате планирования тестов так, будто они все успешно будут пройдены, обычно недооцениваются затраты и время[1].

5. Решайте вопросы, связанные с необходимостью внутреннего взаимодействия и сбора информации. Многие проблемы тестирования возникают из-за того, что тестировщики не знают, что делают рядом их коллеги.

Все эти шаги могут стать важными вехами на пути постановки под ваш контроль проведения тестирования.

Компетентные тестировщики выполняют тесты дисциплинированно и с воображением, а компетентный тест-менеджер решительно и твердо управляет процессом выполнения тестов. Группа тестирования приспосабливается к неожиданным событиям и результатам тестирования и отвечает на них, а также последовательно способствует росту стоимости компании. Во время выполнения тестов группа тестирования освещает путь проекту.

Мы заканчиваем обсуждение фазы "Проведение" в процессе тестирования. Умелое проведение тестирования создает актуальную, полную и точную информацию о качестве системы. Это услуга и продукт, которые мы можем передать разработчикам, коллегам-менеджерам, менеджерам проекта и даже высшему руководству.

Передача этого продукта и оказание услуги дает возможность рабочей группе проекта совершенствовать систему. Кроме того, полученная информация позволяет нам совершенствовать систему тестов. В заключительных главах книги мы поговорим о подготовке отчетов о результатах тестирования, об управлении изменениями системы и, разумеется, об улучшение самого тестирования.

№ этапа	Этап	Выполнено?
3.A	Проведение: выполнение тестов и сбор результатов	☑
3.D	Получение и установка тестовой версии, состоящей из всех или нескольких компонентов системы, предназначенной для тестирования.	☑
3.F	Назначение ответственных за выполнение комплекта тестов на каждой из тестовых версий, отслеживание комплекта и управление им.	☑

[1] Привычка планирования тестов с предположением, что они не пройдут успешно, настолько прочно укоренилась в моем сознании, что я забыл упомянуть об этом в самом начале. Я благодарен Тиму Коомену за то, что он указал на это упущение.

Часть IV

Совершенствование

По завершении проекта я ни разу не мог сказать: "Мы провели совершенное тестирование". Но в каждом проекте я мог работать над совершенствованием тестирования, проводить тестирование лучше, чем в предыдущих проектах. Совершенство — это отличный стандарт, по которому измеряются все наши достижения, поскольку мы намереваемся двигаться по направлению к совершенству, непрерывно становясь лучше.

В заключительной части книги мы исследуем процессы управления совершенствованием разрабатываемой системы, а также самого процесса тестирования. Это происходит, помимо всего прочего, при помощи эффективного обмена информацией, в частности при помощи подготовки отчетов об ошибках и результатах тестирования. Мы можем использовать эту и другую информацию о процессах тестирования, чтобы сделать разумные изменения.

В последней главе мы вновь рассмотрим весь процесс тестирования. Мы вернемся к большой картине, которую изучали в главе 1. Это поможет нам найти общие черты в качественных процессах тестирования и изучить способы решения основных проблем. Наконец, поскольку моя цель при написании этой книги и ваша цель при ее чтении — найти способы улучшения процессов тестирования, мы завершим обсуждение этой темой.

№ этапа	Этап	Выполнено?
4.	Совершенствование: предоставление информации о проекте и его улучшение.	☐
4.A	Документирование ошибок, обнаруженных в ходе выполнения тестов.	☐
4.B	Обсуждение результатов тестирования с ключевыми заинтересованными лицами.	☐
4.C	Управление изменениями и уточнение процесса тестирования.	☐

ГЛАВА 14

Когда качество не оправдывает ожиданий: подготовка отчетов об ошибках

В этой главе рассматривается процесс составления отчетов об ошибках. В известном смысле отчеты об ошибках — это основной материальный продукт тестирования, надежная техническая документация, которая описывает вид ошибки в тестируемой системе, который может повлиять на мнение пользователя о системе[1]. Каждый отчет об ошибке должен предоставлять руководству информацию, необходимую для принятия решения по приоритету ошибки. Для тех ошибок, которые считаются заслуживающими исправления, каждый отчет об ошибке должен предоставлять программисту всю необходимую для исправления этой ошибки информацию.

Подготовка отчетов — это процесс тестирования, удовлетворяющий всем четырем критериям, которые я сформулировал во введении, когда отбирал ключевые процессы. Это процесс, в который мы вовлечены в ходе выполнения тестов каждый день, а иногда много раз в день. Отчеты об ошибках читают программисты, а также руководители разных уровней, вплоть до высшего. Если процесс подготовки отчетов об ошибках осуществляется правильно и сами отчеты готовятся верно, в результате внутри группы тестирования организуется командная работа, а также возрастает доверие к группе тестирования среди сотрудников рабочей группы проекта. Хорошо подготовленные отчеты об ошибках, распределенные по приоритетам руководством и исправленные программистами, являются движущей силой, при помощи которой группа тестирования оказывает влияние на рост качества системы, в то время как плохие отче-

[1] В то же время, как напомнил мне Рик Крейг, прочитав эту главу, в некоторых случаях "сами тестовые сценарии могут стать частью документации программного продукта. По мере того как система стареет и спецификации требований и/или архитектуры [или сами требования и/или архитектура] устаревают, возрастает вероятность того, что документация, описывающая, что система обязана делать, содержится только в тестовых сценариях".

Отчет об ошибке (Bug Report)

Технический документ, описывающий симптомы ошибки для 1) информирования о значении проблемы и обстоятельствах ее возникновения; 2) определения приоритета ошибки для исправления; 3) помощи программисту в локализации дефекта и его исправлении.

ты об ошибках могут привести к тому, что опасные дефекты останутся в системе, переданной в эксплуатацию. Каждый тестировщик в группе тестирования должен владеть этим процессом, чтобы вносить максимально возможный вклад в разработку[1].

Процесс подготовки отчетов об ошибках

Как составлять наилучшие отчеты об ошибках? Процесс 10 представляет собой процесс подготовки отчетов об ошибках, которые я применяю в большинстве своих проектов[2].

Процесс 10. Процесс подготовки отчетов об ошибках

№ этапа	Этап	Выполнено?
1.	Выполнить каждый тест, используя надежные структурированные методы тестирования.	☐
2.	Воспроизвести любые отклонения от ожидаемого поведения.	☐
3.	Локализовать ошибку, проверив действие основных вариантов, а также исследуя возможные обходные пути.	☐
4.	Найти, по возможности, общий случай, наиболее серьезный или наименее желательный.	☐
5.	Сравнить наблюдаемое неправильное поведение с поведением системы при похожих условиях или в предыдущих тестовых версиях.	☐
6.	Обобщить сведения об отказе, включая его воздействие на заказчиков и пользователей.	☐
7.	Сократить отчет, удалив ненужную информацию.	☐

[1] Части этой главы были ранее опубликованы в "*Professional Tester*". Я благодарен Кэролайн Квентин (Caroline Quentin) за ее отличную работу по редактированию той статьи. Некоторые фрагменты этой главы также были напечатаны в книге Эрика Ван Веенендааля (Erik van Veenendaal) "*The Testing Practitioner*". Я благодарю его за включение этого важного процесса в качестве части признаваемого отраслью подхода к обработке ошибок в программах.

[2] Я извлек пользу из онлайновой дискуссии с Бретом Петтикордом, состоявшейся в 1999 году, по поводу этой таблицы. Я благодарю его за вклад в мои рассуждения на эту тему.

Процесс 10 (*продолжение*)

№ этапа	Этап	Выполнено?
8.	Устранить противоречия, удалив сбивающие с толку, вводящие в заблуждение или неточные слова.	☐
9.	Подготовить нейтральный отчет об ошибке, выражая мысли беспристрастно и спокойно.	☐
10.	Передать отчет об ошибке на перекрестное рецензирование, разрешить проблемы, если они обнаружены, и затем ввести отчет в систему управления дефектами.	☐

Поскольку этот процесс критически важен, поясню влияние каждого из этапов на проект.

1. *Структура.* Подготовка полезного отчета об ошибке начинается с надежного, хорошо организованного тестирования. Тестирование может быть ручным и автоматизированным, вестись по плану или быть исследовательским, но оно должно отличаться от простых хакерских бесцельных попыток сломать продукт. Небрежное тестирование приводит к небрежным отчетам об ошибках.
2. *Воспроизведение.* Ошибки, конечно, часто застают меня врасплох! Если бы я мог предугадать заранее, когда я найду ошибку, мне не нужно было бы тестировать систему. Воспроизведение проблемы обостряет мое понимание вопроса и позволяет жестко зафиксировать последовательность шагов, которые заново приведут к отказу. В качестве эмпирического правила я предпочитаю делать три попытки. Время от времени появляются ошибки, которые не всегда воспроизводятся и с большим трудом поддаются исправлению. Я информирую о таких ошибках с пометкой об их нерегулярности, например: "эта ошибка проявилась в двух из трех попыток"[1]
3. *Локализация.* На поведение программы могут влиять многие факторы, а некоторые из этих факторов воздействуют на симптомы ошибки. Часто большое значение имеет способ, при помощи которого изменение в тестовой среде обнаруживает само себя с точки зрения поведения ошибки. Изменяя некоторые важные переменные, каждый раз по одной, в попытке изменить поведение ошибки, я вижу, что могу написать более точный отчет об ошибке. Локализация ошибки -- непростая задача. До некоторой степени я должен понять, как работает система, предназначенная для тестирования. Я должен продумывать тесты, прежде чем выполнять их. Я также избегаю погружения в отладку и трат неразумного количества времени на тривиальные случаи. Однако точная локализация ошибки способствует росту доверия тестировщику и дает программисту преимущество при отладке. Кроме того, я могу найти доступный и полезный обходной путь как часть локализации ошибки (см. главу 3).

---

[1] Стив Сплэйн, один из рецензентов этой книги, рассказал мне, что при обнаружении нерегулярно возникающей ошибки он "обычно включает дополнительную информацию о конфигурации [системы]... поскольку есть вероятность, что причины [ошибки] связаны с несовместимостью...

4. *Обобщение.* Часто симптом, который я вижу при первом обнаружении ошибки, не является наиболее общим случаем. Например, недавно я нашел проблему, при которой программа не импортировала особый вид таблицы из файла в формате Excel. Когда я копнул чуть глубже, я понял, что невозможно импортировать таблицу из *любого* файла в формате Excel, если имя таблицы содержит круглые скобки. Важно при поиске общего случая не доходить до абсурда. Не все отказы программы происходят из-за одной и той же ошибки. Тем не менее, попытка найти общее правило возникновения ошибки углубляет мое представление о системе и позволяет убедить программистов и менеджеров в том, что проблема не является единичным случаем.
5. *Сравнение.* Тестирование часто покрывает одну и ту же область, и когда тестировщик выполняет одни и те же тесты в разных версиях системы, и когда тестировщик выполняет тесты, чтобы покрыть похожие условия. Полученные результаты могут дать программисту разгадку, если мне удастся соединить концы с концами. Является ли отказ регрессией? Работает ли то же свойство в других частях продукта? Хотя такой способ исследования не всегда возможен, например, если тест был заблокирован в предыдущей версии, он позволяет программисту сэкономить массу времени.
6. *Краткое описание.* В этот момент я, по всей вероятности, понимаю проблему в программе достаточно хорошо, чтобы увидеть, как она повлияет на заказчиков или пользователей. В качестве части процессов управления изменениями и разбора дефектов (см. главу 16) важно рассмотреть бизнес-риски, относящиеся к каждой ошибке, как элемент принятия решения об обязательности их исправления. Полезный отчет об ошибке включает в себя краткое описание, которое показывает суть и значение проблемы группе по управлению изменениями или комиссии по разбору ошибок. (Поле для краткого описания может носить названия "Заголовок", "Краткое содержание", "Описание воздействия" и т.п., но смысл в том, что это краткое, в одном предложении, выражение содержания всего отчета.) Краткое описание становится неоценимым, когда наступает время расстановки приоритетов ошибок. Оно также служит названием ошибки для программистов. "Ты уже исправил ошибку со шрифтами? – вопрос, который вы можете задать программисту, будет, наверное, более понятным, чем вопрос "Как продвигается работа над ошибкой 1716?" Подготовка полезного краткого описания гораздо труднее, чем кажется. Математик Блэз Паскаль однажды завершил длинное письмо фразой: "Я написал бы более короткое письмо, но у меня на него не было времени". Я потратил много времени на подготовку полезных кратких описаний, поскольку это наиболее важные предложения в отчетах об ошибках.
7. *Уменьшение объема.* К слову о написании более коротких писем, подумайте, нельзя ли удалить некоторые ненужные слова из отчетов об ошибках? Однажды я читал отчет об ошибке, первая страница которого была посвящена описанию того, как тестировщик набрал много очков при игре в тетрис, что не имело ничего общего с пробле-

мой. Разве меня волновало это достижение тестировщика? Нет. Должен ли я был тратить время на чтение этого? Нет. Вызвал ли тестировщик, написавший этот отчет, внимание к его последующим отчетам? Тоже нет. Лучшие отчеты об ошибках не являются ни загадочными комментариями, ни занудными, бесконечными и бесцельными документами. Я стараюсь пользоваться только теми словами, которые мне нужны, описываю только последовательность действий и даю информацию, действительно имеющую отношение к ошибке, по которой готовится отчет.

8. *Устранение противоречий.* Устранить противоречия означает удалить сбивающие с толку или вводящие в заблуждение слова, которые могут озадачить читателя и не позволят ему понять, что имеется в виду. Идеальный отчет об ошибке подводит программиста к ошибке за руку. Например, что подразумевается, если написано: "Программа прекратила работу". Она перестала отвечать, но осталась на экране? Удалось ли штатно выйти из нее? Или она завершилась внештатно? Она завершилась и закрыла одну или несколько программ? Она завершилась и вызвала фатальную ошибку в операционной системе? Слова "Программа прекратила работу" неоднозначны, подобного не должно быть в отчетах об ошибках. Ясность — вот ключ к решению проблемы[1].

9. *Нейтральность.* Как гонец с плохими новостями, я должен выражать свои мысли в отчете об ошибке спокойно и беспристрастно. Атака на программиста, критика ошибки в коде, юмор или сарказм в отчете об ошибке обычно приводит к бурной ответной реакции. Я стараюсь проявлять деликатность и ясность мысли в описании ошибки и ее последствий, ограничиваю свои отчеты констатацией факта и не занимаюсь предположениями и преувеличениями. Отчеты об ошибках после того, как их занесли в систему управления дефектами, обычно остаются там навсегда и могут быть прочитаны людьми, не участвующими в проекте[2].

10. *Рецензирование.* Как упоминалось ранее, отчет об ошибке — это технический документ. Экспертные оценки, инспекции, быстрые просмотры и т.п. — это мощные инструменты для раннего обнаружения проблем в документах любых видов, включая отчеты об ошибках. Если вы хотите пропустить этот этап как слишком затратный по времени или как ненужный контрольный пункт, чтобы побыстрее ввести отчет об ошибке в систему управления дефектами, подумайте о задержках и возможном срыве планов, которые могут произойти, если программист потратит много времени на воспроизведе-

---

[1] Я получил немалую пользу от некоторых комментариев Джоанны Ротман по поводу устранения противоречий, благодарю ее за содержательные замечания.

[2] Это похоже на то, о чем писал Сем Канер в "*Тестировании программного обеспечения*". Никогда неизвестно, кто будет читать ваши отчеты об ошибках (возможно, адвокат истца?), поэтому следует избегать ненужных слов, которые могут стать свидетельством для наказания вашего работодателя, нужно писать только в точности то, что вы имеете в виду. Однажды я действительно был вынужден работать вместе с адвокатом, когда имела место попытка на основе анализа отчетов об ошибках выдвинуть обвинения против группы разработки.

ние ошибки, которая на самом деле является ошибкой в тесте, или если программист не может понять, о чем этот отчет, и возвращает ошибку как "невоспроизводимую" или "работающую, как спроектировано". Рецензирование не должно быть формальным. Я добивался успеха, когда одним из требований к рецензированию было прочтение одним тестировщиком отчетов другого перед отправкой отчета об ошибке в систему управления дефектами.

Позвольте мне указать на два общих наблюдения по поводу процесса перед тем, как мы пойдем дальше. Во-первых, важно помнить, что написание отчетов является творчеством: два полезных отчета об ошибке по одной проблеме могут отличаться и по стилю, и по содержанию. Цель процесса подготовки отчетов об ошибках – не стандартизовать сами отчеты, а обеспечить их качество. Во- вторых, этот процесс работает наилучшим образом, когда он воспринимается как список контрольных вопросов, а не строгий последовательный процесс. Первый и последний этапы, структурированное тестирование и перекрестный просмотр, являются логическими точками начала и завершения, но если вы чувствуете себя более комфортно, когда ваш текст нейтрален, делайте именно так.

Неприятная ошибка испортила содержимое нового файла, который я создал посредством форматирования некоторого текста шрифтом Symbol. В результате я зря потратил время.

Большая ошибка в большой сборке

В понедельник 16 декабря Луис тестировал большую сборку для проверки функциональности управляемых файлов. Он создал новый файл и ввел текст. Затем он попытался изменить шрифт текста: выделил текст, вызвал меню шрифтов и выбрал шрифт Symbol. Неожиданно там, где только что было нормальное предложение на английском языке, появился набор символов, вопросительных знаков и других бессмысленных значков. "Хм, – задумался Луис, – это явно не из категории "приемлемого поведения". Он начал подготовку отчета об ошибке, сделав пометки в своем блокноте.

№ этапа	Этап	Выполнено?
1.	Выполнить каждый тест, используя надежные структурированные методы тестирования.	☑

Возможно, так Луис описал бы проблему в разговоре с тест-менеджером Джамалом Брауном в тестовой лаборатории, но он не мог предста-

вить ее в таком виде программисту, менеджеру разработки или менеджеру проекта. Учитывая этих потенциальных читателей отчета, Луис вернулся к системе и воспроизвел проблему еще два раза. Затем он начал вводить отчет в базу управления дефектами.

№ этапа	Этап	Выполнено?
2.	Воспроизвести любые отклонения от ожидаемого поведения.	☑

Шаги для воспроизведения.

1. Я запустил редактор SpeedyWriter и создал новый файл.
2. Затем я напечатал четыре строки текста, повторяя фразу "Быстрая лисица перепрыгивает через ленивую коричневую собаку", причем каждый раз использовал различные эффекты: полужирный, курсив, перечеркнутый и подчеркнутый шрифты.
3. Я выделил текст, затем открыл меню шрифтов и выбрал Symbol.
4. Неприятная ошибка превратила весь текст в ничего не значащий мусор, в результате было напрасно потрачено время пользователя.
5. Я воспроизвел проблему трижды из трех попыток.

Локализация ошибки

Подозревая, что это просто проблема форматирования, я сохранил файл, закрыл SpeedyWriter и открыл файл заново. Сохранился мусор вместо текста.

Если вы сохраните файл до изменения шрифта на Symbol, ошибка не проявится.

Ошибка не проявляется в существующих файлах.

Она проявляется только при работе в операционной системе Windows 98 OSR2.

Описание проблемы стало выглядеть лучше. Программист мог прочитать и быстро приступить к отладке проблемы. Однако Луис знал, что продукт SpeedyWriter, как и любое другое приложение, поддерживает широкий диапазон свойств и конфигураций, которые могут повлиять на поведение этой ошибки. Поэтому Луис провел несколько экспериментов в поисках обходных путей. Затем он добавил следующий раздел в отчет.

№ этапа	Этап	Выполнено?
3.	Локализовать ошибку, проверив действие основных вариантов, а также исследуя возможные обходные пути.	☑

Это был особый вид ошибки, поэтому Луис решил проверить и несколько других шрифтов и выяснить, оказывают ли какое-то влияние шрифты, стили шрифтов и определенный текст на ошибку. Он обнару-

жил, что шрифты Wingdings и Arial также вызывают проблему. Поскольку пользователи SpeedyWriter активно применяют шрифты Arial, последнее было существенным. Однако эксперименты с полужирным, курсивом и подчеркнутым вариантами шрифтов не повлияли на проявление ошибки. Луис выяснил, что определенные строки текста и количество текста никак не влияют на ошибку. Он удалил потенциально вводящую в заблуждение ссылку на стили шрифтов и текстовую строку из шага 2 отчета и изменил отчет следующим образом (см. части, выделенные курсивом), чтобы дополнительно обозначить связь ошибки со шрифтом Arial (и исправить грамматическую ошибку).

№ этапа	Этап	Выполнено?
4.	Найти, по возможности, общий случай, наиболее серьезный или наименее желательный.	☑

Далее Луис просмотрел протоколы тестирования предыдущих тестовых циклов комплексного и системного тестирования и изучил некоторые отчеты о тестировании, подготовленные программистами на фазе компонентного тестирования. Он обнаружил, что именно это свойство проверялось, что натолкнуло его на мысль, что эта проблема является ошибкой регрессии в качестве системы. Он добавил соответствующую строку в раздел "Локализация ошибки".

Шаги для воспроизведения.

1. Я запустил редактор SpeedyWriter и создал новый файл.
2. Затем я напечатал текст: четыре строки, повторяющие каждый раз фразу "Быстрая лисица перепрыгивает через ленивую коричневую собаку".
3. Я выделил текст, затем открыл меню шрифтов и выбрал *Arial*.
4. Неприятная ошибка превратила весь текст в ничего не значащий мусор, в результате было напрасно потрачено время пользователя.
5. Я воспроизвел проблему трижды из трех попыток.

Локализация ошибки

То же самое происходит со шрифтами Wingdings и Symbol.

Подозревая, что это просто проблема форматирования, я сохранил файл, закрыл SpeedyWriter и открыл файл заново. Сохранился мусор вместо текста.

Если вы сохраните файл до *изменения шрифта* текста, ошибка не проявится.

Ошибка не проявляется в существующих файлах.

Она проявляется только при работе в операционной системе Windows 98 OSR2.

Новая ошибка для большой сборки (3.1.018.TR), тот же тестовый сценарий прошел без ошибки в сборках 3.1.007.TR и 3.1.017.TR.

№ этапа	Этап	Выполнено?
5.	Сравнить наблюдаемое неправильное поведение с поведением системы при похожих условиях или в предыдущих тестовых версиях.	☑

Теперь Луис четко понимал суть ошибки, как она себя будет проявлять и как она повлияет на пользователя. Он вписал в отчет следующую строку с кратким описанием ошибки.

№ этапа	Этап	Выполнено?
6.	Обобщить сведения об отказе, включая его воздействие на заказчиков и пользователей.	☑

Шрифты Arial, Wingdings и Symbol повреждают новые файлы.

Итак, Луис зафиксировал влияние ошибки на пользователя и основную природу проблемы в восьми словах. Он уверен, что любой из прочитавших его краткое описание ошибки поймет, в чем проблема и почему она важна.

Отчет об ошибке подготовлен. Все, что осталось, — потратить немного времени на наведение глянца. Сначала Луис немного ужал отчет, удалив лишние слова. Например, предложение:

Если вы сохраните файл до изменения шрифта текста, ошибка не проявится.

он записал так:

Сохранение файла до изменения шрифта препятствует возникновению ошибки.

Он также удалил слово "я" из раздела "Шаги для воспроизведения".

№ этапа	Этап	Выполнено?
7.	Сократить отчет, удалив ненужную информацию.	☑

Далее Луис проверил отчет на наличие противоречивых или запутывающих смысл предложений, которые могут смутить читателя. Он изменил шаг 3 с

3. Выделил текст, затем открыл меню шрифтов и выбрал Arial.

на

3. Выделил *все четыре строки* текста, затем открыл меню шрифтов и выбрал Arial.

Шаг 4 изменился с

4. Неприятная ошибка превратила весь текст в ничего не значащий мусор, в результате было напрасно потрачено время пользователя.

на

4. Неприятная ошибка превратила весь текст в ничего не значащий мусор, *содержащий управляющие символы, числа и куски двоичного кода*, в результате было напрасно потрачено время пользователя.

В разделе "Локализация ошибки" он изменил

То же самое происходит со шрифтами Wingdings и Symbol.

на

Воспроизвел те же шаги, используя шрифты Wingdings и Symbol.

Наконец, в строчку раздела "Локализация ошибки", которая звучала как:

> **Она проявляется только при работе в операционной системе Windows 98 OSR2.**

Луис добавил:

> **Она проявляется при работе в операционной системе Windows 98 OSR2, но *не проявляется при работе в Solaris, Mac и других вариантах Windows.***

№ этапа	Этап	Выполнено?
8.	Устранить противоречия, удалив сбивающие с толку, вводящие в заблуждение или неточные слова.	☑

В завершение Луис освободил отчет от всех ярких суждений и пристрастных комментариев, которые могут встревожить или отвлечь читателей. Он решил, что шаг 4 слишком резкий, поэтому изменил формулировку следующим образом.

> **4. Весь текст превратился в управляющие символы, цифры и другие, вероятно случайные, двоичные данные.**

№ этапа	Этап	Выполнено?
9.	Подготовить нейтральный отчет об ошибке, выражая мысли беспристрастно и спокойно.	☑

Наконец, Луис попросил Лин Цу проверить отчет.

"Выглядит очень неплохо, Луис – сказала Лин Цу. – Хотя один вопрос. Эта ошибка имеет какое-либо отношение к проблеме создания файлов, которую ты нашел в предыдущем тестовом цикле?

"Не думаю, – ответил он, – поскольку эта ошибка имеет отношение к конкретным шрифтам. Предыдущая ошибка была связана с конкретными типами файлов".

"Хорошо, – ответила она. – Только одно предложение. Я бы хотела удалить слово "мусор" из раздела "Локализация ошибок". Оно неудачно".

"Ой, я просто упустил это, – улыбнулся Луис. – Я сначала говорил о мусоре на протяжении всего отчета, но потом удалил его отовсюду, кроме этого раздела. Извини".

"Ничего страшного, для этого мы и просматриваем отчеты друг друга, – успокоила его Лин Цу.

№ этапа	Этап	Выполнено?
10.	Передать отчет об ошибке на перекрестное рецензирование, разрешить проблемы, если они обнаружены, и затем ввести отчет в систему управления дефектами.	☑

Краткое описание

Шрифты Arial, Wingdings и Symbol повреждают новые файлы.

Шаги для воспроизведения.

1. Запустил редактор SpeedyWriter и создал новый файл.
2. Напечатал текст: четыре строки, повторяющие каждый раз фразу "Быстрая лисица перепрыгивает через ленивую коричневую собаку".
3. Выделил все четыре строки текста, затем открыл меню шрифтов и выбрал Arial.
4. Весь текст превратился в управляющие символы, цифры и другие, вероятно случайные, двоичные данные.
5. Воспроизвел проблему трижды из трех попыток.

Локализация ошибки

Новая ошибка для сборки 3.1.018.TR, тот же тестовый сценарий прошел без ошибки во всех сборках, начиная с 3.1.007.TR (начало системного тестирования) и заканчивая 3.1.017.TR.

Воспроизводится с теми же шагами для шрифтов Wingdings и Symbol.

Подозревая, что это просто проблема форматирования, сохранил файл, закрыл SpeedyWriter и открыл файл заново, но повреждение данных осталось.

Сохранение файла до изменения шрифта предотвращает ошибку.

Ошибка не проявляется в существующих файлах.

Она проявляется при работе в операционной системе Windows 98 OSR2, но не проявляется при работе в Solaris, Mac и других вариантах Windows.

Это надежное описание ошибки, обнаруженной Луисом. Оно информирует о проблеме, способствует принятию решения руководством о необходимости исправления ошибки на основе ее влияния на заказчика. Оно также дает группе разработке неплохое руководство для работ по отладке.

За рамками описания дефекта

Помимо описания дефекта, системы управления дефектами должны получать некоторые дополнительные сведения, чтобы весь процесс управления дефектами был работоспособным. Два из них — это серьезность и приоритет.

Под серьезностью я понимаю техническое влияние на систему, предназначенную для тестирования, глубину воздействия ошибок. Я часто использую следующую шкалу для измерения серьезности.

1. *Потеря данных.* Ошибка вызывает потерю данных пользователя (конечного пользователя, оператора и т.п.) или системы.
2. *Потеря функциональности.* Ошибка блокирует использование важной функциональной области (может включать в себя нефункциональные проблемы, в частности проблемы производительности, которые приводят к неприемлемым задержкам в выполнении функций).
3. *Потеря функциональности с наличием обходных путей.* Ошибка блокирует применение основной функциональной области, но для пользователя существует разумный обходной путь.
4. *Частичная потеря функциональности.* Ошибка препятствует использованию некоторой несущественной части функциональной области.
5. *Косметическая ошибка.* Ошибка позволяет функциям нормально работать, но с существенными недостатками (особенно в пользовательском интерфейсе или в способности системы реагировать на запросы)[1]. [номер сноски 7]

Под приоритетом я понимаю значимость устранения проблемы, что в некоторых случаях является сложным бизнес-решением. На него могут воздействовать следующие факторы: влияние ошибки на проект и на вероятный успех продукта на рынке, зависимости между ошибками и/или тестами, требования инструкций, стандартов или контракта, а также серьезность ошибки. Я часто пользуюсь следующей шкалой приоритетов.

1. *Срочно.* Ошибка требует немедленного исправления.
2. *Важно.* Ошибка должна быть исправлена в этой версии.
3. *Значимо.* Ошибка может существенно снизить ценность системы хотя бы для одного из заказчиков или пользователей.
4. *Желательно.* Ошибка должна быть исправлена в этой версии, если это возможно в условиях ограничения свойств, бюджета и сроков, в противном случае — в ближайшей очередной версии.
5. *На усмотрение.* Ошибка может быть исправлена в одной из будущих версий, если позволяют другие приоритеты.

---

[1] Эта шкала позаимствована из стандарта 2167A министерства обороны США для тестирования. Существует большое число других вариантов классификации серьезности. Посмотрите статью Тима Дайеса (Tim Dyes) "Tracking Severity" в *Software Testing and Quality Engineering*, Volume 1, Issue 2 (Mar/Apr 1999), доступную в настоящее время на www.stickyminds.com.

Серьезность и приоритет — отличные друг от друга понятия, причем приоритет является более важным из них. Процесс расстановки приоритетов ошибок обсуждается в главе 16, сейчас же позвольте мне кратко объяснить разницу.

Поверхностный взгляд приводит некоторых инженеров-программистов и системщиков, включая тестировщиков, к выводу, что чем выше серьезность ошибки, тем выше ее приоритет, и наоборот. Это не всегда так. Представим, что ошибка вызывает появления на экране неверного сообщения. Это серьезность 5 по моей шкале. А что, если неверное сообщение содержит оскорбительное, непристойное или расистское слово? Если не драматизировать, то предположим, что на всплывающем экране неверно отображается название продукта. Эти виды проблем являются обязательными для исправления в текущей версии независимо от серьезности.

И наоборот, ошибка потери данных, которая проявляется только при определенных несущественных обстоятельствах в маловероятной и устаревшей конфигурации, может иметь серьезность 1, но приоритет 5 в рамках моего определения. Что, если редактор SpeedyWriter теряет данные, если мы сохраняем их на дискете размером 5.25 дюйма и емкостью 360 Кбайт? Вероятно, что вместо исправления ошибки в программе мы должны просто зафиксировать в документации, что наш продукт не поддерживает такие дисководы[1].

Важной для отчета об ошибке является информация о прошедшей тестирование версии продукта. В нашем примере Луис пишет отчет об ошибке в сборке 3.1.018.TR и замечает, что все сборки от 3.1.007.TR до 3.1.017.TR не имели такой ошибки. Для того чтобы зафиксировать эту информацию, необходимо держать под контролем процесс подготовки тестовых версий (см. главу 12).

В интегрированных системах управления тестированием, когда репозитории ошибок и тестов находятся в одной базе данных, эти два репозитория могут быть связаны друг с другом. Результаты тестовых сценариев могут быть привязаны к определенным отчетам об ошибках, а отчеты об ошибках могут быть сопоставлены тестовым сценариям. Возможность отслеживания полезна как для тестировщиков, так и для программистов. Если база данных управления дефектами не поддерживает этого свойства, я добавляю в процесс ввод номера теста в описание дефекта.

Если говорить о возможности отслеживания, некоторые ошибки могут быть привязаны к определенным требованиям, бизнес-правилам, сценариям использования системы или рискам качества. В главе 15 будет показано, как при помощи отслеживания результатов тестирования SpeedyWriter и отчетов о дефектах тестировщик может говорить о дефектах в терминах оценки качества системы и рисков. Итак, удобные системы управления дефектами обеспечивают поддержку такого отслеживания. Если нет, то я добавляю в процесс еще один этап, который включает в себя ввод тестировщиком этой информации в описание дефекта.

---

[1] В книге "*Software Testing Techniques*" Борис Бейзер приводит интересное обсуждение ошибок и того, что он называет "значимость ошибки".

Поскольку ошибки могут зависеть от конкретной тестовой конфигурации, тестировщики должны выработать привычку фиксировать и это. Привязка к конкретной тестовой конфигурации также может быть либо встроенной в систему управления дефектами (за счет просмотра таблицы конфигураций), либо зафиксирована как часть описания дефекта.

Еще одна часть информации, которую полезно фиксировать, это затронутая подсистема, компонент или функция. Например, для текстового процессора я использую следующие категории:

- Пользовательский интерфейс
- Движок редактирования
- Операции с файлами
- Режимы установки и варианты конфигураций
- Документация и пакетирование
- Прочее
- Неизвестное
- Нет данных

Полезно собирать информацию для различных метрик, включая улучшение текущего и будущего тестирования, чтобы направить усилия на те области продукта, которые демонстрируют наибольшие проблемы.

В типичной системе отслеживания дефектов программист, который исправляет ошибки, и, возможно, инженер сборок, который интегрирует исправленный код в репозиторий кода, будут заполнять ряд полей. Эти поля обычно не оказывают большого влияния на группу тестирования, за исключением информации о том, какая сборка содержит исправление кода, связанное с данной ошибкой. Тестировщики, возможно, сочтут полезными отчеты на основе этой информации при подготовке к проверочному тестированию исправлений ошибок в данной тестовой версии.

Наконец, хочу отметить модели устранения дефектов и данные, которые может собирать база данных управления дефектами для поддержки этих моделей. Модели устранения дефектов позволяют нам отслеживать, когда дефект попал в систему, когда был найден и когда был исправлен. (Обычно эти события называются *вводом дефекта, обнаружением дефекта* и *удалением дефекта* соответственно.) Как тестировщики, мы знаем, когда мы нашли дефект и когда его устранили, но нам нужна помощь программистов для того, чтобы отследить, когда каждый дефект попал в систему.

Я использовал здесь слово "*когда*", но часто модели устранения дефектов заинтересованы не столько в конкретных датах или времени, сколько в фазах проекта. Поэтому сбор данных для типичной модели устранения дефектов может означать, что отчеты об ошибке имеют поля для фиксации фаз ввода, обнаружения и удаления, каждая из которых заполняется в определенный момент. Поле может заполняться из списка для выбора, содержащего типичные фазы: требования, архитектура, реализация, компонентное тестирование, комплексное тестирование, системное тестирование, приемо-сдаточные испытания и эксплуатация. В модели инкрементного или итерационного жизненного цикла можно также отслеживать номер инкремента или итерации.

Эти данные полезно занести в репозитории проекта, а также использовать в качестве способа усовершенствования процесса. Имея репозиторий проектных данных, мы можем постепенно выстроить модель устранения дефектов, которая позволит оценить, сколько ошибок будет в каждой разрабатываемой системе и когда мы их обнаружим. Эта цель обозначена в главе 3.

Как только мы получаем достаточно данных для подготовки точных прогнозов об ошибках и фазах, на которых мы вносим, находим и исправляем дефекты, мы можем попытаться улучшить процесс. Мы можем установить цели для обнаружения и устранения дефектов до завершения каждой фазы проекта в предположении, что мы можем предсказать число ошибок, которые мы внесем в систему. Ошибки менее опасны и дороги для проекта, если они обнаруживаются и устраняются на той же фазе, на которой внесены, поскольку они не порождают других ошибок. Согласно терминологии устранения ошибок, обнаружение и устранение всех ошибок, внесенных в ходе каждой фазы проекта, будет означать, что процент потери ошибок равен 0%[1].

Построение успешного процесса подготовки отчетов об ошибках

Итак, чего же достигают группы тестирования и рабочие группы проектов за счет применения удобного процесса подготовки отчетов об ошибках?

Эффективная передача полезной информации

Как было показано в главе 13, тестирование создает информацию, полезную как для внутреннего (в рамках группы тестирования), так и для внешнего (для всей группы проекта или даже компании) употребления. Основной формой передачи информации являются отчеты об ошибках. Для внешнего окружения отчеты об ошибках являются частью информации о качестве, которой тестировщики снабжают проект, включая процесс управления изменениями (см. главу 16). Для того чтобы быть действенными, отчеты об ошибках должны предоставлять программистам сведения, необходимые для исправления важных ошибок. Все вместе отчеты об ошибках являются основой для измерения всего проекта (см. главу 15).

Что касается внутреннего употребления, каждая ошибка позволяет тестировщику лучше понять природу и качество системы, которую он тестирует. (Вспомните примеры, как положительные, так и отрицательные, из нашего примера в главе 13.) Вновь повторюсь, что ошибки, обнаруженные в ходе выполнения предыдущих тестов, дают информацию для дальнейшего тестирования и для понимания ошибок, обнаруживаемых при выполнении последующих тестов.

---

[1] См. "*Metrics and Models in Software Quality Engineering, Second Edition*" Стефана Канна и "*CMM in Practice*" Панкоджа Джалота. Более простое изложение темы представлено в "*Managing the Testing Processes, Second Edition*".

Какую информацию должны предоставлять тестировщики, помимо понятных описаний дефектов, комментариев и классификации дефектов, чтобы помочь программистам исправить ошибки? Эта информация может существенно отличаться в разных проектах. Однако я бы предложил несколько идей.

- История тестирования. Пока вы не перезагрузили всю сеть, в которой выполняете тесты, всегда существует вероятность того, что ранее выполненные тесты могут повлиять на поведение обнаруженного дефекта. Например, в проекте по разработке Интернет-приложения мы вынуждены были отслеживать, какие виды сбоев соединения происходят во время тестирования на каждом устройстве, и отмечали это в каждом отчете об ошибке. Логика заключается в том, что компьютер — это в первую очередь конечный автомат. Любая совокупность действий, пока не будет выполнена перезагрузка, переводит память, процессор, накопители на жестких дисках и другие компоненты в определенное состояние, которое может повлиять на выполнение последующих действий. Сеть компьютеров — это также конечный автомат, и информация о состоянии одного компьютера может повлиять на работу других компьютеров. Информация, которую можно фиксировать в этом отношении, практически бесконечна. Поэтому с практической точки зрения большинство групп тестирования, которые фиксируют историю тестирования в отчетах об ошибках, документируют выполнение тестовых сценариев в системе, предназначенной для тестирования, до того, как была обнаружена ошибка, а также все другие ошибки, которые были найдены в системе до обнаружения данной ошибки.
- Фиксация экранов (*снимки экранов*) и даже видеоклипы. Некоторые операционные системы и большинство инструментов тестирования на основе графического пользовательского интерфейса дают возможность тестировщикам фиксировать вид экрана рабочей станции. Полученные картинки могут быть ценнее 10 000 слов в оказании помощи программистам.
- Цифровые снимки. Наличие цифровой камеры в тестовой лаборатории позволит вам получать снимки экрана или системы после обнаружения ошибки. Фиксация экранов при помощи средств рабочей станции удобна, но она невозможна, если система терпит крах. Кроме того, если вы тестируете аппаратные средства, сбои могут иметь физический характер. Трудно описать, что произошло, когда ломается болт или корпус компьютера, а снимок сделает это понятным.
- Содержимое памяти (*дамп памяти*). Некоторые системы могут быть сконфигурированы таким образом, что если система обнаружит ошибку внутри себя, она сохранит содержимое памяти целиком, включая регистры, в специальном файле. На языке UNIX и универсальных машин этот файл называется "дамп памяти". Многие конфигурации интегрированных тестовых сред позволяют

программистам просматривать дампы памяти в поисках ответов на вопрос, что случилось, когда в системе произошел сбой.

- Информация о внутренней трассировке. На ранних фазах тестирования при обнаружении большого числа ошибок, особенно если есть ошибки, воспроизводимые не каждый раз, можно попросить программистов и инженеров сборки передать то, что иногда называют *отладочной сборкой*. В качестве альтернативы можно инструментировать сборку, чтобы фиксировать информацию, которая может сообщить нам, какие выполняются операторы системы, когда происходит ошибка. Еще один вариант — сборка может содержать специальные операторы вывода, которые протоколируют внутреннюю информацию. Здесь нужно сделать одно замечание. При использовании отладочных сборок для тестирования я прекращаю принимать такие сборки на тестирование с началом фазы системного тестирования. Мне нужно время для выполнения большинства моих тестов хотя бы однажды в сборке, не являющейся отладочной. Инструментирование, отладочные протоколирующие операторы и другой код, который не будет поставляться с системой, могут замаскировать или изменить поведение ошибок.
- Рабочие файлы (*вспомогательные файлы*). Некоторые системы создают рабочие файлы в ходе своей работы. Эти файлы часто размещаются в особых каталогах и имеют имена определенного формата. (Например, приложения Microsoft Office часто создают файлы с расширением *tmp* в текущем рабочем каталоге.) Если тестировщики знают, где находятся такие файлы и как они называются, они могут присоединить эти файлы к отчету об ошибке или сохранить их в специальном месте.
- Протоколы ошибок. Некоторые программы при сбоях создают протоколы ошибок. Если система, предназначенная для тестирования, или какое-либо сосуществующее приложение дает сбой до, во время или после того, как тестируемая система продемонстрировала ошибочное поведение, нужно рассмотреть вопрос о включении файлов с протоколами в отчет об ошибке.

Большинство описанных элементов — файлы. Удобные инструменты управления дефектами, предлагаемые на рынке, позволяют тестировщику присоединять один и более файлов к отчету об ошибке. Если ваши средства управления дефектами не позволяют это делать, то группа тестирования может создать структуру каталогов где-то на сетевом диске общего доступа, в котором тестировщики сохраняют такие файлы, а программисты их просматривают.

Другим важным видом внешней информации являются измерения дефектов. Как упоминалось в главе 13, процесс выполнения тестов должен поддерживать сбор основных измерений для управления тестированием и проектом в целом. Процесс подготовки отчетов об ошибках обязан предоставлять всю важную информацию, необходимую для измерений. Нужно четко следовать процессу, поскольку измерения, для которых существенны даты, должны иметь четкую информацию о датах. Тестировщики должны тщательно определять категории ошибок, например затронутые

подсистемы, чтобы измерения, касающиеся анализа ошибок по категориям, имели смысл.

Последний вид полезной внешней информации — это информация об обходных путях. Не каждая ошибка может и должна быть исправлена. Для таких ошибок полезно знать обходные пути. Тестировщики могут документировать обходные пути для пользователей. Тестировщики могут проверить, что обходные пути переданы службам технической поддержки или сетевой поддержки пользователей. В одном из проектов в конце системного тестирования у нас был период передачи дел, в ходе которого два тестировщика совместно с двумя сотрудниками группы технической поддержки просмотрели все известные неисправленные ошибки и убедились в том, что двое последних разобрались в каждой ошибке и знают все возможные обходные пути.

Потребность в действенной передаче информации означает, что применимы обычные правила написания технической документации. Текст должен быть ясным и обязан вскрывать суть проблемы. Он должен соответствовать аудитории и излагаться понятными ей словами. Если отчеты об ошибках используют жаргонные выражения и сокращения, это должны быть хорошо известные проектные термины.

Одним из признаков неэффективной передачи информации является то, что я обычно называю *отфутболиванием* (ping-pong) *отчетов об ошибках*. Это такая игра, в которой участвуют тестировщик и программист: происходит непродуктивное перебрасывание друг другу отчета об ошибке. Вот некоторые типичные причины отфутболивания отчетов об ошибках.

- Программист утверждает, что ошибка неверно отнесена к его компоненту или фрагменту кода.
- Программист отвечает, что ошибка является невоспроизводимой или это верное поведение программы.
- Возникли личные трения между программистом и тестировщиком, поэтому программист не может допустить, чтобы тестировщик при помощи отчета об ошибке набирал очки против него.

Проблемы, которые являются невоспроизводимыми или рассматриваются как работа программы в соответствии с проектом, могут быть решены посредством тщательной подготовки отчетов об ошибках, в частности, за счет документирования шагов воспроизведения ошибки, сравнения ожидаемого результата с фактическим и установления факта, что проблема воспроизводится не всегда.

Неприятный вопрос — назначение ответственных за исправление ошибки. Необходимо делать различие между тем, где визуально зафиксированы симптомы ошибки, где ошибка находится в системе и где ее лучше исправить. В одном из проектов мы обнаружили ошибку, вызывавшую потерю данных в файлах, которые сохранялись приложениями в системе, предназначенной для тестирования. Истинной причиной была несовместимость аппаратных средств, неправильным было соединение жесткого диска (шина IDE) с тестируемой системой. Проблема была исправлена после того, как было внесено изменение в базовую систему ввода- вывода компьютера, которое оказало влияние на время выполнения операционной системой операций чтения и записи на жесткий диск.

Где же была эта ошибка? В приложении, теряющем данные? В способе передачи информации с жесткого диска в систему? Или в способе, при помощи которого система взаимодействовала с диском? Или в коде чтения и записи BIOS?

С точки зрения качества мы, по всей вероятности, можем не думать о том, где находилась ошибка или как она исправлена, поскольку неприемлемое поведение теперь отсутствует. Однако с точки зрения проекта эти вопросы иногда приобретают огромное политическое значение. Руководство хочет определить последовательность действий, которая позволит точно определять ответственных за исправление ошибки, а выбор подходящего средства для подготовки отчетов об ошибках способствует поддержке этой последовательности действий. Только менеджер разработки, а не группа тестирования, должен отвечать за назначение ответственных за исправление тех или иных проблем системы.

Включение одного симптома в каждый отчет и подготовка одного отчета на каждый симптом

Отчеты об ошибках описывают симптомы — наблюдаемое аномальное поведение, в то время как исправление ошибок — это работа с дефектами в системе. Бывает так, что непохожие отчеты об ошибках описывают один и тот же дефект в системе. Однако опыт показывает, что это нетипично. Дублирование отчетов об ошибках обычно происходит, когда два тестировщика сообщают об одном и том же неверном поведении в двух различных отчетах об ошибках. Дублирование отчетов в этом случае означает дублирование работ тестировщиками, программистами и менеджерами.

Обратное также может иметь место. Тестировщики иногда соединяют два-три симптома, которые проявляются вместе в сценарии использования или в функциональной области системы. Я часто наблюдал, что в этом случае исправлялись не все документированные проблемы. Далее тестировщики вынуждены либо создавать один или несколько новых отчетов об ошибках, описывающих оставшиеся проблемы, либо повторно открывать существующий отчет об ошибке. При этом расходуется время программиста и тестировщика, а система управления дефектами неэффективно воздействует на процесс. В итоге работа по подготовке отчетов об ошибках неверно отражает реальную картину, поэтому получаемые измерения являются искаженными.

Как и в рамках всего процесса, здесь требуется здравый смысл. Я лучше подготовлю два отчета об ошибках, чем не напишу их вовсе. Поэтому я говорю тестировщикам, чтобы они не тратили больше пяти минут на поиск в системе управления дефектами существующего отчета об ошибке. Кроме того, я предпочитаю, чтобы тестировщики обменивались своими отчетами об ошибках со всей группой тестирования по электронной почте либо для рецензирования, либо просто для информации. Я заметил, что такой способ работы снижает количество дублированных отчетов примерно до 5-10%, это вполне приемлемая цифра.

Мне не нравится, если тестировщики тратят слишком много времени на выяснение того, является ли проблема последствием одной или нескольких ошибок. По моему мнению, процесс рецензирования обычно

позволяет выявить достаточное количество таких видов отчетов об ошибках, поэтому я не слишком беспокоюсь об этом.

Установка четкой границы между тестированием и отладкой

Как отмечал Борис Бейзер, вначале место тестирования определялось как дополнение к отладке. По мере становления тестирование стало рассматриваться как отдельная совокупность задач, направленных на локализацию ошибок и управление рисками качества[1]. В компаниях с независимыми группами тестирования тестировщики направляют свои усилия на тестирование, а программисты выполняют задачи отладки.

В главе 16 этот вопрос рассматривается более подробно, здесь же коротко исследуем процесс, связанный с обнаружением симптома, исправлением соответствующего дефекта и проверкой исправления (см. рис. 14.1). Первые три этапа процесса – действия тестирования, которые выполняются группой тестирования и проводятся параллельно с процессом создания отчета об ошибке. Следующие три этапа – это действия отладки, которые выполняются группой разработки и являются частью устранения проблемы, описанной в отчете об ошибке группой тестирования. Заключительный этап – это проверочное и регрессионное тестирование, которое снова является действием тестирования. Следовательно, это совместный процесс, состоящий из семи этапов, направленный на повышение качества тестируемой системы за счет циклических корректирующих действий, управляемых отчетами об ошибках. Это совместный процесс, но каждая отдельная задача выполняется либо группой тестирования, либо группой разработки.

Удобный процесс подготовки отчетов об ошибках поддерживает такое распределение обязанностей. Как отмечалось ранее, хороший отчет об ошибке предоставляет программисту всю информацию, необходимую для воспроизведения проблемы и поиска дефекта. Подготовка отчетов об ошибке и процессы управления дефектами – это единственные инструменты, необходимые для обмена информацией между двумя группами. Это показывают события передачи информации на рис. 14.1. Процесс подготовки отчетов об ошибках должен избавлять программиста от необходимости задавать вопросы тестировщику о том, в чем состоит смысл его отчета об ошибке. Процесс должен также избавлять тестировщика от необходимости тратить время на повторные усилия по воспроизведению ошибки для программиста.

Я не одобряю участия тестировщиков в отладке. Объясню, почему это столь важно для меня, когда я выполняю обязанности тест-менеджера.

1. Есть много работы по тестированию, которую должны выполнить тестировщики. Я обязан отчитываться за эту работу. Когда тестировщики помогают исправлять проблемы программистам, они не заняты выполнением ключевых процессов тестирования.
2. Я не могу управлять тестированием, когда тестировщики по своему усмотрению отдают свои силы и свое время группе разработки.

---

[1] См. книгу Бориса Бейзера "*Software Testing Techniques*".

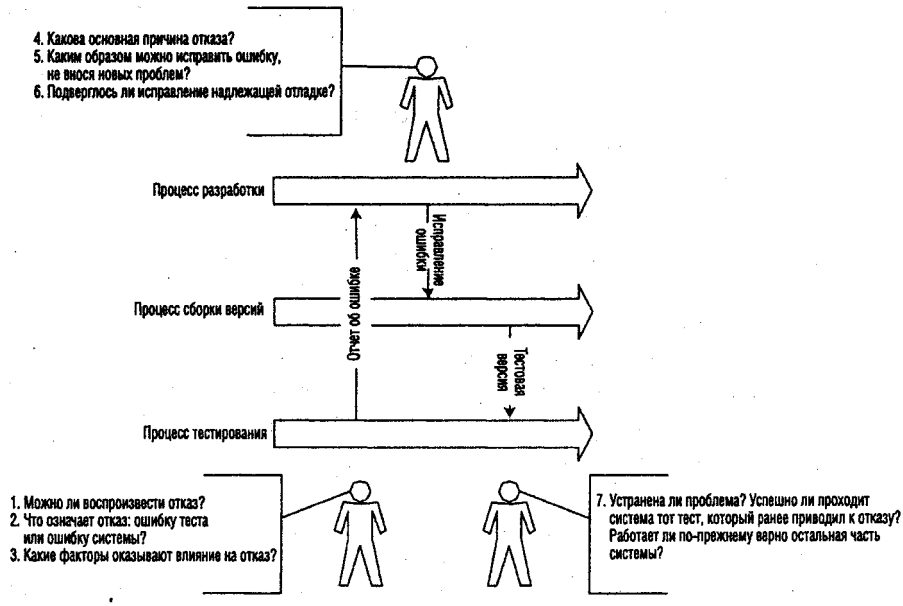


Рис. 14.1. Процесс поиска, отладки и проверки исправления ошибки

Часть управления состоит в том, чтобы распределить людские ресурсы согласно корпоративным целям. Я не могу допустить перераспределения ресурсов по желанию тестировщика, который, честно говоря, может не видеть всей глубины расстановки приоритетов.

3. Помощь в отладке, которую оказывают тестировщики, редко является разумным использованием ресурсов. Даже при условии, что тестировщик — опытный программист, причиной того, что он занимается тестированием, а не программированием, является то, что он более ценен для компании в роли тестировщика.
4. Это укрепляет самомнение и выглядит как работа в команде, но не способствует развитию карьеры. Я с удовольствием организую работу тестировщика так, чтобы поддержать выбор карьеры программиста, добавив задачи создания инструментов и автоматизации тестирования в его рабочую нагрузку. Такие действия больше подходят для освоения программирования, чем участие в отладке, что выглядит больше как путешествие в программирование, а не потенциальная информация в резюме.
5. Эти действия обычно указывают на слабые отчеты об ошибках, подготовленные тестировщиком. Если программист не может понять, что неверно работает, при помощи отчета об ошибке и вынужден спрашивать тестировщика, то решением является бо-

лее жесткий процесс подготовки отчетов об ошибках, а не участие тестировщика в отладке.

В некоторых случаях тестировщики обязаны принимать участие в отладке, особенно если для воспроизведения проблемы необходимы уникальные конфигурации тестов или инструменты тестирования. Это, безусловно, исключение из правил, и удобный процесс подготовки отчетов об ошибках сводит к минимуму эту необходимость.

Решение проблем

Подготовка отчетов об ошибках ставит группу тестирования перед рядом проблем даже в том случае, когда процесс написания отчетов является совершенным. Эти проблемы носят как технический (трудно готовить отчеты о неизвестном), так и политический (необходимо управлять межличностными и групповыми отношениями, частью которых является критика работы других) характер.

Разделение ошибок, свойств и ответов "ну, и что?"

Противоречивость или полное отсутствие спецификаций требований или архитектуры может усложнить процесс подготовки отчетов об ошибках. Когда программисты отвечают на отчеты об ошибках комментариями типа "как раз так система и должна работать", у тестировщиков отсутствует ясный способ для того, чтобы опровергнуть это утверждение. Обычно я пытаюсь разрешить противоречие, обсуждая его с различными заинтересованными сторонами, и не допускаю вовлечения собственного мнения в проблему. Менеджеры технической или сетевой поддержки пользователей, группы продаж и маркетинга, бизнес- аналитики, менеджер разработки и другие руководители проекта, вероятно, имеют собственные мнения по поводу того, как должен вести себя продукт. Кроме того, группа управления изменениями или комиссия по разбору дефектов (см. главу 16) должна уметь отделять ошибки от недокументированных свойств.

Один из возможных ответов программистов и менеджеров на отчет об ошибке: "ну, и что?" или "ни один настоящий пользователь никогда этого не сделает" и т.п. Это может быть признаком того, что менеджер или программист, делающий такие замечания, не беспокоится о качестве системы, запутался в требованиях (зафиксированных или нет) или столкнулся с какой-то похожей проблемой. Либо это признак того, что тестировщик, написавший отчет об ошибке, не смог донести суть проблемы до другой стороны.

Тестировщики должны искать способ увязать любую обнаруженную ошибку с реальной жизнью, особенно при выполнении тестов, которые, возможно, не имеют отношения к реальным сценариям действий пользователя. Тестировщики, которые этого не делают, не должны удивляться, услышав "ну, и что?" Эффективная передача информации о результатах тестирования является прямой обязанностью тестировщиков, поскольку именно мы предоставляем эту услугу. На нас, тестировщиков, возложена

обязанность доказывать, что мы нашли ошибку. Группы разработки и управления проектом не должны верить нам на слово, что некоторая невразумительная проблема указывает на глубокую ошибку в системе.

Программисты являются основными внутренними потребителями наших отчетов об ошибках, поэтому они судят о качестве содержания отчетов. Если их устраивает сообщение тестировщика: "Эй, проверь, что за непонятное поведение я наблюдал", отлично. Редко, но я встречался с таким типом любопытных программистов в проектах с сильным управлением, когда качество рассматривалось как столь же важный компонент проекта, как сроки, бюджет и свойства. Тем не менее, тестировщики обязаны понимать информационные потребности программистов, когда они отправляют программистам отчеты об ошибках.

Менеджеры проектов — это еще одна ключевая группа внутренних потребителей отчетов об ошибках. Они обычно хотят знать, почему они должны вкладывать ресурсы проекта — загрузку программиста и планируемое время в проекте — в исправление ошибок. Мои наиболее неотразимые аргументы описывают, какой ущерб могут причинить ошибки заказчикам или пользователям, что в итоге выльется в (потенциально большое) количество звонков в группу технической поддержки и в другие менее ощутимые затраты на (низкое) качество. Когда программисты испытывают сильное давление со стороны сроков и/или бюджета, по моим наблюдениям, они часто демонстрируют похожие подходы к разбору ошибок. В любом из проектов, где ограничения сроков и бюджета лимитируют число проблем качества, которые могут быть разрешены, поиск влияния ошибок на пользователя является наилучшим подходом при подготовке отчетов об ошибках[1].

Вот почему в своем процессе подготовки отчетов об ошибках я делаю ударение на увязывании отказа с проблемами заказчика, особенно в кратком описании. Краткое описание является ключевым, поскольку это нередко единственная часть отчета об ошибке, которую прочитывают в ходе совещания группы управления изменениями или комиссии по разбору ошибок, так как ошибок может быть очень много.

Я также рекомендую, чтобы тестировщики четко описывали в разделах "Шаги воспроизведения" и "Локализация" поведение, которое они ожидали увидеть, поведение, которое они видели фактически, и значение (с точки зрения пользователя) различия между двумя вариантами поведения. В нашем примере подразумевалось, что повреждение данных является нежелательным поведением, однако для многих ошибок поведение может казаться разумным, но быть ошибочным в определенном смысле. Тестировщики должны отразить это в своих отчетах об ошибках.

Я стимулирую тестировщиков занимать активную позицию в отделении друг от друга ошибок, свойств и "ну и что?", а не просто набивать базу данных дефектов отчетами об ошибках, игнорируя обвинения программистов и других сотрудников проектной группы о наличии в сигнале чересчур большого шума. Вновь повторю, поскольку мы предоставляем

---

[1] Этот раздел появился в результате онлайновой беседы с Бреттом Петикордом, Джеймсом Бачем, Семом Канером, Россом Коллардом, Кэти Айберли и Аланом Йоргенсеном. Я благодарю участников за их вклад в обсуждение, хотя здесь отражена только моя точка зрения.

группе проекта информационные услуги, мы несем ответственность за эффективную передачу этой информации. Если они рассматривают нашу информацию как шум, что-то серьезно нарушено. Мне рассказывали коллеги, что в ситуациях, когда тестировщики игнорировали эту проблему, случалось так, что тестировщики должны были получить разрешение от сотрудников группы разработки или других групп проекта на открытие отчета об ошибке.

Пожалуйста, примите к сведению, что я не выступаю в защиту игнорирования проблем на основании того, что кто-то считает их несущественными или мелкими. Тестировщики, как и программисты, часто не обладают достаточной информацией для принятия решений о том, что точно означает качество для тестируемой системы. Если мы, тестировщики, решим игнорировать определенную проблему, будучи уверенными в том, что ни для кого из пользователей это не будет проблемой, то может случиться так, что позже к нам придут сотрудники групп продаж, маркетинга или аналитики и спросят, почему столь очевидные ошибки не были найдены при тестировании. Однако, как отмечалось в главе 13, есть время, чтобы сконцентрировать усилия на опасных областях, и есть время на наведение лоска в продукте.

Обработка случайно исправленных ошибок и невоспроизводимых симптомов

Однажды я работал в проекте, где менеджер разработки готовил новые версии ежедневно и даже каждый час, если обнаруживались ошибки, требующие срочного исправления. Мы решили не принимать столь часто версии в тестовую лабораторию, поскольку это нарушало бы процесс выполнения тестов требованием постоянно заниматься проверочным тестированием. (Представьте, что вы ведете машину и смотрите только в зеркало заднего вида.) На каждом совещании по разбору ошибок менеджер разработки в ответ на любую серьезную ошибку независимо от того, уже занималась исправлением ошибок его группа или нет, говорил: "Проверьте нашу последнюю версию. Я думаю, что мы устранили эту проблему". В конце концов я не выдержал и сказал ему: "Знаешь ли, ты, верно, думаешь, что исправил все эти ошибки случайно, но мой опыт показывает, что значительно меньше ошибок устраняется случайно, чем это хотелось бы менеджерам разработки. Пока ты не потратишь немного времени на то, чтобы воспроизвести проблему и исправить ошибку, я не собираюсь просить свою группу тратить время на перетестирование ошибок".

Мой ответ был довольно грубым, но я считаю его верным в отношении моих дальнейших действий. Ошибки обычно не устраняются случайно. Честно говоря, менеджер разработки в этой истории использовал запрос на перетестирование каждой ошибки в каждой новой версии, рассчитывая на то, что каким-то волшебным образом многие из них исчезнут и удастся скрыть тот факт, что качество системы ужасно низкое. Он надеялся использовать это средство в качестве механизма ослабления натиска дефектов, о которых сообщала группа тестирования.

Тратить время тестировщика на перетестирование ошибок, на исправление которых никто не потратил ни миллисекунды времени, — это

неэффективное использование времени группы тестирования. В таких обстоятельствах я бы предпочел обсудить с проектной группой процесс, в соответствии с которым до того, как отчет об ошибке будет возвращен группе тестирования для проверочного тестирования, программист должен проверить систему и выяснить, осталась ли проблема.

Другим препятствием, сильно осложняющим нашу жизнь, является то, что некоторые ошибки не воспроизводят один и тот же симптом вновь и вновь по команде, как проверенный штамп. Например, утечки памяти зависят от особых условий, возникающих иногда в течение длительного периода времени до того, как система потерпит крах или работа с приложением будет блокирована. Однажды я работал в проекте, в котором нерегулярно нарушались модемные соединения, но нам не удалось заставить проблему возникать, когда необходимо. В конечном счете, программисты вынуждены были добавить специальные средства в программу, чтобы отследить проблему.

Некоторые смешивают *появляющиеся время от времени* проблемы с *несущественными* проблемами. Во многих случаях периодические проблемы являются серьезными ошибками. Крах систем, нарушение модемных или сетевых соединений или внезапный отказ части системы как реакция на входные данные может полностью нарушить работу пользователя и даже повредить его данные. Свойства, работающие время от времени, — это свойства, применения которых будет избегать пользователь.

Поэтому я всегда скептически отношусь к заявлению, что серьезная периодически возникающая ошибка "просто исчезла", раз ни программисты, ни тестировщики не могут воспроизвести ее. Обычно я держу такие ошибки открытыми на протяжении нескольких версий (две-три недели), чтобы убедиться, что мы не встречаемся с ними вновь. (В некоторых компаниях есть специальное состояние "наблюдение" или "мониторинг", в котором хранятся такие ошибки.) Я спрашиваю программистов, чем может помочь группа тестирования в ходе наблюдения такой проблемы. Например, перезагрузка зависшей системы обычно разрушает данные состояния, которые могли бы помочь программисту отследить ошибку. Вероятно, программисты попросят покопаться в системе и зафиксировать эту информацию при помощи сети. И я должен удостовериться в том, что запись ведется всякий раз, когда ошибка действительно *проявляется*. Ощущение ненадежности системы, которую может вызвать такая ошибка, является исключительно важным для разумной расстановки приоритетов при исправлении ошибок, и это означает, что я должен быть способен отчитаться о частоте возникновения ошибки.

Удаление шума из сигнала

Система управления ошибками является полезным, но не единственным способом коммуникации, и она не должна применяться для обсуждения всех проблем. Поскольку мы пишем отчеты об ошибках для обсуждения проблемы в тестируемой системе, для принятия решения о ее приоритете и устранении, мы должны быть уверены в том, что используем канал,

который создан при помощи процессов подготовки отчетов об ошибках и управления дефектами исключительно для этой цели.

Сложности возникают, когда этот инструмент начинают использовать для эскалации проблем руководству. Я стараюсь избегать пересылки в *систему управления дефектами отчетов об ошибках*, которые сообщают о сбоях в процессе разработки системы, а не в *системе, предназначенной для тестирования*. Например, если инженер сборок не передал в понедельник в 9 часов утра сборку, я не буду писать отчет об ошибке с названием "Сборка не передана в срок". Я пойду к инженеру сборок и объясню ему значение своевременного получения версий. Если проблема повторяется, я передаю ее на уровень руководства инженера сборок. Конечно, если сборка передана в срок, но отказывается работать, вероятно, я напишу отчет об ошибке под заголовком "Сборка 781 не инсталлируется".

Сложности возникают также, когда некоторые делают личные пометки в базе данных управления дефектами. Я работал в проекте, где один из лучших программистов, наиболее отзывчивый партнер со стороны разработки, о котором только мечтают тестировщики, имел привычку делать персональные пометки в базе дефектов. Поскольку он был единственным, это не вызывало больших проблем. Однако если каждый программист сделает для себя одну пометку в день, мы получим 25 отчетов об ошибках в системе, которые не будут иметь никакого отношения к ошибкам.

Наконец, проблема шума существует, когда тестировщики сообщают об ошибках, которые таковыми не являются. В главе 13 отмечалось, как важно делать различия между ошибками в системе, предназначенной для тестирования, ошибками в системе тестов и обычным непониманием того, как должна работать система. В процессе подготовки отчетов о дефектах воспроизведение проблемы и локализация причины сбоя, а также итоговый перекрестный просмотр отчета помогают понять, в чем состоит проблема.

В любом из этих случаев добавляется шум в ценный информационный канал. Шум может ввести в заблуждение тестировщиков и программистов, испортить процессы подготовки отчетов об ошибках, подготовки отчетов о ходе работ по тестированию (см. главу 15) и управления изменениями (см. главу 16), а также исказить измерения проекта, с какой бы моделью устранения дефектов вы ни работали.

Создание доверительных отношений с программистами

В моей группе работал тестировщик, который сказал мне, что ему нравится тестирование, потому что он имеет шанс поймать программистов на ошибке. Он сохранял информацию о том, в чьих компонентах он обнаружил проблемы, и не успокаивался до тех пор, пока не находил ошибку в коде каждого программиста. Такой тип отношений, при котором программист пытается скрыть ошибки от тестировщика, может породить большие проблемы с отчетами об ошибках. Если программисты воспринимают тестирование, как игру в салочки, и рассматривают отчет об ошибке в их компоненте, как будто им теперь нужно "водить", то, без сомнения, возникнет проблема отфутболивания отчетов.

Другая ловушка возникает тогда, когда тестировщики или тест-менеджеры начинают отслеживать работу отдельных программистов, ожидая исправления ошибки. Такой подход может породить целую совокупность проблем. Во-первых, если программисты не готовят отчеты для тестировщиков, тестировщики не обладают достаточной информацией для того, чтобы понять, какие еще задачи нужно поручить программистам. Нарушение приоритетов не способствует расположению программистов и менеджеров к вам. Во-вторых, если тестировщики не имеют авторитета у программистов, они воспринимаются только как раздражители, не приносящие никакого полезного результата. Тестировщики могут иногда привлекать программистов к работе над ошибками с помощью взяток (обед) или интеллектуального вызова (написанием увлекательного отчета об ошибке), но желание получить свободу в управлении другой группой обычно не срабатывает.

Проблемы возникают и в тех случаях, когда тестировщики используют данные отчетов об ошибках для составления графиков или отчетов, которые представляют отдельных людей в неприятном свете. Например, многие инструменты управления дефектами содержат поле "Примерная дата исправления", которую программисты заполняют, исходя из своих возможностей. Подготовка отчета, который для каждого программиста показывает, какие ошибки еще не исправлены и по каким сроки уже прошли, будет унизительной. Аналогично, подготовка отчета, в котором ошибки отсортированы и собраны по компонентам или подсистемам и содержат указание ответственного, т.е. указывают, кто внес ошибку в код, является вредным для обвиняемых программистов.

Еще одна проблема – доверие. Если мы занесем в систему слишком много отчетов об ошибках, которые не являются проблемами, или совершим слишком много ошибок при интерпретации результатов тестирования, рано или поздно создастся впечатление, что тестировщики предоставляют информацию, которой не стоит доверять. Мой опыт показывает, что для нанесения ущерба доверию группе тестирования не надо слишком много шума со стороны группы.

Тема корректной интерпретации результатов тестирования подробно рассматривалась в главе 13. Структурированное тестирование является не только основой процесса подготовки отчетов об ошибках, но и основой для доверительных отношений между программистами и тестировщиками. По-моему мнению, при подготовке отчетов об ошибках помогут следующие пять советов.

1. Ориентируйтесь на оказание услуг всему проекту. Какие информационные услуги вы можете оказать как тестировщик, чтобы содействовать успеху проекта?
2. Сохраняйте спокойствие в ходе обсуждения отчетов об ошибках с программистами, менеджерами и другими сотрудниками проектной группы. Защита качества – это хорошо, но гнев, неумение держать себя в руках неуместны.
3. Открыто обсуждайте отчеты об ошибках с пониманием того, что есть вероятность и вашей ошибки. Защищайте разумные ожидания заказчиков.

4. Передавайте только отчеты об ошибках качества системы, будьте открыты для предложений по повышению качества отчетов об ошибках. Необходимо помнить, что проектная группа, выступая в роли заказчика ваших информационных продуктов, принимает решение в том, что означает качество.
5. Проявляйте гибкость в подготовке и передаче отчетов об ошибках. Если программисту требуется определенный файл или снимок экрана, прикрепленный к отчету об ошибке (если система управления дефектами поддерживает прикрепление файлов), необходимо проявлять стремление к сотрудничеству.

Как только возникнет ощущение, что мы серьезно нацелены на передачу понятных отчетов об ошибках способом, удобным для всех участников проекта, информация об этом быстро распространится по компании.

Выбор инструмента управления дефектами

Хороший инструмент управления дефектами должен способствовать сбору и управлению информацией, о которой говорилось выше. Но нам требуется несколько больше. Если сбор информации и управление – это все, что нам нужно, мы можем в качестве инструмента управления дефектами использовать обычную базу данных. Однако основная ценность удобного инструмента управления дефектами состоит в его способности собирать информацию и управлять ею в ходе всего жизненного цикла каждого отчета об ошибке, начиная с первого обнаружения и заканчивая ее окончательным устранением или отсрочкой.

На рис. 14.2 показан один из возможных вариантов жизненного цикла дефекта. Отчеты об ошибках проходят через последовательность состояний (жизненного цикла) до окончательного устранения в виде либо отклонения отчета, либо проверенного закрытия проблемы, либо отсрочки симптома (и всех связанных отчетов об ошибках) как несущественного

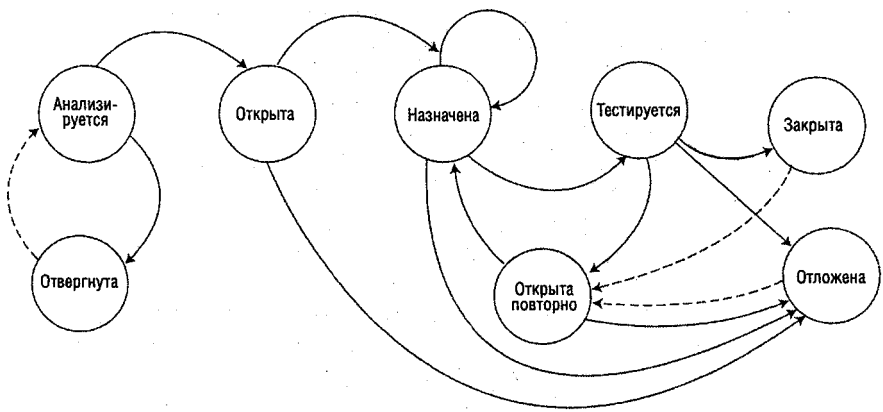


Рис. 14.2. Жизненный цикл ошибки

для исправления в данной версии. Для каждого состояния, не являющегося заключительным, менеджер или группа управления изменениями (см. главу 16) определяет ответственного, который должен переместить ошибку в очередное состояние. Удобная система управления дефектами позволяет реализовать и автоматизировать такой жизненный цикл, хотя необходима поддержка процесса управления дефектами со стороны группы проекта и руководства. Как правило, по мере того как участники процесса изменяют состояние отчета об ошибке, они протоколируют изменения, состояние или вводят запись истории в базу данных отчетов об ошибках.

Степень, в которой сценарий работы систем управления дефектами может быть приспособлен к жизненному циклу ошибок в вашем проекте, является ключевой метрикой качества инструмента управления дефектами. (Если руководство не обладает достаточным опытом в определении жизненного цикла, следует начать со сценария работы, заложенного в инструмент по умолчанию, а затем постепенно настроить его на потребности компании.) Состояния в жизненном цикле отчетов об ошибках должны представлять ключевых участников процессов обнаружения, исправления и проверки устранения дефектов. Переходы между состояниями показывают передачу управления между этими ключевыми участниками.

Заметим, что поддержка жизненного цикла в системе управления дефектами может и, вероятно, должна быть тесно интегрирована с управлением версиями. Удобная система управления дефектами позволяет группе разработки при подготовке тестовой версии делать пометки "готов к тестированию" в отчетах об ошибках и запросах на усовершенствование, связанных с изменениями в версии. По получении версии группа тестирования может получить отчет с перечнем всех элементов, находящихся в состоянии "готов к тестированию". Часто системы управления дефектами могут взаимодействовать с инструментами управления конфигурацией, что позволяет автоматизировать большую часть этого процесса[1].

Внедрение усовершенствований

Подготовка отчетов об ошибках как процесс и как действие в любом проекте редко идет идеально. Как поставить процесс подготовки отчетов об ошибках под контроль, зависит в значительной мере от того, в каком состоянии текущий процесс. Однако хочу внести несколько предложений.

1. Сначала устраните неверное отношение тестировщиков к отчетам. Лучший в мире процесс ничего не сможет сделать, если тестиров-

---

[1] Я подробно рассматриваю тему инструментов управления дефектами в книге "*Managing the Testing Process, Second Edition*". Могу также порекомендовать исследование процесса выбора инструментов тестирования в книге "*Software Test Automation*" Марка Фьюстера и Дороти Грэм.

Нужно ли включать тестовые скрипты или тестовые процедуры в отчеты об ошибках?

Некоторые тестировщики и некоторые инструменты тестирования выступают за включение автоматизированных или ручных тестовых скриптов или процедур целиком в отчеты об ошибках в качестве шагов воспроизведения или в качестве самого отчета. Я обнаружил, что тестировщики, которые используют тестовые скрипты как основу для отчетов об ошибках, обычно не достаточно любознательны в отношении обнаруженной ошибки и пишут малопонятные отчеты об ошибках. Считаю, что этот подход не способствует эффективной работе.

Представьте тестировщика, который осматривает поле, полное камней, зная, что под некоторыми камнями селятся неприятные ошибки. Тестовые сценарии являются полезными инструментами для переворачивания подходящего множества камней. Однако, как только мы обнаруживаем ошибку, простого сообщения программисту "Переверни камень X" недостаточно. Во-первых, многие программисты не пойдут переворачивать камень X только на основании того, что их попросил об этом тестировщик. Во-вторых, как узнать, что одна и та же ошибка все время остается под камнем X? В-третьих, возможно, наиболее важная причина — работа тестировщика состоит в объяснении смысла и значения того, что было обнаружено под камнем X, а в обязанности программиста не входит вытягивать из нас эту информацию. Мы, тестировщики, продаем информационные услуги, и вряд ли можно назвать информацией сообщение: "Перевернул камень X, увидел скопление каких-то объектов, я думаю, что это ошибки". Мы должны достать лопату и копать. Исследовать пространство вокруг места с ошибкой. Локализовать. Обобщить. Сравнить с другими результатами тестирования. Одним словом, думать.

Итак, я *выступаю* за указание в отчете об ошибке на тестовый сценарий (сценарии), который относится к проблеме. Однако это информация, прежде всего, для группы тестирования. Когда придет время проверочного и регрессионного тестирования, т.е. как только разработчик использует отчет об ошибке для исправления проблемы, это позволит нам быстро узнать, какие тесты повторять.

щики и программисты используют процесс подготовки отчетов об ошибках в качестве оружия друг против друга.

2. Отчет об ошибке — это наиболее осязаемая часть информационных продуктов, подготавливаемых тестировщиками. Любая работа по усовершенствованию процесса, нацеленная на получение более качественного продукта, начинается с переговоров с заказчиками. Кто будет читать ваши отчеты об ошибках? Что им нравится и что не нравится в текущем процессе? Какие направления по улучшению отчетов об ошибках им кажутся предпочтительными? Начните задавать эти вопросы. И уже это станет мощным сигналом для рабочей группы проекта, особенно когда они увидят работу над их идеями.

3. Если у вас нет инструмента управления дефектами, приобретите его. Такие инструменты не обязательно дороги, и даже существуют бесплатные продукты[1]. При необходимости можно использовать таблицы, но текстовые файлы, документы обработки текстов или совокупность электронных сообщений здесь не помогут. Поскольку частью подготовки отчетов об ошибках является сбор измерений, мы не можем использовать инструменты, которые не поддерживают общий анализ данных об ошибках.
4. Изучите последовательность действий, которая имеет место при обнаружении, исправлении и проверке устранения ошибок, и сравните ее с тем, что позволяет сделать имеющийся у вас инструмент управления дефектами. Каковы ключевые участники последовательности действий и как происходят передачи ошибок? Можно ли улучшить жизненный цикл, заложенный в инструмент, чтобы более качественно собирать информацию о процессах, соглашениях, передачах и ключевых участниках?
5. Если у вас уже есть инструмент, нужно попытаться приспособить его к поддержке удобного процесса подготовки отчетов об ошибках, прежде чем принимать решение о замене инструмента. Изменение системы управления дефектами — это важное решение, особенно если вы находитесь в середине проекта. Технические проблемы, связанные с миграцией данных из одного инструмента в другой, далеко не тривиальны. Мы с одним из моих коллег как-то провели несколько недель за этим занятием. Имеются также политические проблемы. Системы управления дефектами обычно используются и другими группами, в том числе группой сборки версий, группой разработки, группой технической поддержки. Кроме того, менеджеры этих групп часто делают собственные инструментальные панели, использующие данные управления дефектами. В определенное время замена систем управления дефектами может иметь смысл, однако мне удавалось справляться даже с такой системой, которую я считаю одной из худших систем управления дефектами на рынке.
6. Применяйте удобный процесс подготовки отчетов о дефектах, но проявляйте гибкость и чувство приоритетов. В моих группах было только одно нерушимое правило в процессе — каждый отчет об ошибке должен пройти перекрестное рецензирование.
7. Опробуйте проведение перекрестного рецензирования, прежде чем отказаться от него как от препятствия на пути к эффективному тестированию. Плохие отчеты об ошибках могут привести к бесполезному расходу времени. Перекрестные рецензии можно упростить до прочтения непосредственно на экране тестировщиком от-

---

[1] Например, базу данных Microsoft Access, описанную в моей книге "*Managing the Testing Process, Second Edition*", можно бесплатно скачать с web-сайта www.rexblackconsulting.com. (Даже если вы не захотите использовать мою базу данных, возможно, будет полезно просмотреть набор полей, которые применяются при отслеживании ошибок и управлении ими.) Стив Сплейн в книге "*Testing Web Security*" приводит целый список инструментов управления дефектами. Можно также скачать Bugzilla с сайта www.mozilla.org.

чета об ошибке другого тестировщика перед тем, как отправить его в систему управления дефектами.

8. Рассмотрите введение метрик для измерения качества отчетов об ошибках. Например, каков процент вновь открытых отчетов об ошибках? Другими словами, как много отчетов возвращается с отметками "исправлено", "работает в соответствии с проектом" или "невоспроизводимо", тогда как указанные ошибки не исправлены, на самом деле ошибки имеются в проекте или легко воспроизводятся? Другой метрикой может быть среднее время, в течение которого отчеты об ошибках остаются открытыми. Удобный процесс подготовки отчетов об ошибках должен способствовать снижению этих показателей.

Какие бы изменения вы ни внедряли, важно сохранять терпение и проявлять настойчивость. Люди вырабатывают привычки в работе, особенно в отношении задач, которые приходится выполнять вновь и вновь. Нужно иметь это в виду при работе над улучшением процесса подготовки отчетов об ошибках.

ГЛАВА 15

Четвертый элемент: подготовка отчетов о результатах тестирования

Давным-давно философы верили, что мир создан из четырех элементов: земли, ветра, огня и воды. Современная химия учит нас, что существует более сотни элементов. Тем не менее, проекты, в которых я работал, обычно состояли из четырех основных элементов.

- *Свойства.* Создаем ли мы необходимое множество функций, которые решают проблемы пользователей и заказчиков?
- *Сроки.* Можем ли мы передать систему, решающую эти проблемы, в заданные сроки.
- *Бюджет.* Можем ли мы передать систему способом, который является привлекательным с финансовой точки зрения для заказчиков, пользователей, спонсоров проекта и создателей системы?
- *Качество.* Можем ли мы передать свойства в состояниях, которые подходят для применения пользователями и заказчиками?

Менеджеры проектов знают, в каком состоянии находится проект с точки зрения бюджета, поскольку они могут получить эту информацию от финансового директора или бухгалтеров. Менеджеры проектов знают, в каком состоянии находится проект с точки зрения сроков, отслеживая ход проекта относительно контрольных точек и выполненной работы. Менеджеры проектов знают, в каком состоянии находится проект с точки зрения свойств, поскольку программисты, конфигураторы и инженеры по сборке версий могут сообщить, какие свойства реализованы. Лучшие методы организации работ в создании программных систем и в управлении проектами, применяемые компетентными специалистами, дают возможность менеджеру проекта четко разобраться в трех из четырех элементах. Но как менеджеры проектов могут пролить свет на четвертый элемент?[1]

---

[1] Фрагменты этой главы впервые появились в журнале "*Software Testing and Quality Engineering*", Volume 2, Issue 2 (Mar/Apr 2000). Я благодарю Брайана Лоуренса, Брайана Марика и Элайн Уэмбек за их помощь в редактировании этой статьи.

Часть ответа на этот вопрос приведена в предыдущей главе. При помощи отчетов об ошибках мы информируем проектную группу о возникновении ситуаций, где ответ на вопрос о качестве отрицательный – система не подходит для использования таким способом. Слишком часто менеджеры проектов вынуждены работать в неведении относительно более широкого вопроса о качестве, особенно в сравнении с другими элементами проекта. Я уверен, что эта диспропорция является одной из причин того, что страдает качество продукта.

В главе 1 отмечалось, что успешный процесс тестирования должен предоставлять информацию, которая позволяет компании управлять рисками качества системы. Эта информация, передаваемая точно, своевременно и в доверительной манере, группе управления проектом, освещает четвертый элемент проекта – качество. В этой главе я предлагаю посмотреть, как происходит раскрытие четвертого элемента проекта и как мы может наилучшим образом выступать в роли прожекторов и носителей этого знания для проекта[1].

Процесс подготовки отчетов о результатах тестирования

Общий процесс подготовки отчетов о результатах тестирования, показанный в Процессе 11, описать легко. Тот же процесс применим и для получения обычных отчетов о ходе проекта, и для анализа критических состояний или особых событий, хотя, по всей вероятности, вы будете адресовать отчеты разным аудиториям и отвечать на разные вопросы.

Например, в проекте разработки Интернет-приложения я представлял совокупность графиков хода тестирования на еженедельных совещаниях по проекту тестирования. Однако дважды в неделю у нас проходило совещание по разбору ошибок, где мы принимали сложные решения о том, какие ошибки исправить/отложить, используя отчет, в котором перечислялись краткое наименование, серьезность, приоритет и дата подготовки отчета по всем открытым ошибкам, начиная с момента последнего совещания.

В одном случае мы провели специальное совещание, посвященное анализу результатов выполнения некоторых тестов производительности и сравнению полученных результатов с ранее сделанными прогнозами производительности на основе статического и динамического моделирования. На этом совещании использовалось множество подробных графиков, показывающих производительность системы в разных условиях,

---

[1] Сравнение тестирования с прожектором приведено в книге Сема Канера, Джеймса Баха и Брета Петтикорда "*Lessons Learned in Software Testing*", в которой они пишут, что "[тестировщики] – это прожекторы проекта". Джонатан Бах, брат Джеймса Баха и менеджер тестовой лаборатории Satisfice, имеет персональную табличку с надписью TESTER (тестировщик). Эта табличка – элемент памятной серии о маяках штата Вирджиния, на которой изображены три маяка с надписью "Храните негасимый свет". Мне особенно нравится сравнение тестирования с маяком, поскольку свет маяков помогает лодкам обойти скалы точно так же, как разумное тестирование не допускает попадания проектов на мель.

и они сравнивались с таблицами ожидаемой производительности, полученной при помощи моделирования.

Процесс 11. Процесс подготовки отчетов о результатах тестирования[1]

№ этапа	Этап	Выполнено?
1.	Определить аудиторию, которая обычно включает всех лиц, заинтересованных в процессе тестирования и в качестве системы. Определить цели проекта.	☐
2.	Определить результаты, которые должны быть представлены. Обычно это информация, которая должна отвечать на вопросы аудитории о тестировании, особенно о том, что означают результаты тестирования для целей проекта.	☐
3.	Отобрать измерения и составить отчеты и графики, которые отвечают на поставленные вопросы.	☐
4.	Представить результаты тестирования аудитории в соответствии с требованиями.	☐
5.	При необходимости уточнить формулировки отчета и виды графиков, а также способ подачи информации аудитории, каждому заинтересованному лицу в отдельности и проекту в целом, повторив этапы 1 – 4.	☐

Каждое совещание имеет свою аудиторию с отличающимися потребностями в информации, с разными вопросами. Я использую в качестве темы нашего примера еженедельную отчетность о ходе тестирования, но прошу иметь в виду, что этот процесс применим и к другим типам отчетов о результатах тестирования.

Несмотря на линейную последовательность задач, отчетность о результатах часто является наиболее сложным процессом, с которым сталкиваются специалисты-тестировщики в любом проекте. Чтобы готовить эффективные отчеты о результатах тестирования, тестировщики должны обладать тремя качествами. Во-первых, они должны быть способны получать глубокие знания о ходе тестировании в целом, о тенденциях и следствиях обнаруженных группой тестирования ошибок и о результатах тестирования в настоящий момент. Во- вторых, они должны владеть навыками устной и письменной речи, в том числе для передачи сложных и неприятных сообщений. В-третьих, они должны обладать навыками дипломатии для поддержания общего ощущения доверия группе тестирования и информации, которую они предоставляют, особенно, когда их сообщения указывают на то, что проект в опасности. Теперь посмотрим, как тест-менеджер проекта "Суматра" Джамал Браун достигает результата.

---

[1] Некоторые читатели узнают в этом описании вариацию парадигмы цель-вопрос-измерение Виктора Бэзили. Хороший источник онлайновой информации об этом методе можно найти в статье Розенберга и Хьятта "Developing a Successful Metrics Program" по адресу satc.gsfc.nasa.gov/support/index.html.

Инструментальная панель, система сбалансированных показателей (Dashboard, Balanced Scorecard)

Это набор метрик, которые обычно представлены в виде небольшого числа диаграмм или графиков. Они дают представление о текущем статусе некоторой деятельности. Аналогией здесь является приборная доска автомобиля, на которой набор из двух-трех измерительных устройств и предупреждающие светодиоды обеспечивают водителя важной информацией о состоянии транспортного средства, не отвлекая его от наблюдения за дорогой и движением. Термин "система сбалансированных показателей" означает тот факт, что при удачном подборе диаграммы или графики такого рода дополняют друг друга. Иными словами, любое, отдельно взятое измерение может склонить к неверным выводам, тогда как кумулятивный эффект всех метрик должен образовать контекст для других измерений.

Джамал определяет содержимое инструментальной панели и отчеты о результатах тестирования большой сборки

В нашем гипотетическом примере вернемся к тому моменту, когда Джамал готовил план тестирования. 17 сентября Джамал разослал черновик плана тестирования по электронной почте. Он также отправил предлагаемый набор графиков для отчетности о ходе работ по тестированию, которые он назвал инструментальной панелью тестирования.

Он направил это сообщение ключевым участникам проекта "Суматра", заинтересованным в тестировании. В этот список вошли вице-президент поддержки разработок Джон Албертсон, вице-президент системных разработок Гарольд Джоунз, вице-президент по продажам Сайсли Райс, менеджер разработки новой версии Дженни Кауфман, менеджер по маркетингу продукта SpeedyWriter Кейт Эрнандес, менеджер системных операций и администрирования Кейт Ли и менеджер подготовки версий Ясухиро Канагава. Эти семь человек совместно с менеджером по комплексному и системному тестированию Джамалом Брауном и вице-президентом по финансам (ведущей учет в проекте) Сандари Копленд образовали группу управления проектом "Суматра", согласно описанию в разделе "Ключевые участники» плана проекта. За исключением Сайсли и Сандари, которые приглашались по необходимости, группа управления проектом "Суматра» собиралась еженедельно для обсуждения хода проекта.

Цели проекта были определены в разделе "Резюме плана проекта" плана проекта "Суматра".

Ознакомившись с планом проекта "Суматра", Джамал знал аудиторию и цели проекта.

№ этапа	Этап	Выполнено?
1.	Определить аудиторию, которая обычно включает всех лиц, заинтересованных в процессе тестирования и в качестве системы. Определить цели проекта.	☑

Группа проекта "Суматра" должна подготовить и передать обновление продукта SpeedyWriter 3.0, выпустив экспериментальную версию SpeedyWriter 3.1. Свойства "Суматры" включают в качестве начальных мер проекта устранение всех известных ошибок, критичных для заказчиков, а также поддержку необязательной подсистемы управления документооборотом. Группа проекта "Суматра" должна выявить отдельные свойства и расставить для них приоритеты в спецификации системных требований "Суматры". В ходе проекта разрешается вводить изменения в рамках проекта, включая добавление, изменение и удаление свойств. Приоритеты таких изменений должны определяться группой управления изменениями проекта "Суматра".

Группа проекта "Суматра" должна передать окончательное множество свойств для выпуска и продажи в срок к началу второго квартала будущего года. Кроме того, группа проекта "Суматра" должна завершить проект, превысив бюджет не более чем на 10%. Для обеспечения передачи продукта в соответствии с целями по срокам и бюджету группа управления проектом "Суматра" может произвести изменения в рамках проекта, включая, если это необходимо, упрощение или удаление свойств.

Группа проекта "Суматра" должна реализовать множество свойств и подготовить версию для передачи, которая, по мнению руководства проекта "Суматра" и высшего руководства компании Software Cafeteria, будет удовлетворять разумным ожиданиям качества пользователей и заказчиков. Более того, группа проекта "Суматра" должна гарантировать, что все ключевые риски качества системы, как для новых, так и для существующих свойств продукта SpeedyWriter, снижены до приемлемого уровня. Чтобы обеспечить передачу качественного продукта, руководство проекта "Суматра» может произвести изменения в рамках проекта, включая, если это необходимо, упрощение или удаление свойств.

План проекта озвучивает цели проекта, в том числе цели тестирования. Для того чтобы помочь аудитории понять, достигнуты ли эти цели, Джамал должен ответить на четыре глобальных вопроса.

1. В каком состоянии находится качество системы "Суматра", если измерять его при помощи результатов тестирования?
2. Где мы находимся с точки зрения покрытия критических рисков качества системы, и что сообщают нам результаты тестирования об оставшихся рисках?
3. Идет ли проект тестирования гладко и эффективно?
4. Удается ли выполнять запланированные тесты в срок?

Первые два вопроса относятся преимущественно к системе, в особенности к изменению ее качества во времени. А два заключительных вопроса больше касаются процесса, в первую очередь проблем сроков и бюджета. Тем не менее, Джамалу было известно из опыта, что качество системы, переданной группе тестирования, сильно зависит от способности группы выполнять запланированные тесты в срок и в рамках бюджета. Поэтому при подготовке отчетов о ходе тестирования Джамал должен был дать от-

веты на эти четыре вопроса при помощи четырех таблиц, которые несложно построить и понять.

№ этапа	Этап	Выполнено?
2.	Определить результаты, которые должны быть представлены. Обычно это информация, которая должна отвечать на вопросы аудитории о тестировании, особенно о том, что означают результаты тестирования для целей проекта.	☑

Совместно с группой управления проектом Джамал установил графики хода работ и отчеты о ходе тестирования. Они отобрали несколько графиков на основе материалов, применявшихся Джамалом в предыдущем проекте SpeedyWriter 3.0. Теперь вернемся к тестированию большой сборки.

Последний раз мы видели Джамала в субботу, 21 декабря. Группа тестирования только что завершила первый полный цикл тестирования. Джамал обновил отчет о результатах тестирования. Он отправил по электронной почте набор графиков и инструментальную панель тестирования группе управления проектом, а также занес их в проектный репозиторий.

Утром в понедельник, 23 декабря, Джамал представлял отчет о ходе работ на еженедельном совещании, посвященном ходу работ в проекте. Джамал заметил, что на совещание пришли генеральный директор Software Cafeteria Раджеш Гупта, менеджер технической документации Речел Вудз и менеджер поддержки пользователей Косимо Неграев, а также Сайсли Райс и Сандари Копленд. Это хорошо, подумал он, поскольку это первое совещание, на котором можно сделать выводы о состоянии большой сборки. Однако никто из пяти новых членов совещания не участвовал в определении инструментальной панели тестирования. Джамал отметил для себя, что нужно будет рассказать про саму инструментальную панель, когда он будет говорить о ходе тестирования.

Ход тестирования был третьим вопросом в повестке дня, после отчета Дженни Кауфман о ходе разработки и доклада Ясухиро Канагавы о подготовке версий. Когда они завершили свои выступления, Кейт Эрнандес, председательствовавшая на совещании, обращаясь к Джамалу, сказала с улыбкой: "Итак, Джамал, вот момент, которого мы все ждали. Как выглядит большая сборка с точки зрения качества?

Джамал, который только что присоединил свой ноутбук к проектору, открыл титульный слайд "Состояние комплексного и системного тестирования. Инкремент 3 (большая сборка). Сборка 1".

"Итак, – начал он, – прежде всего я бы хотел выразить признательность сотруднице группы Кейта Ли, который сделал все возможное и невозможное для того, чтобы цикл состоялся. Петра любезно согласилась прийти в субботу и помочь тестировщикам, а затем помогла мне установить новую версию по окончании тестового цикла. Кейт, пожалуйста, поблагодари Петру от моего имени".

"Сделаю, – ответил Кейт.

Джамал начал отчет о ходе работ следующими фразами: "Вы спрашиваете, как там большая сборка? Чтобы дать ответ, я покажу четыре традиционных графика, которые, в честь наших гостей, я буду пояснять по

мере того, как мы будем их исследовать. Я также готов по окончании совещания более подробно рассказать о любом из графиков".

"В целом тестирование идет хорошо. Но есть ряд существенных соображений. Я скажу об этом, когда мы будем рассматривать инструментальную панель с результатами тестирования".

"В ходе начального обсуждения вопроса о том, какую информацию должно предоставлять тестирование, мы пришли к соглашению, что еженедельные отчеты о ходе работ по тестированию должны давать ответы на четыре вопроса, – сказал Джамал, переходя к следующему слайду (см. рис. 15.1). "Во-первых, в каком состоянии находится качество системы "Суматра", если измерять его при помощи результатов тестирования? Ответ на этот вопрос дает итоговая кривая на этом графике".

"График показывает среднее число вновь открываемых и закрываемых ежедневно отчетов об ошибках для каждого тестового цикла, а также среднее число незакрытых отчетов об ошибках".

"Незакрытых? – прервал его Раджеш.

"Это разница между общим числом открытых и закрытых отчетов об ошибках, – пояснил Джамал. Раджеш кивнул в ответ.

"Этот график мог бы выглядеть как множество мелких символов, большинство которых соответствуют естественному отклонению процесса, поэтому я показываю средние значения ежедневных цифр вместо самих ежедневных цифр. Поскольку мы получаем версии еженедельно, здесь также представлены ежедневные средние значения открытых, закрытых и незакрытых ошибок для каждого тестового цикла, в котором выполнены все тесты для каждой версии.

Кроме того, здесь показано общее число открытых и закрытых отчетов об ошибках, а эти две кривые – ежедневные итоги. Неровности обуславливаются процессом тестирования. Скачки в общем числе открытых ошибок в пятницу происходят из-за исследовательского тестирования, в ходе которого обнаруживается больше ошибок, чем при помощи плановых скриптов. Скачки в понедельник, а иногда и во вторник в общем количестве закрытых ошибок происходят благодаря успешному проверочному тестированию исправлений. Я показываю общее количество и средние значения за неделю в разных масштабах. Понятен ли смысл графика?" Все кивнули.

"Хорошо, – Джамал сделал паузу, а потом продолжил. – Итак, график показывает нам несколько моментов, связанных с качеством. Во-первых, количество вновь обнаруженных ошибок остается значительным, но управляемым источником риска в проекте. Большое количество вновь обнаруженных ошибок должно ожидаться в этой точке проекта, поскольку появляется новая функциональность. Далее, количество новых ошибок стабильно в течение последних четырех недель и колеблется между 20 и 30 в день. Для управления наплывом новых ошибок я предлагаю не передавать разработчиков, завершивших программирование новых свойств "Суматры", в новые проекты разработки, как это происходило в недавних проектах. Я предлагаю оставить всех разработчиков, в настоящее время занятых в проекте "Суматра", на обозримый период для исправления ошибок.

Во-вторых, количество исправленных и отложенных ошибок выглядит неплохо. Общее число закрытых отчетов об ошибках незначительно

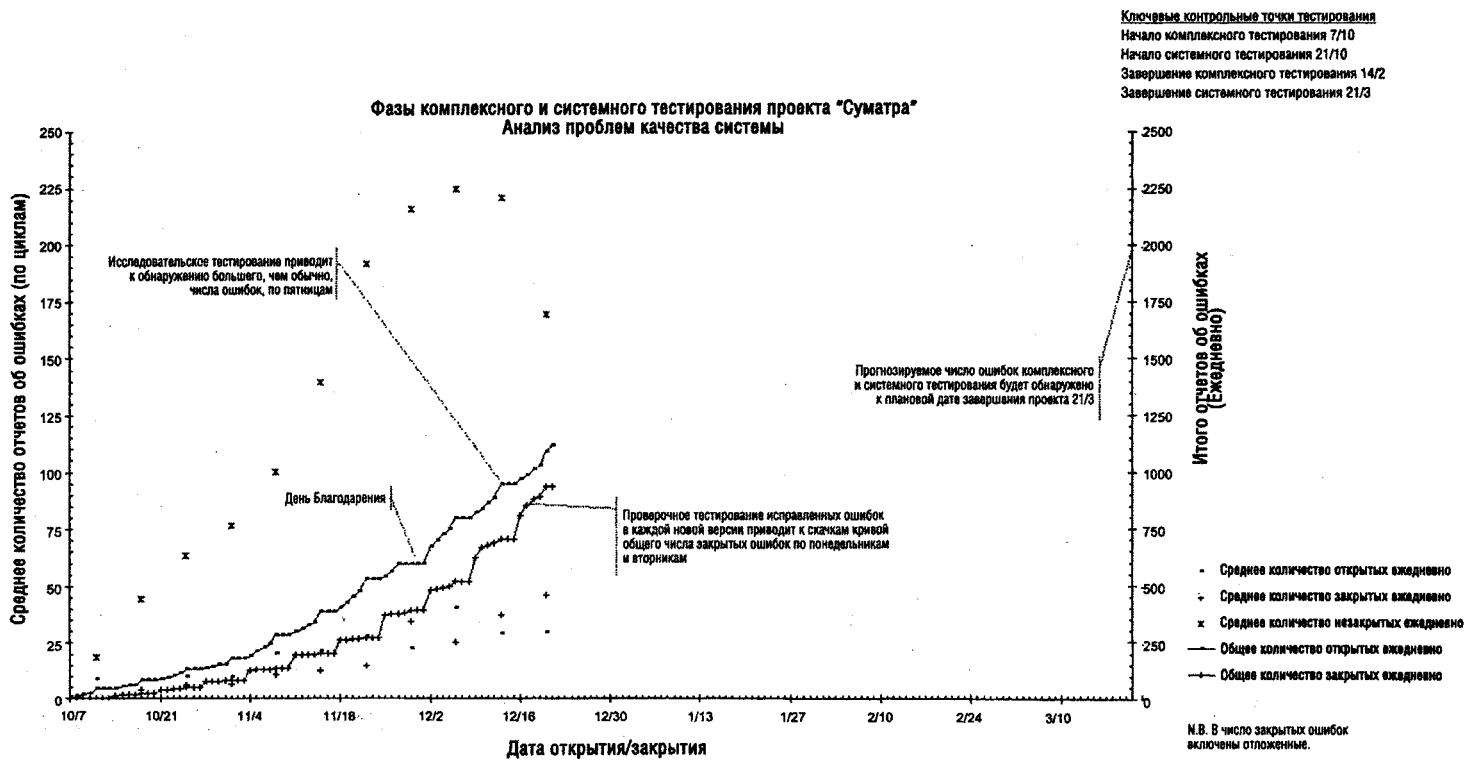


Рис. 15.1. Оценка качества, представленная на графике открытых и закрытых ошибок

отличается от общего числа открытых отчетов. Незакрытые ошибки сейчас находятся под контролем, причем среднее число ежедневно закрываемых ошибок превосходит количество ежедневно открываемых ошибок в течение последних трех недель. В результате сильно сократилось число незакрытых ошибок, которое выглядело настораживающим в последние четыре недели.

В-третьих, чтобы быть предельно объективным, скажу, что некоторые из незакрытых ошибок временные. Количество незакрытых ошибок и разрыв между кривыми открытых и закрытых ошибок рисует чрезмерно пессимистичную картину. Обе кривые растут в течение недели, поскольку многие исправления ошибок включаются в следующую сборку, ожидая тестирования и подтверждения закрытия. Я провел анализ, который показывает, что в среднем ошибка откладывается или закрывается в течение 12 дней после появления отчета о ней. Это означает, что большая часть ошибок будет исправлена в одной из следующих двух версий. И ошибки действительно исправляются. Только примерно одна ошибка из 20 не проходит проверочное тестирование, и ни один отчет об ошибке не проходит проверочное тестирование более чем дважды".

"Хм, — сказала Сайсли, задумчиво проводя рукой по волосам, — ты можешь отслеживать число повторно открываемых ошибок?"

"Да, наш инструмент управления дефектами собирает эту информацию, которая также очень полезна".

"На моей предыдущей работе мы называли это "степенью рецидива, — сказала она с улыбкой.

"Не хочу обидеть твоих бывших коллег, Сайсли, но я стараюсь по возможности избегать таких потенциально сильных выражений".

"Между прочим, мы это ценим, — вставила Дженни.

Джамал с улыбкой посмотрел на Дженни и продолжил свой доклад: "Наконец, вы можете видеть, что мы уже слегка перевалили через половину прогнозируемого общего числа ошибок, которое равно 2000. Мы выполнили почти все тесты по крайней мере один раз, поэтому, вероятно, получим несколько меньшее число ошибок. Другими словами, система оказалась лучше, чем ожидалось".

"Приятно слышать, — пробормотала Кейт Эрнандес.

"Да, приятно, — согласился Джамал, — но неудивительно. Напомню, мой прогноз делался на основе исторических данных предыдущих проектов в Software Cafeteria. Мы внесли некоторые усовершенствования в процесс по итогам последнего проекта, поэтому и получаем чуть лучшие результаты".

"Я очень надеюсь на это, — шепнул Гарольд.

Джамал согласно кивнул и перешел к следующему слайду (см. рис. 15.2). "Следующий вопрос, на который может ответить тестирование: выполнили ли мы тесты, нацеленные на проверку рисков системы качества, и что нам говорят эти результаты об оставшихся рисках? Данный график обобщает наше представление о современном состоянии этих областей риска.

В начале проекта я содействовал проведению анализа рисков с привлечением сотрудников разных направлений, в котором участвовали многие из сидящих в этой комнате. Мы определили риски и расставили приоритеты для различных возможных проблем качества. Мы сопостави-

ли соответствующие уровни тестирования на основе уровней технических и бизнес-рисков".

"Джамал, не мог бы ты напомнить о двух типах рисков? — попросила Сайсли.

"Конечно, — ответил Джамал. — Технический риск — это вероятность существования ошибки в самой системе и ее влияния на систему. Бизнес-риск — это вероятность того, что данная ошибка окажет влияние на работу пользователей и заказчиков, и вероятность влияния этой ошибки на пользователей и заказчиков. Понятно?

"Конечно, — улыбнулась Сайсли. — Спасибо, Джамал".

"Пожалуйста, — Джамал продолжил. — В ходе проектирования и разработки тестов мы количественно оценивали связь между каждым тестовым сценарием и каждым отдельно взятым риском качества. Связь между тестовым сценарием и риском качества может быть сильно, средней, слабой или несуществующей, что мы представили числами 9, 3, 1 и 0 соответственно. Завершив первоначальное проектирование и разработку тестов, мы смогли вычислить общее покрытие каждого отдельного риска качества и просуммировать его по категориям рисков. Используя эти цифры для каждой категории рисков, мы можем сказать, какое значение покрытия будет получено при выполнении 100% запланированных тестов относительно категории риска качества. Серые столбцы на графике показывают текущий процент тестов, выполненных для каждой категории рисков качества. Разумеется, каждый серый столбец должен достичь 100%, хотя пропущенные тесты и исследовательское тестирование могут дать нам чуть меньший или чуть больший результат".

"Что с математической точки зрения означает чуть меньше и чуть больше? — спросила Сандари.

"Диапазон плюс-минус 10%, вероятно, был бы приемлем, — ответил Джамал. — Я был очень осторожен в оценках, поэтому ожидаю, что мы достигнем 100% покрытия по большинству рисков, и только немногие будут покрыты меньше, чем на 95%. Поскольку мы повторяем каждый тест примерно от двух до семи раз в рамках процесса регрессионного тестирования, даже такое низкое покрытие, как 50%, будет указывать на то, что каждый тест был выполнен по крайней мере однажды".

"А что если, не дай Бог, сроки будут отложены и будут проведены два дополнительных цикла тестирования? — продолжала Сандари.

"В этом случае мы превысим покрытие. Я подсчитал числовое значение покрытия, которое представляет 100%-ное запланированное покрытие для каждого риска качества, на основе тестов, которые мы планировали выполнить за день, когда мы начали первый цикл комплексного тестирования. Если мы добавим тесты, то увеличим покрытие. Это уже произошло с определенной частью исследовательского тестирования, которое увеличило покрытие для некоторых рисков. — Джамал с любопытством осмотрел присутствующих. — Понятны ли проблема тестового покрытия и его графическое представление?" Все закивали.

"Итак, выполняя тесты, мы находим ошибки, — продолжал Джамал. — Отчет о каждой ошибке мы классифицируем по категориям рисков качества. Это позволяет нам понять, где мы обнаруживаем ошибки с точки

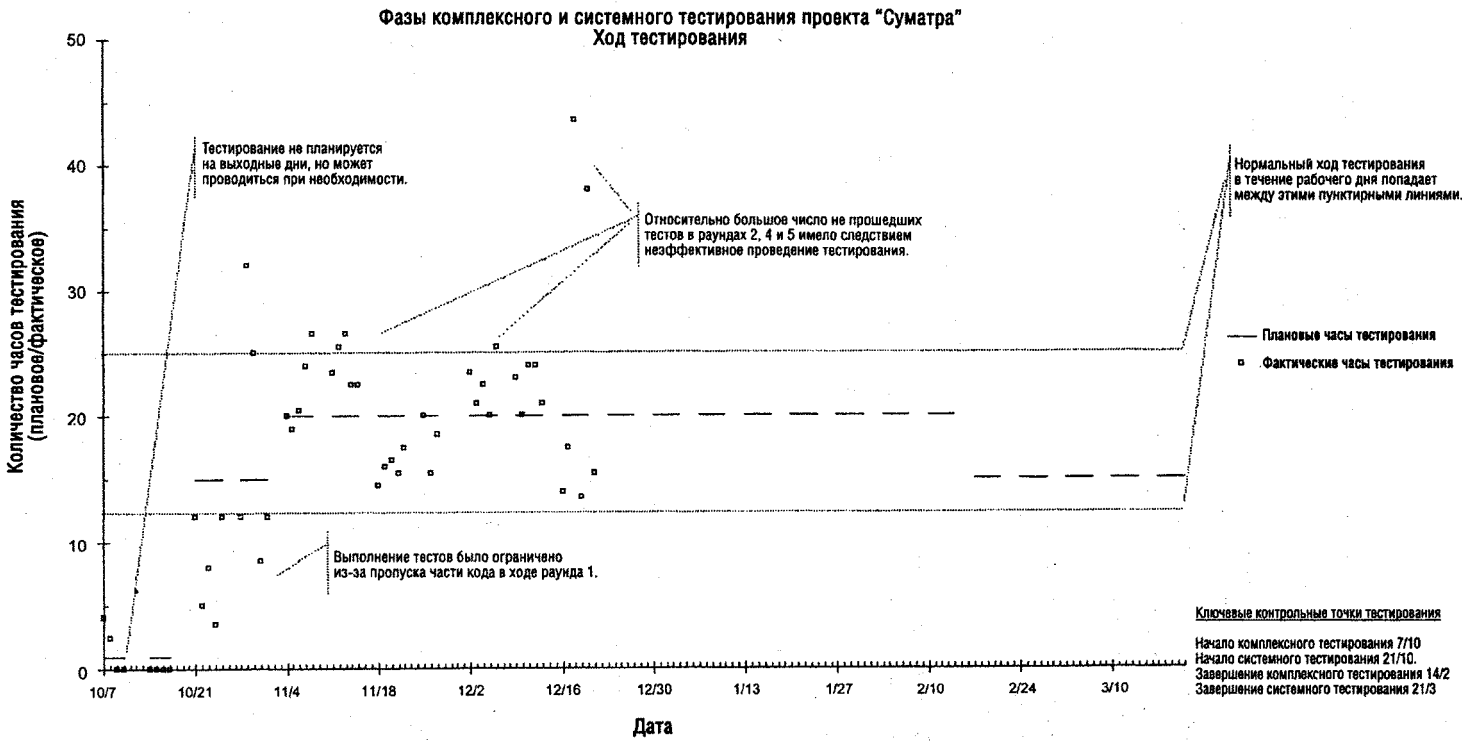


Рис. 15.2. Тестовое покрытие и результаты оценки рисков качества

зрения нацеливания тестов на проверку рисков качества. Белые столбцы на графике показывают процент ошибок по категориям".

"Таким образом, эти столбцы никогда не покажут 100%? – спросил Раджеш.

"Верно, – ответил Джамал. – Они дадут 100% все вместе, но отдельный столбец может достичь 100% только в очень маловероятном случае, когда мы будем находить ошибки, относящиеся только к одной категории рисков качества".

"Позвольте мне пройти по тестовому покрытию и обнаруженным ошибкам для каждого риска качества слева направо. Первые четыре пары столбцов говорят нам одно и то же про первые четыре области рисков. Нагрузка, функциональность, обработка ошибок и надежность покрыты примерно наполовину выполненными тестами и являются наиболее значительными источниками ошибок. Вместе на них приходится примерно половина всего количества ошибок. Итак, мы выполнили довольно много тестов в этих областях и нашли много проблем, но мне кажется, что мы этого и ожидали. Мы можем ожидать, что найдем еще некоторое количество проблем в этих областях, но не думаю, что там скрываются какие-то большие сюрпризы. Как я уже упоминал ранее, мы должны сохранить людей, занятых исправлением ошибок, чтобы управлять проблемами, которые мы уже нашли в этих областях и которые будем продолжать находить, но, по моему мнению, мы сейчас хорошо представляем себе эти риски.

"Будем продолжать находить? – спросил Раджеш.

"Да, – пояснил Джамал, – обычно там, где обнаружено много ошибок, вы найдете еще ошибки. Правило, справедливое для огромного числа проектов, заключается в том, что ошибки обычно концентрируются определенным образом. Один из вариантов, когда ошибки скапливаются в определенных категориях рисков качества. Таким образом, я полагаю, что тестирование позволит найти примерно половину остающихся ошибок именно в этих четырех областях".

"Интересно, – прокомментировал Раджеш. – Я было подумал, что вы должны в каждой категории в определенный момент "вычерпать все до дна".

"Да, это действительно кажется логичным, – согласился Джамал. – Однако правило скопления ошибок, противоречащее интуиции, давно наблюдается тестировщиками программного обеспечения"[1].

"Перейдем к следующей области риска. Тестирование интерфейсов в основном уже завершено. Мы нашли несколько ошибок, большинство в настоящее время закрыто".

"Поэтому комплексное тестирование было успешным? – спросила Кейт Эрнандес.

---

[1] Это правило, на которое содержатся намеки в книге Фреда Брукса "*Мифический человеко-месяц*", впервые, насколько мне известно, было сформулировано в явном виде Гленфордом Майерсом в "*Искусстве тестирования программ*". Позднее Роберт Сабурин и Ким Дэвис выполнили интересную работу по исследованию скопления ошибок и по применению этих знаний в тестировании. Прочитайте их статью "Exploring, Discovering and Exterminating Bug Clusters in Web Applications, которая доступна на www.amibug.com.

"Комплексное тестирование будет продолжаться еще два месяца, пока мы не получим последний инкремент. Однако уже сейчас можно сказать, что оно было успешным, поскольку дало нам много возможностей для исправления ошибок, и мы использовали эти возможности. Фаза системного тестирования не проходила бы столь гладко сейчас, если бы не было успешного комплексного тестирования и исправления найденных ошибок, в котором мы все участвовали, — заключил Джамал.

"Работа основных функций и сопровождение проверены столь же подробно, как и первые четыре категории рисков, с чуть меньшим числом обнаруженных ошибок. Меня устраивает ситуация в этой области рисков, снова с учетом того, что мы оставим основные ресурсы для исправления ошибок, которые, я уверен, мы найдем в этих областях.

Совместимость и качество данных протестированы довольно подробно. Я не ожидаю никаких неприятных сюрпризов в этих областях, хотя мы обнаружили и еще найдем серьезные ошибки в области качества данных".

Дженни кивнула и добавила: "Да, тестирование качества данных, проведенное твоей группой, практически спасло нас. Мы никогда бы не нашли некоторые из ошибок в ходе модульного тестирования".

"Согласен, сложность тестовой среды и некоторые усложненные сценарии использования системы в области качества данных имели большое значение, — ответил Джамал и продолжил свой доклад.

"Производительность и безопасность проверены так же подробно, как и первые четыре элемента, но ошибок было найдено вчетверо меньше. Я думаю, мы выглядим неплохо в этих областях риска, хотя я настаиваю на завершении всех запланированных тестов. С точки зрения бизнес-рисков эти компоненты очень важны. Даже если бы мы не нашли до сих пор ни одной ошибки, я бы все равно предложил продолжить тестирование этих областей".

"Ты целиком и полностью прав, Джамал. Лучше переборщить с тестированием производительности и безопасности, — согласился Косимо. — Эти две области, пожалуй, наиболее болезненно отражаются на звонках в нашу группу поддержки".

"Правда? — спросил Раджеш, удивленно поднимая брови.

"Да, — ответил Косимо. — Я не могу привести никаких цифр, однако скажу, что по крайней мере раз в неделю у меня возникают серьезные проблемы в одной или в обеих этих областях".

"Понимаю, что это не входит в нашу повестку дня, — извинился Раджеш перед Кейт и Джамалом, — но, — продолжил он, обращаясь к Косимо, — не мог бы ты классифицировать звонки в службу поддержки в соответствии со списком рисков качества Джамала и сообщить мне, что получится?

"Конечно, — ответил Косимо. — Мы можем запланировать проведение такого анализа после выхода версии Суматры, чтобы посмотреть, что происходит с точки зрения анализа рисков, но мы также можем начать делать это прямо сейчас".

"Прости за то, что прервал тебя, Джамал, — сказал Раджеш с улыбкой, завершая обсуждение с Косимо.

"Я думаю, что это было вполне к месту, — ответил Джамал и продолжил. — Исследование удобства использования практически завершено. Мы думаем, что здесь не осталось ничего опасного. Отличная работа твоей группы по разработке интерфейсов пользователей, Дженни".

Дженни кивнула, улыбаясь.

"Тестирование локализации только началось, — продолжал Джамал. — Пока мы не обнаружили ничего плохого, но мы еще не проделали достаточной работы. Спросите меня через пару недель, и я буду гораздо лучше представлять ситуацию. Сейчас же могу лишь сказать, что "мы не знаем, чего мы не знаем".

"Две недели? — спросила Сандари.

"Да, — ответил Джамал, — мы не хотим тратить кучу денег на повторное тестирование локализации, поэтому запланировали только два раунда ближе к концу. Эти тестировщики не работают за небольшую плату".

"А нельзя было сделать это самим? — упорствовала Сандари.

"Мы могли бы это сделать внутри компании, но мне бы потребовался значительно больший бюджет, чтобы нанять специально выделенный штат полиглотов и экспертов по стандартам. Языки, совместимость со стандартами, визуальное представление дат, времени, валюты и т.д. и т.п. — это все труднее сделать, чем кажется".

Сандари пожала плечами и признала свое поражение, сказав: "Я догадываюсь, что это не бесплатный обед".

"Конечно не бесплатный шведский стол, будь уверена, — с улыбкой подтвердил Джамал. Он продолжал: "Тестирование документации выполнено наполовину, ничего серьезного там не обнаружено, верно, Речел?

"Верно, — согласилась менеджер документации.

"Обработка дат и недостаточная конкурентоспособность также, похоже, имеют невысокий риск, поскольку мы уже предприняли основные шаги по его снижению, — подытожил Джамал. — Вот где мы находимся с точки зрения рисков. Есть ли вопросы?

Увидев, что нет вопросов, Джамал перешел к следующему графику (см. рис. 15.3). "Следующий вопрос к тестированию — измерение качества продукта и хода проекта тестирования: сколь гладко и эффективно идет проект тестирования?

"Не вижу того, что говорило бы нам о качестве системы, — прокомментировал Раджеш.

"Если тестировщики не могут рационально и производительно пройти тестовые сценарии для проверки поведения системы, которые, как правило, являются аналогией действий пользователей, то каким же окажется опыт пользователей и заказчиков? — ответил Джамал.

"Да-а-а-а ... " — только и смог сказать Раджеш, потирая ухо и заново изучая график.

"Я показываю, сколько часов в неделю отводится на выполнение тестов, исходя из запланированного для каждого цикла тестирования. Это непрерывные линии. Заметьте, что у нас не планируется работа в выходные, в результате чего у этих линий есть разрывы. Прямоугольники показывают, сколько часов в день фактически тратится на тестирование. Количество часов учитывается только тогда, когда тест завершен (прошел, не прошел или предупреждение), и никаких дополнительных часов не мо-

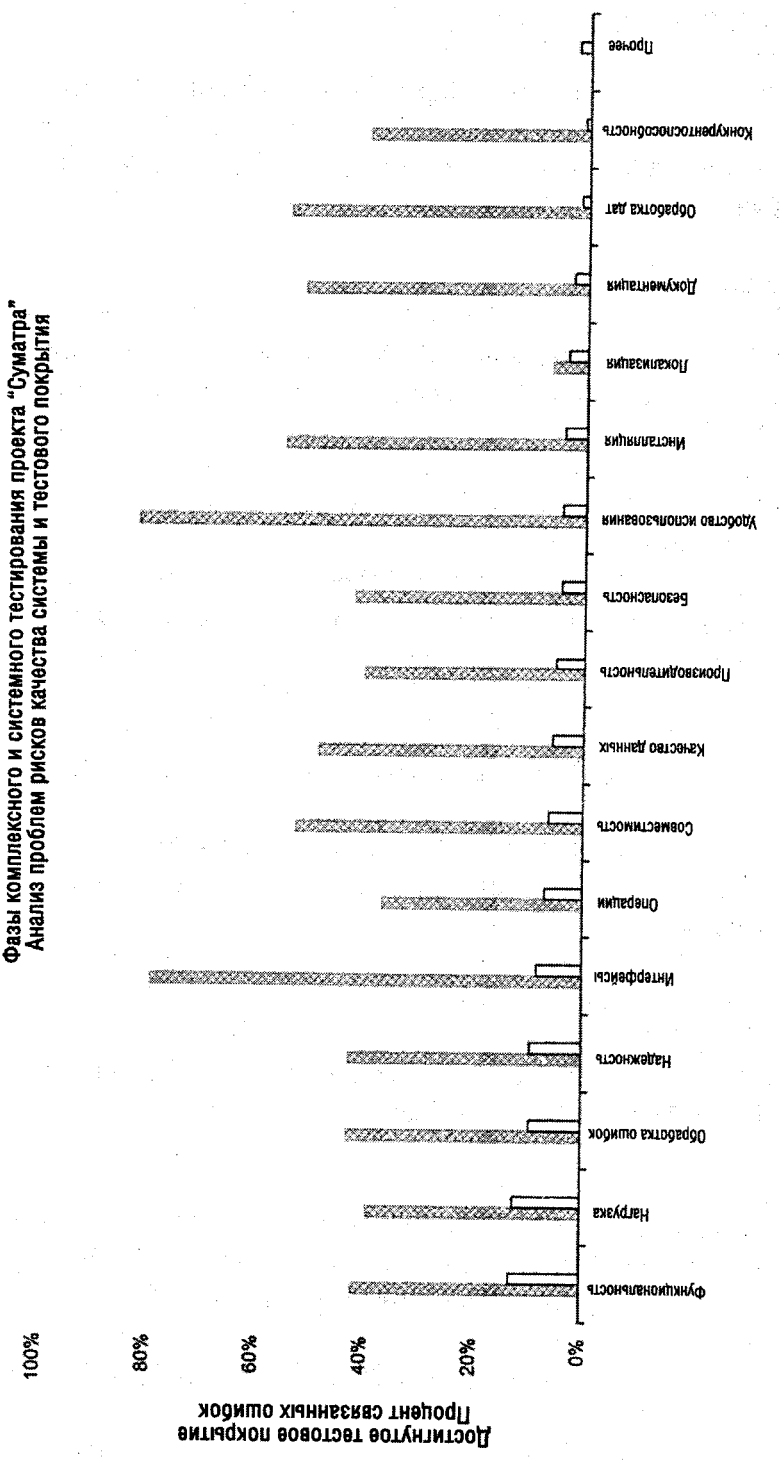


Рис. 15.3. Эффективность использования продукта и тестирования

жет быть потрачено на этот тест до тех пор, пока вновь не будет запланировано его выполнение", – сказал Джамал.

"Это означает, что часы на выполнение теста могут быть распределены больше, чем на один день, но будут учтены только в последний день? – спросил Раджеш.

"Верно, – сказал Джамал. – Возможно, это не совсем верно, но это упрощает учет. Поскольку средние затраты на выполнение теста меньше, чем один день, картина достоверная".

"Понятно", – согласился Раджеш.

"Как вы видите из графика, раунды 2 и 4 были в целом не очень эффективны, то же можно сказать о раунде 5. Относительно большое число тестов завершилось неудачей в эти раунды, что привело к тому, что продолжительность раундов была несколько большей. Мы полагаем, что это также отражает общую неуклюжесть продукта, связанную с количеством незакрытых ошибок. По мере того как число незакрытых ошибок сокращалось, мы видим, как график начал сходиться в области между пунктирными линиями, ограничивающими "нормальную" эффективность, – подвел итог Джамал.

"Таким образом, этот контрольный график – именно то, что ты хотел сказать? – спросил Гарольд.

"О, нет, это не контрольный график в смысле статистического управления процессом, – быстро ответил Джамал. – Сейчас у меня нет таких данных. Пунктирные линии показывают что-то вроде ежедневного отклонения, которого я ожидаю от тестовых сценариев, занимающих дни целиком".

"Итак, – спросил Раджеш, – сколько тестов в настоящий момент не прошли без ошибок?

"Хорошо, что ты спросил, – с улыбкой сказал Джамал. – Теперь я могу перейти к следующему графику (см. рис. 15.4). Этот график дает ответ на четвертый и последний вопрос: удается ли выполнить запланированные тесты в срок? Он также сообщает нам некоторые сведения о качестве системы, хотя и не в явном виде.

Как и на предыдущем графике, сплошные линии показывают плановые тесты, а квадратики – фактическое выполнение тестов. Это накопительный график, показывающий как плановые, так и фактически цифры общего числа выполненных за день тестов для каждого раунда тестирования. В конце каждого раунда мы снова начинаем с нуля. Как вы видите, квадратики довольно близко подходят к сплошным линиям, т.е. мы довольно хорошо выполняем запланированные тесты, хотя и немного неэффективно, как было показано на предыдущем графике".

"Маленькие минусы показывают число неуспешно завершенных тестов, а маленькие плюсы – число успешно пройденных тестов, и снова итоговые цифры для каждого раунда. Я ожидаю, что определенное число тестов будет заблокировано недостающей функциональностью. Вот почему первые четыре линии общего числа запланированных тестов короче, чем остальные. Праздник Дня Благодарения делает третью линию общего числа запланированных тестов особенно короткой".

"А где на графике Рождество? – спросил Ясухиро.

"Здесь на лицо плюсы группы тестирования с сотрудниками, относящимися к разным культурам, – сказал, улыбаясь, Джамал. – Тестировщи-

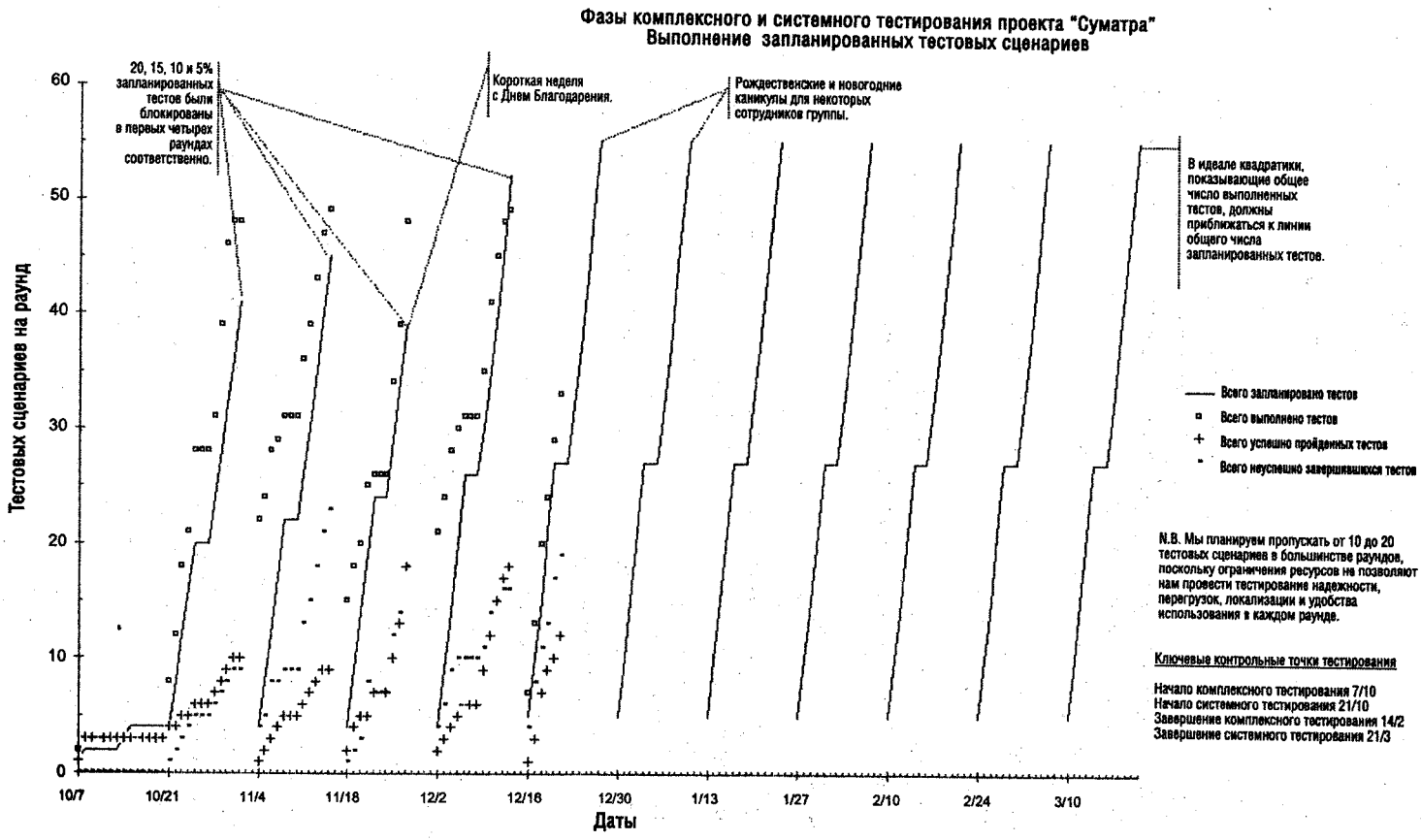


Рис. 15.4. Выполнение тестов

ки, которые не отмечают Рождество, согласились перенести каникулы на другое время. Поэтому мы будем проводить тестирование на рождественской неделе. Я уже предупредил ключевых участников из других групп, к кому могут обратиться за помощью. Дженни, я не знаю, сколько программистов будет работать на той неделе, но если немного, то, вероятно, будет незначительное увеличение по незакрытым ошибкам в течение очередных двух циклов, либо программисты догонят по ошибкам, обнаруженным в течение рождественской недели".

"Мне не повезло, как Джамалу, решить проблему каникул, — ответила Дженни. -- Вероятно, будет небольшой разрыв в устранении незакрытых ошибок".

"Джамал, я не совсем понимаю, почему ты сказал, что график не говорит о качестве явно, — заметила Сайсли. — Ты показываешь число прошедших и непрошедших тестовых сценариев".

"Число тестовых сценариев — штука загадочная, — ответил Джамал. — Легко привести случай, когда эти цифры не имеют никакого отношения к качеству. Я могу объединить два неудачно завершившихся тестовых сценария и получить один сценарий, либо я могу разбить один прошедший тестовый сценарий и получить два успешных сценария. Я могу добавлять тесты по ходу тестирования. Мы редко получаем к концу проекта цифру "100% прошло успешно", поскольку мы обычно принимаем решение не исправлять все ошибки, что означает, что некоторые тесты не прошли. Поскольку мы как тестировщики занимаемся поиском ошибок, мы стремимся выполнять тесты, которые находят ошибки. В то же время это означает, что процент неудачно завершившихся тестов часто рисует слишком пессимистичную картину качества системы. Поэтому некоторое число неудачно завершившихся тестов вполне ожидаемо".

"Каково обычное число неудачно завершившихся тестов при завершении системного тестирования? — спросил Раджеш.

"Не могу точно вспомнить, — ответил Джамал, — но в двух последних версиях, по-моему, было 15–20%. Если нужно, я могу уточнить цифру".

"Не нужно. И так все довольно ясно, — сказал Раджеш. — Нам еще далеко до завершения".

"Да, сейчас непрошедших тестовых сценариев примерно вдвое больше, — ответил Джамал. — Это подводит нас к контрапункту. Сейчас я могу довольно быстро показать, что подсчеты тестовых сценариев также имеют некоторое значение для качества системы. Поскольку мы проектируем тесты, нацеленные на проверку поведения системы, они имеют некоторое отношение к действиям, которые могут выполнять пользователи, и к вариантам поведения, которые пользователи могут наблюдать. Поэтому большое количество непрошедших тестов действительно дает нам информацию о качестве системы. Это похоже на живопись в стиле пуантилизма. Если вы будете смотреть с очень близкого расстояния, то увидите точки, которые сами по себе ничего не значат. Как только вы отходите, вы начинаете видеть отчетливую картину".

№ этапа	Этап	Выполнено?
3.	Отобрать измерения и составить отчеты и графики, которые отвечают на поставленные вопросы.	☑

"Что тебя беспокоит, Джамал", – спросил Раджеш.

"Во-первых, у нас остается несколько областей значительного риска, о которых мы либо знаем недостаточно, либо знаем о наличии в них серьезных проблем. Сейчас мне кажется, что в ближайшие две-три недели эти области высокого риска будут протестированы, а обнаруженные ошибки будут исправлены. Однако мы должны следить за этим. В ближайшие недели очень большое значение имеет разбор ошибок, поскольку я думаю, что будут найдены новые существенные проблемы. Мы пока еще не подошли к тому моменту, когда обнаруживают только заключительные небольшие ошибки".

Раджеш кивнул.

"Во-вторых, я кратко говорил о застарелых проблемах в Software Cafeteria, когда мы забирали разработчиков, занятых исправлением ошибок, обнаруженных в ходе системного тестирования, и назначали их в новые проекты. Это были, как правило, опытные разработчики, которым в новых проектах ставились задачи проектирования. В результате исправление ошибок концентрировалось в руках тех людей, которые нередко дестабилизировали архитектуру и саму систему, – сказал Джамал. – Я не собираюсь обижать группу Дженни. В проекте "Суматра" работает не один человек. Но есть разные уровни квалификации. Раньше мы говорили, что все мы работаем с тривиальными заключительными ошибками, и соответственно уменьшали число людей и общую квалификацию персонала. Но на самом деле мы работали с существенными ошибками, которые имели далеко идущие последствия. Это выливалось в большие проблемы, когда нужно было соблюдать сроки, а сложности возникали как раз в конце проекта".

"Замечание принято, – сказал Раджеш. – Продолжай, пожалуйста".

"В-третьих, как я говорил, и число неудачно завершившихся тестов, и среднее время выполнения тестового сценария пока выше тех значений, которые я бы хотел видеть в конце проекта. Я не сомневаюсь, что эти цифры улучшатся через два месяца, но снова скажу, что за этими индикаторами надо внимательно следить. Если к началу февраля они не начнут снижаться, это будет означать, что у нас есть проблема. Еще раз, это связано с необходимостью сохранения подходящей комбинации уровня и квалификации персонала в группе Дженни к моменту завершения проекта.

Наконец, напомню, что суммарная цифра закрытых дефектов на первом графике включает в себя ошибки, отложенные до следующей версии, той, что появится после Суматры. Их число выросло, а число незакрытых ошибок уменьшилось, поскольку на совещаниях группы управления изменениями мы приняли решение, что ошибки приоритета 2 могут быть отложены".

Раджеш вопросительно посмотрел на Кейт и Дженни, которые сидели рядом.

Кейт приняла вызов и сказала: "Я думаю, что пользователи воспримут правильно нашу агрессивную политику по откладыванию ошибок. Эта политика является частью стремления передать версию, которая обладает всеми функциональными возможностями высшего и среднего приоритета, вовремя. Реалии таковы, что другим вариантом является незаконченная финальная итерация, поскольку очень много программистов занято исправлением ошибок".

"Мне понятен этот компромисс, – ответил Раджеш. – Но давайте не будем сильно снижать качество и передавать свойства с массой ошибок. Это не имеет смысла". Кейт и Дженни не успели ничего возразить, Раджеш повернулся на стуле и бросил взгляд на Джамала: "Итак, господин тест-менеджер, каково ваше мнение об этой "агрессивной политике откладывания ошибок"?

Джамал посмотрел Раджешу в глаза и сказал: "Не хочу уклоняться от вопроса, господин генеральный директор, но я вижу свою основную роль в том, чтобы оценивать качество и сообщать о результатах оценки. Я не являюсь полицейским качества. Поэтому как господин тест-менеджер я остаюсь нейтральным к этой политике".

Раджеш почесал бороду и, продолжая пронизывающе смотреть на Джамала, спросил: "А что думает господин Джамал Браун как мыслящий человек и законный владелец акций компании?

"Лично я предпочел бы несколько повысить планку качества. Однако, – согласился Джамал, – этого всегда хочется. Что я вижу постоянно в течение рабочего дня – это проблемы качества. Мы находим 100–150 ошибок еженедельно. Это искажает мое восприятие, делает меня чувствительным к рискам качества системы. Поскольку я сосредоточен на этих рисках, вероятно, я не вижу других. Риски сроков и бюджета, которые видят Дженни, Кейт и Сандари, не столь очевидны для меня.

Честно говоря, Раджеш, – подвел итог Джамал, – я думаю, чтобы определить, не занизили ли мы планку качества, тебе придется изучить все или, по крайней мере, большое количество ошибок, которые были отложены. Если у тебя нет времени на это, может быть, Косимо выступит в роли твоего полномочного представителя. Если ты или он увидит ошибки, которые вас пугают, то, возможно, нам нужно будет прекратить откладывание исправления ошибок, чтобы слегка приподнять планку. Я не располагаю всей информацией о том, как заказчики и пользователи отнесутся к нашему продукту с учетом заявлений компании о приоритетах ошибок и об их разборе".

"И это ты хотел сказать мне в моем кабинете за закрытыми дверями, Джамал?

"Да", – твердо ответил Джамал.

"Хорошо, когда будет следующее совещание группы управления изменениями? – спросил Раджеш.

"30-го, в понедельник, – ответила Кейт. "Мы отменили три совещания на этой неделе, поскольку не можем обеспечить присутствие сотрудников разных групп".

"До совещания я просмотрю отложенные ошибки, и я буду на совещании со списком ошибок, исправление которых надо будет возобновить, – сказал Раджеш. – У нас все с ходом тестирования?

"Да, – сказал Джамал.

"Хорошо. Отличная презентация. Очень понятная, – подвел итог Раджеш. – Что у нас дальше в повестке дня?

№ этапа	Этап	Выполнено?
4.	Представить результаты тестирования аудитории в соответствии с требованиями.	☑

Построение успешного процесса подготовки отчетов о результатах тестирования

В нашем примере Джамал использует удобный процесс для еженедельной отчетности о ходе тестирования. Что мы можем сказать в общем виде о работе группы тестирования и рабочей группы проекта и о преимуществах, которые они получают от удобного процесса подготовки результатов тестирования?

Своевременная передача полезной, заслуживающей доверия информации

В качестве аналогии процесса тестирования и подготовки отчетов о его результатах рассмотрим газету. Газета с обозрением местных новостей в других городах, а не в том, в котором вы живете, бесполезна. Газета, не рассматривающая финансовые и деловые проблемы, бесполезна для управления вашим портфелем инвестиций. Газета на непонятном вам языке бесполезна. Газета, которой вы не доверяете, бесполезна. Вчерашняя газета почти бесполезна, а газета месячной давности представляет лишь исторический интерес. Я полагаю, что специалисты-тестировщики, считающие себя репортерами по теме качества, и ведущие тестировщики и тест-менеджеры, которые считают себя издателями газеты о качестве, могут использовать это сравнение для определения направления подготовки отчетов о результатах тестирования.

При обдумывании вопроса по поводу отчетов о результатах тестирования следует спросить себя: мы, группа тестирования, готовим нужные отчеты, в нужное время, с нужной периодичностью и передаем через подходящие каналы? Еще лучше, если вы спросите об этом ваших менеджеров и коллег, всех лиц, заинтересованных в процессе тестирования и качестве системы.

Считаю важным последовательное определение информации, которая должна быть получена в процессе тестирования, совместно с коллегами и руководителями. В нашем примере Джамал построил четыре графика, которые составили инструментальную панель проекта тестирования. Эти графики были получены в результате следования процессам, определенным ранее, начиная с целей проекта и ключевых заинтересованных лиц и заканчивая отдельными измерениями, представленными графически.

Одна из сложностей состоит в том, что руководители и коллеги могут испытывать затруднения при определении метрик, которые должны измеряться группой тестирования, а вы, по всей вероятности, столкнетесь с проблемой отсутствия собственных идей, если впервые занимаетесь проблемой организации процесса подготовки отчетов о тестировании. К счастью, существует немало хороших источников идей. Одним из них является данная глава, а также главы 4 и 5 моей предыдущей книги "*Managing the Testing Process, Second Edition*". Стефан Канн в "*Metrics and Models in Software Quality Engineering, Second Edition*", Борис Бейзер в "*Software System Testing and Quality Assuranse*" и Панкодж Джалот в "*CMM in Practice*" обсуждают различные метрики тестирования и контроля качества программных

продуктов. Анна Эллисон написала полезную статью "Meaningful Metrics", которую можно прочитать на www.stickyminds.com. Другие интересные источники вдохновения перечисляют в библиографическом списке и упоминаются в сносках этой главы.

Джамал представил то, что обычно называют *инструментальной панелью тестирования* или *системой сбалансированных показателей*, т.е. он кратко описал ход тестирования. Этот перечень отчетов подходит для совещания, посвященного ходу тестирования, но коллеги-менеджеры, возможно, обратятся за более подробными сведениями. Например, менеджер технической поддержки или менеджер разработки, вероятно, захочет подробно изучить отчеты об ошибках либо на регулярном совещании, либо в ходе неформального обсуждения. Один вариант не подойдет всем. Получатель информации, а не ее создатель определяет ее значимость и пользу.

Точность — это другой важный элемент, характеризующий полезность информации. Базовые проектные решения часто принимаются на основе результатов тестирования. Должны ли мы задержать выпуск продукта? На что надо потратить дефицитные ресурсы в конце проекта? Точным должно быть не только представление данных (отчетов об ошибках, результатов тестирования, трассировки покрытия и т.п.) в виде краткого отчета (т.е. на инструментальной панели), но и анализ, который проводится на основе этих данных. Наконец, эффективное озвучивание результатов означает, что вся аудитория получила в ходе презентации точное понимание смысла отчета о ходе работ по тестированию.

Когда вы рассказываете о ходе тестирования, вам необходимы навыки публичных выступлений для того, чтобы суметь донести смысл ваших идей. Одно время я страдал от боязни сцены. Иногда я делал краткие или оборонительные комментарии, представляя результаты тестирования публично, что приводило к меньшей эффективности моих презентаций. Когда я нервничал, я забывал рассказать о ключевых деталях, оборонительно реагировал на открытые и честные вопросы, демонстрировал нервозность, что отвлекало аудиторию. Всему, что я знаю о публичных выступлениях, я научился методом проб и ошибок. Я уже много лет рассказываю о результатах тестирования, а также выступаю на конференциях, на сайтах для клиентов и на семинарах. Сейчас существует множество книг, учебных курсов и компаний, которые помогут вам овладеть ораторским искусством. Я был бы счастлив, если бы в свое время мог ими воспользоваться.

Обычно информацию, поступающую из проекта тестирования, лучше всего представить графически, как это сделал Джамал, или при помощи таблиц. Широкое распространение и простота использования таких инструментов, как Word, Excel, Visio и Access (называю только несколько примеров, с которыми я лично знаком), с одной стороны, дают возможность провести сложный анализ и подготовить представление информации.

С другой стороны, применение этих инструментов может привести к появлению пугающего числа документов, графиков, таблиц и отчетов. Бурный рост псевдоинформации может задавить проект, внутри несущественных данных могут прятаться небольшие фрагменты информации, критичной для проекта. Число человеко-лет, которые были потеряны,

Как потратить впустую время и деньги на бесполезные отчеты о состоянии дел

Я работал в одном проекте, в котором три высокооплачиваемых консультанта из большой фирмы трудились с полной загрузкой над подготовкой пакета ежедневных отчетов о ходе работ по тестированию. Допустим, что часовая ставка составляет $200 в час у каждого, что весьма умеренно, и подсчитаем. Получается примерно $5000 в день.

Проект шел по непонятной траектории в течение двух месяцев, после чего спонсор проекта решил пригласить новую команду консультантов по тестированию в лице вашего покорного слуги. Прежде всего я спросил менеджера проекта, какие графики и отчеты из ежедневного пакета о состоянии дел она использует в качестве ключевых индикаторов. Хотя я предполагал, что лишь часть отчетов является действительно полезной, ее ответ ошеломил меня:

"Я их не читаю совсем, я просто выбрасываю эти проклятые бумажки в корзину".

Я поспрашивал других участников проекта. Ответ был таким же.

По моим оценкам, по крайней мере $200 000 в том проекте вылетели в трубу из- за подготовки отчетов, которые никто не читал. Что еще хуже, проект выполнялся без всех тех многочисленных преимуществ, которые предоставляет программа тестирования. Возврат инвестиций в тестирование сократился до обнаружения и исправления ошибок, которое происходило на тактическом уровне между тестировщиками и разработчиками. Единственной хорошей новостью в этом проекте было то, что три консультанта на 100% были заняты подготовкой ежедневных отчетов. Следовательно, работа, которую они выполняли по подготовке бесполезного пакета отчетных документов, не отвлекала их ни от каких других дел.

и число проектов, которые потерпели неудачу из-за того, что их менеджеры рисовали бесполезные таблицы и графики, неизвестно, но, без сомнения, очень велико.

Я начал понимать эту проблему, когда стал изучать вопросы дизайна, в частности, web-сайтов. Наибольшую помощь оказала отличная трилогия Эдварда Тафта (Edward Tufte), посвященная дизайну и представлению информации. "*The Visual Display of Quantitative Information*", "*Envisioning Information*" и "*Visual Explanations*" вдохновили меня на более успешную работу по созданию графиков и отчетов по тестированию. Рекомендую всем эти книги. Комментарии Тафта по выбору подходящих цветов, штриховок и меток, а также его критика лишних и непонятных значков на графиках и в диаграммах, оказались весьма полезными для меня.

Точность данных и информации, эффективные навыки устного общения, тщательно продуманные графики, отчеты и другие документы позволяют сохранять доверие к результатам тестирования. Я также вижу, что чем выше поднимаешься по иерархии руководителей, тем больше значение имеет подача информации. Я не предлагаю делать информа-

цию кричащей. Напыщенные мультимедийные звуки и видеоклипы, вращающиеся графические фрагменты обычно не информируют, а отвлекают. Только потратив больше времени для того, чтобы понять потребности аудитории в информации, и подготовив действительно работающее и лаконичное представление данных, удовлетворяющее эти потребности, можно получить хорошую презентацию.

Своевременность и периодичность – это также элементы полезности результатов и доверия к ним. Коллеги хотят получать самую свежую и менее обработанную информацию как можно чаще, а более высокопоставленные руководители, включая высших, ожидают менее частого поступления отчетов о тестировании. Руководители рассчитывают, что отчеты будут краткими и информативными. Внимание старших и высших руководителей к ходу тестирования мне представляется как топливный бак. У меня есть ограниченный запас топлива, и я не хочу его израсходовать прежде, чем завершится проект.

Наконец, выбор правильных каналов для передачи информации оказывает влияние на полезность, своевременность информации и доверие к ней. Возможны следующие каналы для распространения информации:

- Совещания о состоянии дел в проекте
- Совещания по *разбору ошибок* (bug crawl)
- Внеплановые совещания, посвященные обсуждению критических ошибок, препятствующих продолжению работы
- Совещания группы управления изменениями (см. главу 16)
- Совещания, посвященные началу фазы тестирования (известные, как обзор готовности продукта к тестированию)
- Совещания, посвященные завершению фазы тестирования
- Совещания по выпуску, поставке версии или принятию решения о возможности продолжения работы
- Телеконференции
- Электронная почта
- Внутренняя сеть компании
- Индивидуальные беседы

Способ, вид, степень детализации и презентация информации могут существенно отличаться при использовании различных каналов. Что подходит для одного канала (полезно, своевременно и укрепляет доверие), может совершенно не подходить для других каналов, иногда вызывая серьезные проблемы, связанные с доверием к специалистам-тестировщикам. И, наоборот, внимательное отношение к передаче полезной и своевременной информации нужным людям при помощи соответствующих каналов определяет положительное отношение и укрепляет доверие к вам.

Связь состояния тестирования с ходом проекта и его влияние на будущее проекта

Другое направление укрепления доверия – это увязывание состояния тестирования и хода проекта в целом, а также планирование шагов по про-

движению вперед. Если мы завоевали доверие, люди будут следовать предлагаемым шагам, возможно, не в точности, как мы хотим, но мы будем оказывать влияние на ход проекта. Если мы не обладаем необходимым доверием, люди пренебрежительно отнесутся к нашим идеям и не будут доверять нашим оценкам хода тестирования и проекта в целом.

Эффективные отчеты о результатах тестирования — это не только точные данные. Это не только красивые графики, отчеты и таблицы. Это не только вдумчивый и подробный анализ. Это не только уверенная и убедительная подача информации. Это не только правильный выбор канала передачи. Да, все это вместе — необходимо, но это лишь тактика. Наша цель — *эффективная передача информации* о ходе тестирования лицам, заинтересованным в тестировании. Другими словами, информацию специалиста-тестировщика *слушают*, ее *слышат*, *понимают* и, если этого требует информация, *действуют согласно ей.*

Мой опыт показывает, что для обеспечения эффективной передачи информации о результатах тестирования, я должен обернуть ее в такую упаковку, чтобы увязать с проектом и его целями. Например, что я могу сказать, если цели проекта — поставка развлекательной видеоигры с некоторым набором нетривиальных свойств в виде архивированной прикладной программы для DVD в рамках прибыльного бюджета и разумных сроков? И что я должен сказать, если цели проекта — поставка бездефектной системы для проведения лучевых процедур больных раком? Очевидно, что одинаковое представление результатов тестирования, даже количественно одинаковых результатов, — это не просто плохая идея, это этически неприемлемо.

Кроме того, я должен спросить себя, чего ожидает компания от завершенного проекта тестирования и что хочу сделать я. Отчет о результатах — это то место, где в письменной или устной форме, графически или при помощи некоторой комбинации методов озвучиваются достижения. Если одно из достижений, которого я пытаюсь добиться, может заставить проектную команду действовать на основе полученной от меня информации, т.е. может оказать влияние на ход проекта, то отчет о результатах — это главный этап процесса влияния.

Если это так, я должен предусмотреть, как группа проекта отреагирует на результаты, о которых я доложу. Нельзя допускать, чтобы отсутствие определенных проблем на ранних этапах проекта навело на мысль, что и далее проблемы не будут найдены. И наоборот, если обнаруживается масса проблем и проект испытывает трудности, необходимо определить возможные пути выхода из кризиса. Под "связью с ходом проекта" и "путями выхода из кризиса" не обязательно понимать "выпуск версии в срок во что бы то ни стало". Напротив, я понимаю это как определение разумного множества компромиссов и решений о равновесии сроков, бюджета, свойств и качества в рамках широкого контекста технологии, проекта и компании.

Это особенно касается незапланированных обсуждений, например внепланового совещания по обсуждению критической ошибки. Вспомним взрыв космического челнока "Челленджер". Трагедия этой катастрофы вышла за пределы обсуждения гибели на глазах всего мира семи человек ясным и холодным утром. Более глубокая трагедия заключалась в том

факте, что за день до того, как это случилось, инженеры Morton Thiokol предоставили информацию о случайной связи между понижающейся температурой и вероятностью проблем с уплотнительными кольцами руководителям запуска и официальным лицам NASA. Если бы эта информация была подана правильно, то почти наверняка запуск был бы отложен. Это не означает, что инженеры ошиблись. Инженеры правильно оценивали информацию. Но руководители запуска и официальные лица не поняли ее смысла. К сожалению, ни одна из сторон не подумала о ликвидации этой коммуникационной проблемы.

В книге "*Visual Explanations*" Эдвард Тафт, используя многочисленные источники, в том числе презентацию Ричарда Фейнмана комиссии американского конгресса, переработал подачу информации. Хотя я не обладаю обширными знаниями в науках о материалах, графики Тафта очень четко объясняют, что может случиться с уплотнительными кольцами при холодной погоде.

В публичной лекции по материалам этой книги, включая обсуждение катастрофы с Челленджером, Тафт говорил о трех основных уроках, связанных с представлением данных.

1. Покажите причинно-следственную связь.
2. Покажите все существенные данные.
3. Покажите руководителям то, что они действительно должны увидеть, чтобы принять решение на основе полученной информации.

Подводя итоги, Тафт спросил: "Какова задача для размышления, в решение которой мы хотим вовлечь человека [который заслушивает отчет]?"[1]

Конечно, катастрофа Челленджера – это тот случай, когда инженеры могли более эффективно донести суть проблемы. Однако бывают случаи, когда тестировщики перегибают палку и представляют ситуацию в очень негативном свете. Рассмотрим следующие варианты представления хода тестирования.

Оба этих варианта описывают один и тот же момент времени в одном и том же проекте. Первый вариант – это отчет о ходе тестирования, который связан с целями и задачами проекта и предлагает путь решения проблем. Второй отчет упорно и слишком сильно напирает на отрицательные стороны. Это может привести к тому, что отчет и сам докладчик будут проигнорированы как сеющие панику и безысходность. Итак, оба отчета описывают одну и ту же ситуацию в проекте, но первый отчет связан с ходом проекта и, по всей вероятности, окажет влияние на дальнейшие действия[2].

---

[1] Анализ катастрофы Челленджера, проведенный Тафтом, с большим перечнем дополнительных источников для дальнейшего чтения можно найти в "*Visual Explanations*", стр. 38 – 53. Три упомянутых урока обобщают текст на первой странице главы 1 "Мастерство графического представления информации" в "*The Visual Display of Quantitative Information*". Публичная лекция, на которой я присутствовал, состоялась в Остине, штат Техас, 28 января 2002 г. Во время написания этой книги расследование разрушения челнока "Челленджер" продолжалось.

[2] Касаясь более широкой темы, как стать влиятельным тестировщиком, рекомендую прочитать статью Джоанны Ротман "The Influential Test Manager", которая была впервые опубликована в журнале "*Software Testing and Quality Engineering*", Volume 2, Issue 2 (Mar/Apr 2000) и доступна в настоящее время на www.stickyminds.com и www.jrothman.com.

Отчет 1. По состоянию на прошлую субботу мы выполнили все из 517 запланированных тестовых сценариев. В настоящее время система управления дефектами показывает 247 отчетов об ошибках, которые либо не закрыты, либо назначены разработчику, но еще не исправлены. На этой неделе мы нашли только 10 новых ошибок, в отличие от трех предшествующих недель, когда мы обнаруживали по 40 ошибок. Однако, поскольку скорость исправления ошибок с начала системного тестирования составляет примерно 30 ошибок в неделю, плановая дата окончания системного тестирования (через три недели) находится под большим вопросом. Вероятно, мы должны провести совещание по подробному разбору дефектов, чтобы отложить проблемы, которые, по нашим ощущениям, не являются критическими для данной версии. После этого обсуждения мы, по всей вероятности, должны уточнить график работ.

Отчет 2. Мы завершили выполнение всего комплекта тестов, и что за неразбериха. Группа тестирования продолжает находить ошибки, и на большинство из них никто не смотрел. Мы находим каждую неделю больше ошибок, чем разработчики могут исправить. Не вижу способа, который бы позволил нам завершить системное тестирование через три недели.

Демонстрация тенденций развития качества и возможных результатов

Объединение проблем хода тестирования и проекта в целом подводит нас к следующему признаку качества процесса подготовки отчетов о ходе тестирования: там, где имеет смысл, необходимо показывать тенденции и возможные результаты.

Первые результаты тестирования в проектах разработки, сопровождения, конфигурации и интеграции, вероятно, покажут некоторое текущее состояние качества. По мере развития проекта качество должно, по идее, улучшаться. Состояние качества меняется со временем. По мере того как происходят эти изменения, развиваются тенденции. Тенденции, если их правильно представить, могут дать информацию не только о том, в каком состоянии находится проект, но и куда он идет. Следовательно, информация о тенденциях позволяет руководителям прогнозировать, каким будет качество в будущем. В каком состоянии проект с точки зрения устранения ошибок? Когда система будет готова к поставке?

В нашем примере инструментальная панель Джамала показывает тенденции, а также оценки того, куда движется проект. Сглаживающаяся кривая совокупного числа открытых ошибок демонстрирует стабильность или, по крайней мере, неспособность группы тестирования найти еще больше ошибок при помощи существующей системы тестов. Кривая совокупного числа закрытых ошибок, сходящаяся с кривой открытых ошибок, указывает на качество — решение проблем, обнаруженных при тестировании. График выполнения тестов показывает выполненные тестовые сценарии (тесты, прошедшие без ошибок, и тесты, завершившиеся неудачей) и невыполненные тестовые сценарии. Данный график позво-

ляет руководителям увидеть, сколько еще нужно выполнить тестов, какова доля тестовых сценариев, которые не могут быть выполнены, и соотношение между успешно и неуспешно завершившимися тестами.

Представление тех же данных, которое не дополнено динамикой изменения значений и прогнозом развития ситуации, вероятно, будет лишено элемента предвидения. Помните, что одним из путей, при помощи которого тестирование увеличивает стоимость компании, является сопровождение проекта своевременной, точной и заслуживающей доверия информацией о ходе проекта. Информация о тенденциях и возможных результатах имеет существенное значение для сопровождения проекта.

Конечно, есть такая аудитория, которая интересуется прежде всего состоянием на текущий момент. Подробные тактические отчеты для разработчиков, бизнес- аналитиков, персонала технической и сетевой поддержки пользователей должны, вероятно, быть направлены на вопросы, требующие немедленного реагирования. Чтобы предоставить эти сведения, можно использовать краткое описание тестовых сценариев, комплектов тестов, дефектов и даже полные отчеты об ошибках. Краткий отчет о тестовых сценариях может содержать перечень сценариев по одному на строку, включая текущий статус, связанные отчеты об ошибках, протестированные конфигурации и т.д. Краткий отчет об ошибках может показывать ошибки по одной на строку, включая номер, краткое описание, дату открытия, серьезность и прочую полезную информацию.

Существует одно исключение из правила, связанного с необходимостью показывать тенденции: если вы проводите приемо-сдаточные испытания единственной версии, для которой не являются проблемами вопросы интеграции и конфигурирования. В этом случае наиболее подходящим способом отчета о результатах тестирования является, по всей вероятности, моментальный снимок: либо система подходит для использования и готова к поставке, либо предложение поставщика не готово. Оценку нескольких конкурентных продуктов от разных поставщиков, из которых делается выбор, также удобно проводить при помощи моментального снимка текущего состояния.

Применение подходящих механизмов и документов

В нашем примере Джамал использовал показ слайдов в качестве стиля подачи информации и устное пояснение каждого графика и его значения в качестве механизма подачи результатов. Слайды содержали графики, построенные при помощи программы обработки таблиц. Он также применял электронную почту для предварительной рассылки таблиц участникам совещания, чтобы они могли при желании подготовиться к совещанию. Это один из способов использования механизмов подготовки и доведения результатов, причем он не является необычным.

В моем методе управления проектами тестирования подготовка кратких отчетов о результатах тестирования начинается с определения выполненных тестов, сниженных рисков, найденных и исправленных ошибок и взаимосвязей всех этих элементов с запланированными или ориентировочными измерениями и оценками завершения. Я организую проект тестирования на основе структурированной и измеряемой систе-

мы тестов с тщательно спроектированными и разработанными тестами, четкой взаимосвязью между рисками качества и тестами и с формализованными базами управления дефектами. Из этой системы тестов я извлекаю имеющие смысл измерения и суммарные данные. Некоторые тест-менеджеры более подробно документируют системы тестов, чем это делаю я, некоторые комфортней себя чувствуют с меньшим количеством документации. Частично связь результатов тестирования и контекста, как обсуждалось в главе 1, должна использоваться для повышения эффективности работы системы тестов в вашей компании.

Я также люблю численно измерять результаты тестирования. При этом я получаю более глубокое представление, чем только при качественном подходе. Количественная оценка ключевых индикаторов лежит в основе моего понимания состояния тестирования. Я вижу, что мои клиенты, коллеги и члены группы тестирования обычно более уверенно себя чувствуют, если результаты имеют количественную оценку. Процитирую Лорда Кельвина: "Если... вы не можете выразить [то, о чем вы говорите] в числах, ваши знания скудны и имеют неудовлетворительное качество"[1].

Работая с количественными оценками результатов тестирования, я считаю важным для себя выделить из информации значимые данные. Я обычно получаю поток данных тестирования, который меняется ежедневно. Просто вывалить кузов самосвала с данными на людей — это не передача информации, это только собьет их с толку. Вот почему процесс, описанный выше, связывает измерения тестирования с соответствующим контекстом, с целями проекта и с вопросами по поводу этих целей.

Соображения эффективности получения информации и обмена ею также имеют значение. В идеале вы можете обобщить сведения о состоянии дел с тестированием автоматически и безо всяких усилий при помощи инструментальной панели и других отчетов о ходе тестирования. Получатели отчетов должны уметь проникать в легко доступные источники данных об ошибках, ходе тестирования, покрытии рисков. (По различным причинам юридического характера вы, вероятно, можете закрыть доступ к определенным данным о процессе тестирования, особенно, если у вас есть ощущение, что они могут быть неверно интерпретированы, сбить с толку или даже умышленно указать неверное направление.)

Подход одного из моих крупных клиентов состоял в размещении инструментальной панели тестирования в локальной сети компании. Все графики автоматически и одновременно обновлялись на регулярной основе, на основе извлечения исходных файлов данных из стандартного репозитория. Как только процесс был запущен, он сразу показал свою эффективность, хотя в создание инструментов и процесса было вложено всего три человеко-месяца.

---

[1] Более подробно о преимуществах различных измерений, связанных с ошибками и тестированием, можно прочитать в главах 4 и 5 моей книги "*Managing the Testing Process, Second Edition*". Другой взгляд представлен в статье Майка Кона (Mike Cohn) "A Measured Response", опубликованной впервые в журнале "*Software Testing and Quality Engineering*", Volume 4, Issue 5 (Sep/Oct 2002) и доступной в настоящее время на www.stickyminds.com. Наконец, посмотрите статьи Сема Канера "Cem Kaner on Rethinking Software Metrics" на www.stickyminds.com и "Measurement of the Extent of Testing" на www.kaner.com.

Наконец, позвольте мне отметить, что диаграммы и графики, показанные в примере, можно подготовить при помощи таблиц Excel. В качестве альтернативы вы можете вложить средства в разработку своих собственных средств, чтобы создать всеобъемлющий инструмент, как это сделал мой клиент. Однако если вы приобрели или рассматриваете вопрос о приобретении коммерческого инструмента управления тестированием, этот инструмент должен генерировать все необходимые отчеты о ходе тестирования. Вероятно, необходимо сконфигурировать инструмент и совокупность отчетов. Как только это будет сделано, вы будете тратить свои силы только на распечатку подготовленного отчета из базы данных или предопределенного графика из программы обработки таблиц. Если инструмент не способен генерировать отчеты или делает это неуклюже, он не удовлетворяет разумным ожиданиям качества и функциональности профессионалов в тестировании. Если можно, откажитесь от такого инструмента в пользу другого, который умеет делать все необходимое.

Передача информации в удобной для каждого потребителя форме

До некоторой степени я обобщил понятие аудитории. Каждой группе: менеджерам, топ-менеджерам, разработчикам и т.д. — нужна своя информация. Обобщение полезно при определении точки отсчета для измерений и отчетности, но я хочу, чтобы вы исследовали возможность подстройки ваших отчетов под отдельное заинтересованное лицо. Вероятно, вы удивитесь, обнаружив новые и полезные способы, при помощи которых вы сможете своевременно предоставлять нужную информацию проектной группе.

В качестве примера не из области разработки программного обеспечения рассмотрим информационные источники, которые вы цените. Точно ли совпадают ваши интересы со стереотипами, которые существуют относительно вашего пола, возраста, этнической принадлежности, политических убеждений и т.п.? Поскольку я — мужчина, проживающий в южной части Соединенных Штатов (Техасе), согласно стереотипу, я должен читать спортивный раздел газет. Но я этого не делаю. Я никогда не смотрю соревнования профессиональных спортсменов по телевизору. Однако покажите мне статью о том, как приготовить индийскую или китайскую пищу, и я жадно буду ее читать.

Я работал с менеджерами по маркетингу, которым нравилось торчать в тестовой лаборатории, наблюдая за отказами тестовых сценариев. Некоторые менеджеры разработки любили просматривать тестовые сценарии, чтобы убедиться, что модульное тестирование является дополнением к остальным видам тестирования. Некоторые менеджеры проектов ожидали от меня самого свежего отчета о состоянии тестирования даже в ситуациях, которые продолжали развиваться. Другие менеджеры проектов предпочитали получать тщательно проанализированные результаты тестирования.

Вновь отмечу, что основной момент — это передача информации. Поэтому я нахожусь в поиске путей, позволяющих настроить доставку сообщений на волну слушателей. Я готовлю специальные сообщения о ходе

работ, провожу неформальные обсуждения с глазу на глаз и даже посещаю совещания о состоянии дел других менеджеров с избранными отчетами.

Когда я настраиваю сообщение на волну слушателя, то мысленно ставлю себя на его позицию. Что ценит слушатель? Как тестирование служит его интересам? Что я могу сообщить из того, что он хочет узнать для принятия решения о дальнейших действиях? Что и как я должен рассказать для того, чтобы в голове у слушателя в соответствии с его точкой зрения выстроилась четкая картина о ходе тестирования?

Наконец, хотелось бы отметить, что поскольку тестирование часто выступает в качестве услуги компании, важно, чтобы любое взаимодействие группы тестирования и заинтересованных в тестировании лиц укрепляло положительное впечатление от вклада группы тестирования в проект. Это особо касается отчетности о результатах тестирования, которые часто являются неприятными новостями.

Настройка передачи информации на конкретного слушателя демонстрирует уважение к этому человеку, что способствует положительному восприятию работы группы тестирования. Те из нас, кто имеет опыт технической карьеры, должны продумать, как общаться с людьми, далекими от глубокого понимания высоких технологий. Например, можно ли информировать о ходе тестирования или об отдельных проблемах в свете целей проекта, а не в терминах дампов памяти, неинициализированных указателей, неподдерживаемых элементов управления окнами и так далее? Тестировщикам полезно изучать технологии создания систем. Однако необходимо иметь в виду, что многие технические словечки, сокращения, понятия и жаргон неприемлемы для непосредственных исполнителей и менеджеров, которые не являются техническими экспертами. Использование неприемлемых слов не производит впечатления на аудиторию, а только запутывает ее.

Я вовсе не хочу сказать, что представление результатов должно быть покровительственным. Если люди хотят понять сложный технический вопрос, чтобы разобраться в определенной части хода тестирования, фраза "это сложно, вы вряд ли поймете" будет оскорбительной для аудитории и опасной для проекта. Такое сообщение также унизит вас как профессионала, поскольку вы недооцениваете свои коммуникативные способности. Теперь вы, вероятно, согласитесь со мной, что эффективная передача информации — важный навык тестировщика-профессионала, и прямо сейчас отведете время на обучение тому, как делать это лучше.

Решение проблем

Процесс передачи информации сложен по своей природе. Передача информации о ходе тестирования особенно затруднена по целому ряду причин.

Подготовка презентации

Формальные совещания и презентации, в отличие от электронной почты и других неинтерактивных способов отчетности о результатах тестирова-

ния, вероятно, наиболее напряженные мероприятия в плане информирования о результатах тестирования. Однако такие форумы являются наиболее эффективными. Отправка отчета в корпоративную сеть не означает, что все его прочтут. Отправка по электронной почте файла в формате Excel, полного диаграмм и графиков, объемом в 2 мегабайта, не означает, что каждый откроет прикрепленный файл. При любом неинтерактивном способе отчетности о ходе тестирования ваши возможности в получении обратной связи сильно ограничены. Поэтому во многих проектах интерактивное представление результатов тестирования, либо на телеконференциях, либо лично, необходимо делать время от времени.

Новичкам в ритуалах управления, в том числе в проведении совещаний и презентаций, полезно прочитать книгу Патрисии Энсворт "*The Accidental Project Manager*". И новичкам, и людям с опытом я рекомендую книгу Эдварда Тафта "*Visual Explanations*", в которой обсуждаются правила подготовки презентаций. Хотелось бы в дополнение к их идеям, относящимся к базовым навыкам презентации результатов, дать следующие советы, касающиеся прежде всего тест-менеджеров.

- На совещания, посвященные ходу работ, приходите, подготовив слайды и другие сопроводительные документы. Чем выше статус аудитории, тем меньше слайдов я приношу с собой, но тем более тщательно готовлю каждый слайд. Чем хуже новости, тем больше сопутствующих данных я беру с собой и тем тщательнее я готовлюсь. Однако я стараюсь показать плохие новости на небольшом множестве понятных графиков, что концентрирует внимание аудитории на наиболее важной информации. В основном примере Джамал постоянно фиксировал внимание на том, что сообщают графики о качестве системы, предназначенной для тестирования, а не на отвлекающих и непродуктивных вопросах, которые могли бы возникать, например: хорошо ли идет тестирование?
- Демонстрируйте вашей аудитории слайды, графики и отчеты, указывая на хорошие и плохие новости. То, что является очевидным для специалистов- тестировщиков, привыкшим читать такие графики, ничего не значит для коллег- менеджеров и высшего руководства. Что еще хуже, они могут неправильно интерпретировать вашу информацию. В нашем примере Джамал тщательно разъясняет смысл графиков, как в общем (т.е. вопросы, на которые отвечает каждый график), так и в частностях (т.е. каково значение информации, представленной на каждом из графиков, для проекта).
- Принимайте предложения и вопросы с улыбкой, старайтесь не протестовать. Тестировщики (как и разработчики, инженеры сборки и другие непосредственные исполнители) получают кучу предложений в ходе обсуждения состояния тестирования. Я пытаюсь выслушивать, воспринимать и выражаю готовность к обсуждению, но я никогда не спешу изменять мои тщательно выстроенные план тестирования, систему тестов и процессы тестирования в ответ на импровизированное предложение или отклик на совещании о ходе проекта.

Эти идеи могут оказать помощь в начале работы. Тем не менее, я пытаюсь добиться того, чтобы мои глаза и уши, сердце и ум были открыты, ко-

гда я работаю над представлением результатов тестирования. Совместно с моими коллегами я усиливаю и осваиваю стили, которые зарекомендовали себя, и расстаюсь с неудачными стилями. Это особенно актуально в свете следующей проблемы.

Носитель плохих новостей

Легко сообщать людям хорошие новости, которые четко отражают их достижения и не требуют действий с их стороны. Например: "Мы выполнили все тесты и не нашли ошибок, поэтому сегодня мы завершаем системное тестирование согласно графику". Труднее передавать плохие новости, особенно когда из них следует, что тот или иной работник совершил ошибку и должен предпринять действия для устранения последствий.

Реакция на плохие новости — это из области психологии, но не является чем- то неизвестным в сфере программирования. На первой странице главы "Назревание катастрофы" в книге "*Мифический человеко-месяц*" Фред Брукс поместил изображение скульптуры Геракла, убивающего гонца Лихаса за то, что он принес плохие вести. Бедный Лихас. Время от времени многие тестировщики подвергаются аналогичному остракизму.

Помимо иррациональных причин такой реакции на плохие вести и их носителя, часто существуют вполне разумные причины. Прежде чем обвинять руководителей Morton Thiokol и официальных лиц NASA в катастрофе Челленджера, подумайте вот о чем. К руководителям и официальным лицам обратились с предложением отсрочить миссию, в которой был политически заинтересован лично президент США Рональд Рейган. В космическом полете должны были впервые участвовать учитель и женщина. И с точки зрения холодной войны, и с точки зрения равенства полов этот полет имел большое символическое значение в середине 80-х годов. Откладывание полета было бы не только непопулярным решением, это было бы также затратным решением. Наконец, представьте позицию менеджеров и руководителей Morton Thiokol, к которым обращались специалисты с просьбой занять жесткую позицию по отношению к всесильному и выгодному заказчику. Единственное честное решение, которое могли бы сделать менеджеры и официальные лица, так это попросить инженеров доказать их опасения. Аналогично, когда мы предлагаем отложить выпуск версии или потратить еще денег на тестирование, едва ли разумно ждать от руководства похвалы.

Вряд ли можно полностью овладеть искусством подачи плохих вестей. Если тестировщикам и тест-менеджерам удается эффективно подавать плохие новости разумным людям, это следует оценивать как признак высокой квалификации. Чтобы добиться цели в таких ситуациях, требуются мастерство в деталях, подходящий стиль и манера поведения при сообщении новостей. Успех зависит от менеджера и от группы управления, но, по моему мнению, нелишними будут следующие советы.

- Сохраняйте спокойствие и терпение. Логическое представление лучше передает факты, чем эмоциональное. Иногда приходится пояснять какие-то детали разными способами, чтобы всем было понятно. Я рассматриваю передачу информации как часть моей работы даже в проблемной аудитории и в сложных обстоятельствах.

- Будьте тактичны. Я имею доступ к информации, которая может смутить слушателей, унизить коллег, если подать ее необдуманно. Я тщательно выбираю данные, относящиеся к неудачно завершившимся тестовым сценариям и отдельным отчетам об ошибках, поскольку их могут посчитать поводом для резкой критики или обвинения. Бездумно подготовленный отчет может быть использован в качестве оружия против других, даже если я не предполагал этого.
- Не будьте дон Кихотом, одиноким рыцарем качества[1]. Не считайте себя ответственным за качество системы и не соглашайтесь принимать на себя такую ответственность, если это не входит в ваши должностные обязанности. К моим обязанностям обычно относится работа эксперта по управлению рисками качества системы. В контексте подготовки отчетов о результатах тестирования это означает наличие достаточной квалификации для подготовки отчетов по отдельным рискам, отдельным тестам, отдельным ошибкам, а также для определения тенденций и последствий результатов тестирования. Я прилагаю все усилия к тому, чтобы результаты говорили сами за себя, чтобы дать возможность руководству выбрать те ошибки, которые нужно исправить, а программистам внести эти исправления.
- Не радуйтесь ошибкам. Талантливый тест-менеджер Рейнольдс Макнари (Reynolds MacNary) однажды сказал мне: "Будь оптимистом снаружи и пессимистом внутри". Как тест-менеджер, я хочу, чтобы мои тестировщики выполняли тесты, находили ошибки и снижали риски в порядке приоритетов. Я удовлетворен профессионализмом моей группы, когда они работают именно так. Однако непрошедшие тесты, ошибки с высоким приоритетом и опасные инциденты, связанные с рисками, *являются* плохими вестями для проекта. Я считаю, что не следует принимать уж слишком мрачный вид, когда докладываешь о проблемах, особенно в том случае, когда просто говоришь кому-нибудь: "Я рассказал тебе, как обстоят дела".

Здесь основной вопрос -- стремление деперсонализировать презентацию плохих новостей, чтобы четко отделить сообщение о плохих новостях, неудачу в проекте от носителя новостей, профессионала, который хочет сделать своевременное предупреждение.

Для сравнения представьте врача, который должен сообщить пациенту, что он страдает от опасного, возможно, смертельного заболевания. Врач, конечно, помогает больному, сообщая ему о болезни, поскольку больной может сразу же заняться лечением. Вероятно, что врач заинтересован в лечении пациента и в совершенствовании своих навыков в борьбе с недугом. Более того, может статься, что врач в течение нескольких лет предупреждал пациента, чтобы он изменил какие-то привычки, которые привели его к болезни. Однако какими бы ни были обстоятельства,

---

[1] В книге Мигеля Сервантеса "*Дон Кихот*" главный герой страдает от повреждения рассудка, в результате чего он воображает себя странствующим рыцарем. Он защищает честь девушки, которая оказывается проституткой, и бьется в поединке с ветряной мельницей, которая представляется ему опасным великаном. Тестировщики, которым кажется, что они одинокие рыцари качества, часто выглядят такими дон Кихотами.

исключительно неразумно в такой ситуации говорить: "Я предупреждал пять лет назад, чтобы вы бросили курить", предлагать волшебное зелье, преувеличивать возможности в умении лечить такие заболевания или вероятность выздоровления, жестоко обращаться с пациентом в то время, когда он испытывает страдания, или поскорее вытолкнуть его за дверь, чтобы принять следующего.

Для носителя плохих новостей самые большие проблемы возникают, когда необходимо выступать на совещании, на котором принимается решение о выпуске или поставке продукта. Ставки здесь как никогда высоки, и ощущается острая потребность в позитивной информации. Однако если вести нерадостные, голос тестировщика обычно одинок, когда он рассказывает о рисках, относящихся к качеству системы, в то время как другие говорят о давлении сроков, бюджета, свойств, что делает *немедленную поставку* системы абсолютно критической.

Некоторые тест-менеджеры очень хотят иметь право на остановку выпуска системы. Наверное, они думают, что играют в игру с нулевой суммой, в игру, в которой качество важнее всех остальных соображений. Но может ли сотрудник или часть сотрудников, занятых в проекте, достичь успеха, если весь проект провалился? Что касается меня, то в игре, в которую мы играем, сумма может существенно превысить нуль или, наоборот, стать существенно меньше нуля в зависимости от того, принимает ли группа проекта, в особенности группа управления проектом разумное решение по поводу выпуска или поставки системы.

Поэтому я не выступаю в роли полицейского, который следит за качеством. На самом деле я неоднократно сталкивался со случаями, когда мои клиенты хотели возложить на меня эти обязанности. Но я хочу занимать свое место за столом, я хочу быть доверенным и влиятельным советником по вопросу качества. Я абсолютно счастлив, находясь на совещании, посвященном принятию решения о выпуске системы, когда я сосредоточен на представлении четкой, хорошо поданной оценки качества и готовности системы. (Как отмечалось в главе 7, я часто применяю критерии завершения заключительной фазы тестирования, например критерии завершения системного тестирования, в качестве измерительного инструмента.) Я даже осмеливаюсь высказать мнение по поводу текущего состояния рисков качества в сравнении с рисками бюджета, сроков и свойств. Однако в случае, подобном рассматриваемому нами примеру, часто тест-менеджер или тестировщик не располагает всей информацией, необходимой для принятия взвешенного решения по поводу выпуска продукта. В то же время, своевременно предоставляя полезную информацию о качестве, которой можно доверять, тест-менеджер остается незаменимым поставщиком ключевой части информации, которая необходима группе управления проектом для принятия разумного и взвешенного решения. Вот в этой роли я и стремлюсь выступать[1].

Эти советы предназначены для передачи плохих новостей разумным людям. А что делать с неразумными? Есть такие, кто поступает неразумно

---

[1] Для дополнительных размышлений на эту тему ознакомьтесь со статьей "Release Criteria" Джоанны Ротман, впервые опубликованной в "*Software Testing and Quality Engineering*", Volume 4, Issue 2 (Mar/Apr 2002) и доступной в настоящее время на www.stickyminds.com и www.jrothman.com.

без всякого оправдания часто и в ущерб себе, и в ущерб коллегам. Некоторые используют неблагоразумие в качестве стиля управления, особенно это касается применения тактики оказания давления[1].

Некоторые люди, считающие положение с тестированием неприятным, пытаются заглушать или затемнять значение плохих новостей и заодно запугивать и показывать несостоятельность их носителя. В среде, где люди прибегают к этой тактике, я пытаюсь отстаивать свою позицию, сохраняя благоразумие, спокойствие и последовательность. (Если меня никто не слушает или не проявляет благоразумия, то, как показывает мой опыт, лучше просто помолчать.) На тех совещаниях, где используется подобная тактика, я обычно разговариваю с теми, кто слушает, я строю отношения с восприимчивыми коллегами, пытаясь направить информацию в обход тех, кто устраивает обструкцию, сбивает с толку или переходит к персональным обвинениям.

В "*Искусстве войны*" Суньцзы предупреждает читателя, что мудрый охотник не нападает на тигра в его норе, а мудрый полководец не ввязывается в сражение, чувствуя слабость своего положения. Я предлагаю это сравнение не для того, чтобы рекомендовать воинственное поведение со своими коллегами-менеджерами, а для того, чтобы стимулировать к поиску естественных союзников группы тестирования, которые знают статус тестовых сценариев и ход устранения ошибок и принимают участие в этой работе.

Наконец, нужно отметить, что некоторые люди неразумно воспринимают отчеты о результатах тестирования по абсолютно понятным причинам. Возможно, они себя так ведут из-за нереалистичных ожиданий от проекта, сформировавшихся и укрепившихся у них. Для того чтобы сообщения о плохих новостях были результативными, все должны быть готовы к этому с самого начала проекта, когда мы определяем контекст проекта, оцениваем риски качества, делаем оценку затрат в проекте и проводим планирование. Вероятно, люди ведут себя неблагоразумно потому, что они не считают группу тестирования надежным источником полезной и своевременной информации. Трудно наладить и укрепить доверие постфактум, но мы должны сохранять внимание в ходе всего проекта для того, чтобы управлять как качеством работы группы тестирования, так и впечатлением от этой работы[2].

---

[1] О работе с патологически неразумными людьми и с теми, кто поступает неразумно в тактических целях, рассказывается в книге Эда Йордона "*Путь камикадзе*". Неплохое обсуждение проблем, связанных с дополнительным давлением на работу команды и производительность персонала, можно найти в книге Тома де Марко и Тима Листера "Peopleware".

[2] Помимо управления, имеют место также обсуждения с глазу на глаз, которые проходят между тестировщиками и программистами и другими непосредственными исполнителями. В статье Карен Джонсон (Karen Johnson) "Delivering Unwelcome News to Developers", впервые опубликованной в журнале "*Software Testing and Quality Engineering*", Volume 4, Issue 5 (Sep/Oct 2002) и доступной в настоящее время на www.stickyminds.com, приводятся некоторые полезные идеи о взаимодействии на этом уровне.

Постоянно меняющаяся сложная история

Иногда мы можем описать новости как "хорошие" для тестирования. Иногда мы считаем их "плохими". Обычно присутствует комбинация и тех, и других. Но одно прилагательное мы можем присоединить навсегда к новостям о тестировании — "изменяющиеся".

В ходе выполнения тестов постоянно обнаруживаются новые проблемы. Если я потрачу пару часов на подготовку слайдов для совещания по состоянию дел в проекте, то к концу совещания мой отчет будет несвежим. С другой стороны, если я быстро составлю презентацию, то, вероятно, приду на совещание с недостаточно хорошо подготовленным отображением состояния дел в тестировании. Поэтому приходится идти на компромисс между точностью и своевременностью, между самыми свежими данными и ясностью анализа результатов. Как и проблема интерпретации результатов тестирования (см. главу 13), правильный баланс в принятии компромисса зависит от контекста. В данном случае контекст определяется информационными потребностями аудитории.

Интересует ли аудиторию точный подсчет тестовых сценариев, число прошедших и непрошедших тестов и т.п.? Если да, то я должен в течение некоторого времени проверить, что все мои отчеты согласованы, но это означает, что мои результаты устареют к моменту, когда люди их получат. Хочет ли аудитория получить самые свежие результаты, самые актуальные новости, даже если детали неясны? Если это так, я должен сделать процесс подготовки отчетов о результатах тестирования быстрым, несмотря на некоторую их неточность во времени.

Чтобы правильно сбалансировать актуальность и точность отчетов о результатах тестирования, я привлекаю к обсуждению моих коллег-менеджеров и других заинтересованных лиц. Я показываю, сколько времени необходимо для достижения определенного уровня точности и согласованности в отчетах и что написание отчетов отрывает меня от процесса тестирования. Таким образом мне обычно удается получить рекомендации для принятия верного решения. Слово "решения", вероятно, наиболее подходящее. Желания потребителей моих отчетов могут слегка отличаться. Я уже говорил, чтобы умело передавать информацию, необходимо нацеливать каждое сообщение на конкретного слушателя. Хотя форма подачи отчетов о результатах тестирования для группы слушателей адресуется некоторому среднему слушателю, я могу подготовить специально адаптированную информацию для различных заинтересованных лиц. Адаптированные отчеты могут удовлетворить предпочтения отдельных потребителей в отношении сочетания своевременности и точности информации.

Даже при неограниченном количестве времени на подготовку отчетов невозможно все предусмотреть. Я не могу знать все. Например, я не могу по памяти подробно рассказать о каждом из 300—400 незакрытых отчетов об ошибках. Точно так же я не могу ответить на конкретные вопросы относительно определенного условия, спрятанного в одном из 200—300 тестовых сценариев. Когда мне задают такие вопросы, я улыбаюсь и говорю что-нибудь вроде: "Да, ты победил в игре "Поймай тупицу"! Я не знаю ответа на этот вопрос, но я изучу проблему сразу после совещания, если ты не против".

Поскольку я обычно хорошо подготовлен, откровенное признание недочета вместе с готовностью немедленно предоставить информацию всякий раз одерживает победу. Однако я должен быть готов к подробному обсуждению серьезных ошибок, наиболее серьезных сбоев тестовых сценариев и наиболее опасных рисков. Проницательные менеджеры часто задают мне детальные вопросы об обстоятельствах, вызвавших потерю данных или функциональности, появление узких мест производительности или неисправность коронного свойства.

Сочетание переживания за качество с бесстрастным представлением результатов

Как тестировщики-профессионалы, мы обычно концентрируем усилия на проблемах и рисках. Это верно, поскольку часто роль группы тестирования – помочь в управлении рисками системы качества. Однако стремление к этому может превратиться в дон-кихотовское поведение, о чем я писал раньше. Как сохранить высокое стремление к качеству и избежать его превращения в препятствие для проекта?

Прежде всего необходимо осознать, что поскольку мы понимаем, что отчеты об ошибках и информация о неудачно завершившихся тестовых сценариях ставят новые задачи перед программистами, мы, тестировщики, имеем естественное человеческое желание знать, как группа управления проектом оценивает ход тестирования. Во всяком случае, мы хотим, чтобы нашу работу ценили.

Однако важно понимать, что тестирование не происходит само по себе. Компании содержат группы тестирования не для удовлетворения праздного интеллектуального любопытства, а для получения специальных услуг по управлению рисками качества в контексте конкретного проекта. Эти услуги помогают группе управления проектом уравновесить четыре элемента – качество, свойства, сроки и бюджет, – чтобы принимать решение на основе полной информации.

Внедрение усовершенствований

С учетом того, что все тестировщики должны находить точные пути передачи информации о результатах тестирования, каким образом можно внедрить усовершенствования? В широком смысле я могу предложить следующие отправные точки.

1. Начните с вопросов заинтересованным лицам относительно того, что они думают о текущих информационных продуктах и услугах, которые они получают от группы тестирования. Этот шаг сам по себе увеличит доверие к группе тестирования, поскольку он демонстрирует, что вы заботитесь о пользе и своевременности своих результатов.
2. Будьте готовы показать примеры. Вы можете дать легкое и нечеткое описание того, что можно получить из предлагаемых графиков или отчетов, но только примеры, особенно моделирующие реаль-

ные ситуации, дают людям возможность почувствовать, какую информацию они будут получать. Кроме того, вы сможете изучить их реакцию на нее.

3. Делайте шаги разумного размера. Реализация полностью автоматизированной и интегрированной инструментальной панели может быть серьезным шагом. Зачем совершать такие значительные действия, если вы не полностью уверены в том, что именно это вам нужно? Может быть, проведение в течение некоторого времени анализа вручную с использованием средств обработки таблиц, базы данных, программ для построения графиков и т.п. поможет точно выразить все, что требуется.
4. Сохраняйте низкий уровень накладных расходов. Время, которое группа тестирования тратит на анализ данных тестирования, построение графиков, обновление интранет-сайтов и т.д. — это время, в течение которого группа тестирования не проводит оценки качества системы, предназначенной для тестирования, и не занята работой, направленной на повышение качества этой оценки. Предоставляйте только ту информацию, в которой нуждаются люди, и ни графиком больше.
5. Работая над улучшением процесса, непрерывно отслеживайте эффективность процессов подготовки отчетов о тестировании и восприятие эффективности этой работы. Информационные потребности людей и их восприятие работы группы тестирования могут быстро меняться, профессионалы-тестировщики должны быть готовы ответить на эти изменения.

Обсудив подготовку отчетов о результатах тестирования, достигли ли мы последнего этапа в процессе тестирования? Во всяком случае, мы прошли весь путь от определения контекста тестирования до формирования оценки качества и передачи информации об этой оценке группе проекта. И это все, за что нам платят, не так ли?

В некоторых проектах это так, но в большинстве проектов, в которых я работал, был еще один фактор, заслуживающий пристального внимания, — изменения. Определение проекта включает в себя цели свойств, сроков, бюджета и качества системы. Это определение часто меняется по мере выполнения проекта. Изменения иногда указывают на проблемы в выработке четких требований, архитектуры и планирования работ, но они могут также обуславливаться тем, что мы учимся на протяжении проекта. В следующей главе будет показано, как несовершенная подготовка и обучение оказывают влияние на изменение проекта и как эти изменения сказываются на тестировании.

ГЛАВА 16

Увеличение возможностей для обучения: управление изменениями и их влиянием на тестирование

В последних двух главах рассматривался способ, при помощи которого основные достижения тестирования передаются проектной группе. Отчеты об ошибках дают проектной группе вещественные основания для улучшения качества системы или для того, чтобы предпринять соответствующие шаги по снижению влияния известных ошибок, когда те попадают в эксплуатационную версию. Отчеты о результатах тестирования показывают руководству, что некоторые риски, вероятно, ниже, чем ожидалось (т.е. некоторые тесты завершились успешно), и предоставляют менеджерам полезную информацию о проекте в целом, что позволяет им вести проект к успеху.

Эта глава о том, как тестировщики продолжают вносить вклад в совершенствование проекта, по мере того как проект развивается и изменяется[1]. В большинстве проектов мы не собираемся поставлять ту систему, которая была показана тестировщикам на первом тестовом цикле. Мы надеемся, что в результате выполнения тестов качество будет улучшаться. Давление сроков и бюджета может повлиять на изменение запланированного множества свойств системы или запланированного объема тестирования. Может оказаться, что некоторые свойства реализовать сложнее, чем ожидалось вначале, в результате эти свойства будут удалены из системы. Естественно, мы собираемся многое узнать о системе во время выполнения тес-

---

[1] Фрагменты этой главы были ранее опубликованы в виде статьи "Test Release Processes" в журнале "*Journal of Software Testing Professional*", Volume 1, Issue 2. В статье использовалось большое количество цитат моих коллег-тестировщиков. Хотя я удалил цитаты для того, чтобы не нарушать общий стиль книги, идеи, изложенные в них, остались. Я бы хотел поблагодарить за неоценимый вклад: Рэндала Беккера, Росса Колларда, Гэри Даусона, Харви Дойча, Тима Дайеса, Дэнни Фота, Пеги Фоутс, Дарси Мартина и Барбару Руф.

тов, получить информацию, которая поможет понять, что нужно делать дальше. И, конечно, слишком оптимистичные сроки и бюджеты, нереалистичные ожидания, связанные с тестированием, ошибочная вера в непроверенные технологии часто рушатся в ходе выполнения тестов, что приводит к довольно драматичному пересмотру оценок проекта.

Эволюция и изменения происходили практически во всех проектах, в которых я принимал участие. С учетом этого обстоятельства проекты тестирования должны развиваться и изменяться в качестве ответных мер на получение информации об изменениях проекта. Изменения могут охватывать все стороны проекта тестирования, включая оценку рисков, бюджеты, графики, планы, персонал, систему тестов, тестовые версии, выполнение тестов, отчеты об ошибках и результатах тестирования. Короче говоря, тестировщики должны быть готовы к тому, чтобы перестроить весь процесс тестирования и каждый входящий в него ключевой процесс в ответ на изменения в проекте. Тестировщики должны соответствовать задачам проекта в целом за счет готовности, желания и способности вносить изменения[1].

Процесс управления изменениями

Процесс 12 – это простой обобщенный процесс управления изменениями. Возможно, вы заметили, что я рассматриваю процесс разбора ошибок в качестве элемента процесса управления изменениями. Иногда они разделены, так что позвольте пояснить, почему я их объединяю.

В большинстве случаев изменение есть изменение. Источник запроса на изменение (отчет об ошибке, подготовленный группой тестирования, или запрос на усовершенствование от группы маркетинга) не влияет на то, нуждается ли система в изменении в ответ на запрос. Источник запроса на изменение может повлиять на приоритет запроса на изменение. При разработке заказных систем запрос на изменение может инициировать увеличение стоимости системы для клиента, тогда как исправление ошибки может на это не повлиять. Однако для продуктов для массового рынка мы либо удовлетворяем потребности пользователей, либо нет. На рынке коммерческих продуктов есть такие ситуации, когда имеет смысл отложить до следующей версии исправление ошибок, которые со всей определенностью относятся к нарушениям установленных и утвержденных требований. Возможны также случаи, когда имеет смысл не откладывать запрос на усовершенствование, который явно находится за рамками тех же самых требований[2].

---

[1] Эта глава в первую очередь ориентирована на процесс тестирования проекта разработки, сопровождения или интеграции, для которого определение успешной передачи системы может со временем меняться. Если тестирование проводится для принятия бинарного решения – принять или отвергнуть систему, изменения в этом случае менее вероятны, возможно, лишь в контексте измененного контракта. Аналогично, если тестирование является частью оценки конкурирующих систем или компонентов при принятии решения о выборе одного из них для поставки в компанию, в этом случае критерии выбора обычно определяются заранее.

[2] Сем Канер, Джеймс Бах и Брет Петтикорд отмечают в "*Lessons Learned in Software Testing*": "Рассматривайте проект как непрерывное структурированное обсуждение того, что разумно делать дальше" и "Всегда есть изменения на поздних этапах проекта".

Независимо от того, объединяете вы разбор ошибок и управление изменениями в один процесс или используете два отдельных процесса, риск изменений обычно возрастает по мере того, как развивается проект. Чем меньше остается времени, тем меньше пространства для маневра. Поэтому чем ближе мы находимся к плановому окончанию проекта, тем меньше изменений мы можем реализовать, не подвергая риску существующие свойства, бюджет, сроки и, возможно, качество. Следовательно, в конце проекта этот процесс становится более критическим.

Процесс 12. Процесс управления изменениями

№ этапа	Этап	Выполнено?
1.	Соберите запрашиваемые изменения и исправления ошибок, предлагаемые для включения в текущую, будущую или в аварийную версию системы.	☐
2.	Обсудите предлагаемые запросы в ходе регулярного или экстренного совещания группы управления изменениями, при помощи электронной почты или на телеконференции.	☐
2.A	Оцените связанные с запросами получаемые преимущества в свойствах, бюджете, сроках и качестве, затраты, проблемы и риски реализации, тестирования и выпуска версии. При необходимости отложите рассмотрение до очередного совещания и получите дополнительную проясняющую информацию.	☐
2.B	Присвойте приоритеты каждому отчету об ошибке или отвергните его.	☐
2.C	Идентифицируйте передаваемые материалы реализации, тестирования и интеграции версии; оцените даты завершения для каждого запроса.	☐
3.	Спланируйте, реализуйте, протестируйте и интегрируйте изменение или исправление ошибки, отмечая новые затраты, преимущества, проблемы и риски.	☐
4.	Представьте результаты реализации, тестирования и интеграции версии и передаваемые материалы для окончательного утверждения.	☐
4.A	Оцените предстоящие затраты, преимущества, проблемы и риски, относящиеся к свойствам, бюджету, срокам и качеству.	☐
4.B	Сравните преимущества добавления изменения с затратами, проблемами и рисками.	☐
4.C	Примите окончательное решение: одобрить или отвергнуть изменение в соответствующей версии.	☐
5.	Если внесение изменения утверждено, включите новые или измененные компоненты системы, проектную документацию и другие передаваемые материалы в систему управления конфигурацией.	☐

И наоборот, чем раньше мы начинаем изменения, тем меньше риск каждого изменения. Иначе говоря, чем раньше мы начинаем изменения, тем в большей степени один или несколько элементов проекта (свойства,

бюджет, сроки и качество) могут быть предметом обсуждения для последующего изменения. С одной стороны, если требования не зафиксированы, является ли требование, которое возникло сегодня и о котором мы не слышали вчера, *изменением* в настоящем смысле этого слова? С другой стороны, если большинство свойств были реализованы с применением определенной архитектуры системы, предложение перейти к новому дизайну, вероятно, будет иметь далеко идущие последствия, по крайней мере, для графика и бюджета и, весьма вероятно, для свойств и качества.

Поэтому грамотный процесс управления изменениями зависит от времени проекта. Если процесс управления изменениями начинается в первый день, вероятно, это приведет к ненужным накладным расходам. Если процесс не начать прежде, чем проект выйдет из-под контроля, может оказаться слишком поздно. Группа управления проектом должна принять решение о том, когда начинать управление изменениями и насколько формально и осмотрительно делать это со временем. Ключевым для проекта является грамотный выбор уровня управления изменениями в нужное время.

В главном примере книги мы рассмотрим тот период времени в проекте, когда формальный и грамотный процесс управления изменениями особенно необходим. Как только изменение в том или ином элементе проекта с большой вероятностью начинает оказывать существенное воздействие на другие элементы, проектная группа в полном составе должна тщательно рассмотреть каждое предлагаемое изменение.

Джамал готовит и согласовывает план

Джамал Браун, тест-менеджер компании Software Cafeteria, большую часть рождественских каникул отсутствовал на работе. Однако он иногда заглядывал в сеть, чтобы прочитать отчеты об ошибках, уточняя предлагаемые приоритеты после совещания с авторами отчетов. Он также обновлял информацию об отслеживании тестовых сценариев в ходе ежедневных телеконференций с тестировщиками, которые работали во время каникул.

Кроме того, он потратил время на изучение самых последних запросов на изменения проекта "Суматра". Они хранились в той же базе данных, которая применялась для управления дефектами, но были помечены как запросы на изменения. Конечно, иногда запрос на изменение становился ошибкой и наоборот, но с точки зрения управления изменениями между ними не делалось больших различий. Все новые запросы на изменения и отчеты об ошибках, по крайней мере, касающиеся проекта "Суматра", выносились трижды в неделю на рассмотрение группы управления изменениями.

№ этапа	Этап	Выполнено?
1.	Соберите запрашиваемые изменения и исправления ошибок, предлагаемые для включения в текущую, будущую или в аварийную версию системы.	☑

Большинство изменений казались весьма незначительными, например новый экран помощи. Некоторые были существенными, в частности мастер создания счета в системе управления документооборотом из продукта SpeedyWriter. Да, думал Джамал, похоже, здесь потребуется объемное тестирование на совместимость, если учесть, что мы поддерживаем различные системы управления документооборотом.

Одно из предлагаемых изменений было одновременно весьма привлекательным, сложным в реализации и определенно очень рискованным. Мухаммед Аджами, работающий в группе продаж компаниям Fortune 1000 и государственным учреждениям, предложил, чтобы Software Cafeteria начала работать в качестве поставщика прикладных услуг (ASP). Он сказал, что некоторые из его клиентов, сытые по горло высокой стоимостью владения и низкой надежностью систем на базе персональных компьютеров, ищут выход в виде тонких клиентов, например браузеров и информационных приложений, чтобы снизить затраты и увеличить надежность. Эти заказчики постоянно спрашивают, нет ли возможности передать субподрядчикам непрерывную поддержку требований к серверной стороне. Вместо того чтобы поддерживать работу большого числа серверов, которые необходимы для нормальной работы пакета OfficeArrow в их среде, он хотят приобретать такую услугу у поставщика прикладных услуг. Мухаммед предложил, чтобы Software Cafeteria воспользовалась такой возможностью, начиная с "Суматры".

Это был бы отличный и к тому же уникальный ход. Джамалу не было известно ни об одном провайдере прикладных услуг, который предлагал бы в качестве приложений офисные пакеты. Возможно, что об этом пока никто не задумывался, либо это была совершенно ужасная идея. Либо эта идея не так ужасна, однако ужасно трудно ее осуществить.

Джамал решил провести к началу совещания группы управления изменениями неформальный анализ рисков. В ходе анализа рисков в первую очередь он обращал внимание на риски качества, но не забывал и про остальные риски. Когда Джамал завершил анализ, предложение Мухаммеда насторожило его еще больше, поэтому он решил разослать документ, получившийся в результате анализа рисков, до совещания группы управления изменениями и предоставить его также на совещании. Поскольку могло показаться, что он сурово критикует Мухаммеда, Джамал решил сначала отправить письмо ему.

Привет, Мухаммед!

Надеюсь, ты отлично провел праздничную неделю. Я немного поработал на этой неделе, ничего серьезного. Я также просмотрел новые запросы на изменения.

Должен сказать, что я весьма впечатлен и заинтригован твоим предложением по поводу нашего участия в предоставлении прикладных услуг, начиная с Суматры. Однако я провел небольшой анализ рисков и выявил несколько проблем, связанных с тестированием. Я собираюсь обнародовать эти соображения в понедельник в ходе совещания группы управления изменениями, но мне хотелось бы показать их сначала тебе, чтобы ты не чувствовал, что на тебя нападают исподтишка. Пожалуйста, посмотри прикрепленный документ. Я разошлю этот документ по электронной почте в понедельник утром перед совещанием. Если ты обнаружишь неточности или захочешь обсу-

дить документ до его рассылки, пожалуйста, не стесняйся: позвони мне домой (830) 438-4830 или пришли электронное сообщение.

С уважением,

Джамал

В понедельник утром Мухаммед не ответил, поэтому Джамал разослал документ членам группы управления изменениями. В состав группы входили вице-президент по поддержке разработок Джон Албертсон, вице-президент по системной разработке Гарольд Джоунз, менеджер разработки новой версии Дженни Кауфман, менеджер по маркетингу продукта SpeedyWriter Кейт Эрнандес, менеджер системных операций и администрирования Кейт Ли, менеджер поддержки пользователей Косимо Неграев, менеджер подготовки сборок Ясухиро Канагава, а также заинтересованные представители группы продаж. Джамал также включил в список рассылки генерального директора Software Cafeteria Раджеша Гупту, который собирался быть на совещании, чтобы выступить по вопросу о слишком большом количестве отложенных ошибок.

Совещание началось в 14.00. Кейт Эрнандес, как спонсор проекта "Суматра", председательствовала на совещании. В повестку дня входили обсуждение новых отчетов об ошибках, новых запросов на изменение и, наконец, любых обновлений, связанных с реализацией исправлений и запросов на изменения.

За последнюю неделю накопилось довольно приличное количество необработанных отчетов об ошибках. Поскольку был только первый понедельник, когда сотрудники вернулись после праздников, никто не просмотрел все отчеты. Постоянно задавались вопросы: "О чем этот отчет? Я еще не успел его прочесть", и тратилось довольно много времени на обсуждение, стоит исправлять ошибки или отложить их. Увидев, что совещание начинает выходить из-под контроля, Кейт вмешалась.

"Послушайте, — сказала она. — Я понимаю, что все заняты, но времени не прибавится, если мы будем по два-три раза возвращаться к обсуждению одних и тех же ошибок и запросов на изменения. Позвольте мне напомнить вам три фундаментальных правила группы управления изменениями. Первое — все приходят на совещание, подготовившись. Совещания слишком затягиваются, если для каждой ошибки и каждого запроса на изменение требуется проводить специальную пресс-конференцию. Если совещания затягиваются, люди прекращают их посещать, а затем мы поставляем продукт с ошибками. Помните SpeedyWriter 2.7? Второе — мы тратим на обсуждение одной ошибки или одного запроса на изменение не более пяти минут. Я пытаюсь добиться, чтобы наши совещания длились не больше часа (сегодня мы уже перекрываем этот отрезок времени, поскольку еще есть ошибки и запросы на изменения), но это возможно только в том случае, если мы не будем отвлекаться на другие темы. Если кто-то хочет говорить больше пяти минут об ошибке или изменении, чтобы принять решение, мы переносим рассмотрение этой ошибки или изменения на отдельное совещание. Наконец, помните, что мы принимаем решение о том, реализовать или отвергнуть изменение, только один, а не два и не три раза. Кроме того, мы стремимся принять решение на совещании, ближайшем к моменту помещения ошибки или запроса на изменение в систе-

му управления дефектами. У нас нет ни времени в этом проекте, ни сил на остановку анализа, долгие задержки в рассмотрении запросов и гамлетовские дебаты. Время идет, пока мы здесь сидим, что лишает нас пространства для маневра, поэтому давайте принимать решения, пока у нас еще есть выбор".

Раджеш, сидевший сзади и предоставивший Кейт функции по укрощению строптивой аудитории, спросил с улыбкой на лице: "Гамлетовские?

"Чинить иль не чинить – вот в чем вопрос, достойно ль смиряться под ударами рассерженных заказчиков иль ... " – пояснила Кейт.

"Понятно", – сказал Раджеш.

"Итак, – резюмировала Кейт, – в следующий раз совещание состоится уже в Новом году, верно? Я бы хотела, чтобы в Новом году мы вернулись к четким и компактным совещаниям группы управления изменениями, которые у нас были до каникул. Что касается оставшейся части сегодняшней встречи, я попрошу поднять руки тех, кто прочитал отчет об ошибке или запрос на изменение до начала его обсуждения. Если голосование не будет единогласным, мы отложим обсуждение этого вопроса до пятницы".

Поворчав, группа управления изменениями выбрала более эффективный способ работы. Наконец они добрались до запроса на изменение 72, посвященного предоставлению прикладных услуг. "Все ли прочитали этот запрос на изменение? – спросила Кейт.

Все подняли руки. Это не было случайным совпадением, поскольку Джамал отправил голосовое сообщение с подтверждением получения, попросив всех прочитать электронную почту. Он не получил подтверждения от Мухаммеда, поэтому позвонил ему. Мухаммед признал, что прочитал письмо Джамала. Хотя оно не слишком его обрадовало, он согласился, что замечания необходимо представить аудитории.

"Хорошо, – сказала Кейт. – Насколько я понимаю, этому вопросу уже уделил немного внимания наш бесстрашный тест-менеджер. Я поговорила с Дженни, и хотя ей не кажется, что реализация слишком сложна, она склонна согласиться с замечаниями Джамала по поводу рисков".

Кейт Ли добавил: "Ясухиро, Косимо и я обсудили за обедом анализ Джамала, мы тоже согласны с ним".

"Хорошо-хорошо, попридержи коней, – попросила Кейт. – Мухаммед, по повестке дня ты как инициатор имеешь право выступить первым и представить запрос. Обычно мы спрашиваем мнение всех по кругу, но я думаю, что Джамал будет основным оппонентом".

"Нет, – встревожился Джамал, – я не оппонент. Я просто обеспокоен..."

"Конечно, ты обеспокоен, ты же тест-менеджер. Не за это ли мы тебе платим?" – спросила язвительно Кейт.

"За это больше, чем обычно", – сказал Джамал с улыбкой.

"Я думал, что буду выступать первым", – сказал Мухаммед, немного раздражаясь.

"Прости, ты прав, пожалуйста, бери слово".

"Хорошо. Итак, я обсуждал Суматру с несколькими клиентами из числа Fortune 1000 и правительственных организаций. Они действительно одобряют общую концепцию Суматры, но обеспокоены общей стоимостью владения. Одна из причин, по которой они предпочитают пакет OfficeArrow, состоит в том, что им не нужно инсталлировать приложение

на каждой рабочей станции, что было бы кошмаром для большой компании. И им нравится идея системы управления документооборотом. Это пока довольно большое препятствие, особенно для больших юридических фирм. Но им не нравится мысль о необходимости оплачивать поддержку и сопровождение сложной конфигурации серверов. Некоторым нашим клиентам довольно непросто поддерживать трехзвенную конфигурацию серверов. А теперь мы говорим о необходимости еще большего количества персонала для поддержки. Полагаю, что мы можем потерять некоторые потенциальные продажи.

Вот тогда и появилась у меня идея о том, не могли бы мы стать провайдерами прикладных приложений. Вместо того чтобы наши клиенты сами поддерживали и сопровождали свои серверы, мы создаем централизованную группу серверов и сопровождаем некоторое число клиентов. Вся техническая экспертиза будет располагаться внутри нашей компании, поэтому им не нужно будет нанимать специалистов. Это было бы уникальное предложение. Судя по разговорам потенциальных клиентов, по моим оценкам, мы могли бы легко получить не менее 1 миллиона долларов прибыли в течение первого года только с учетом тех компаний, которые уже заинтересованы в таком виде услуг. Если это технически не очень сложно реализовать (и, судя по реакции Дженни, это действительно несложно), я бы сказал, что мы должны попытаться это сделать".

Кейт кивнула, а затем обратилась к Джамалу: "Джамал?"

"Отлично, — сказал Джамал, — начну с того, что мне действительно нравится идея Мухаммеда. У меня нет никакого желания быть "Мистером Нет" на этих совещаниях, и я полагаю, что меня нельзя обвинить в неправильном подходе к обсуждению запросов на изменения. Однако я считаю, что время этой отличной идеи еще не наступило.

Сегодня утром я отправил этот документ каждому из вас, — сказал Джамал, начиная раздавать копии, — думаю, что вы знакомы с содержанием. Это анализ рисков и последствий, который я провел после того, как прочитал и обдумал предложение Мухаммеда. Я классифицировал различные проблемы по категориям рисков качества — тем же категориям, которые мы использовали при проведении первоначального анализа рисков качества с применением инструмента анализа видов ошибок и их влияния, а затем я проранжировал все риски на основе интуиции и других точных научных приборов, — он улыбнулся. — Я использовал ранги "очень высокий", "высокий", "средний", "низкий" и "очень низкий". Не собираюсь занимать ваше время изучением "средних" и более низких рангов. Вы можете прочесть документ, если хотите узнать подробности. Но позвольте мне кратко обрисовать пять основных проблем.

Во-первых, в области эксплуатации и сопровождения я оцениваю риски проблем, которые могут угрожать непрерывной работе, как очень высокие. Мы должны будем создать, протестировать и поддерживать замысловатые сетевые операции и сложный центр данных. Я ранее работал у провайдера Интернет-служб и могу сказать, что технически это далеко не простая задача". Кейт Ли кивнул, а Джамал продолжал: "С точки зрения следствий для тестирования, мы должны создать тестовую среду, которая повторяет среду эксплуатации или, по крайней мере, очень близка к ней, что будет, без сомнения, дорогим и не простым с организационной точки

зрения для группы тестирования и для эксплуатации. С точки зрения следствий для бизнеса, у нас будут высокие первоначальные затраты и затраты на непрерывные операции, включая аппаратное и программное обеспечение, сетевое оборудование и персонал. Мы можем легко тратить миллион долларов в год только на поддержку центра данных и системы в рабочем состоянии.

Во-вторых, что касается совместимости, отказов в работе оборудования, операционной системы, браузера и сервера, здесь риски и влияние очень высоки. Для Fortune 1000 и правительственных организаций придется хранить огромные объемы данных. Стоимость Network Attached Storage – от $25 тыс. до $250 тыс. на конфигурацию, а Storage Appliance Networks – от $1 млн. Так что мы получаем очень дорогую тестовую среду и очень сложную операционную среду.

В-третьих, я думаю, что мы столкнемся с очень высокими рисками, связанными с безопасностью и секретностью. Клиенты будут справедливо рассчитывать, что их документы никогда не увидят другие клиенты. Для тестирования это означает существенное увеличение тестовых сценариев безопасности. С точки зрения бизнеса, в случае каких-либо инцидентов в этой области мы можем столкнуться с серьезными исками.

В-четвертых, неизвестна степень рисков с точки зрения конкуренции. Могу только предположить, что клиенты будут сравнивать нас с другими провайдерами в других сферах предоставления услуг, но последствия этого сейчас неочевидны. В-пятых, я думаю, что мы столкнемся с высокими рисками и серьезными последствиями, связанными с областью качества данных – проблемами с обработкой, хранением и извлечением данных. Наши клиенты будут справедливо полагать, что мы в состоянии безопасно хранить их данные, включая восстановление данных после сбоев и восстановление файлов при их случайном удалении. С точки зрения тестирования, это не так серьезно. Мы просто расширим существующие тестовые сценарии, чтобы проверить различные ситуации, связанные с предоставлением услуги доступа к прикладным программам. Однако для бизнеса последствия являются пугающими, особенно с точки зрения возможных исков, если данные клиента бесследно исчезнут.

Итак, я уже сказал, что выбрал пять основных проблем. Я также хочу упомянуть производительность, доступность и стабильность. Наши клиенты, очевидно, будут ожидать очень качественных решений по этим позициям, не так ли? А тестирование производительности среды большого центра данных потребует не только соответствующей по размерам и возможностям тестовой среды, но и гораздо большего числа лицензий на инструменты, чем я имею сейчас. Кроме того, я не уверен, что смогу провести к марту месяцу тестирование наработки на отказ".

"Тестирование наработки на отказ?" – переспросил Мухаммед.

"Извини, это точное статистическое измерение того, как часто система будет терпеть крах. Похоже на рекламную акцию про 99.999%, которая прошла недавно".

Некоторое время стояла тишина. Все смотрели по сторонам.

"Отменная презентация, – признал Раджеш. – Вероятно, ты имеешь в виду, что нам придется построить как минимум два центра данных, один – для работы, второй – для тестирования".

"Да, это так", — ответил Джамал.

"Почему?" — спросил Раджеш.

"Потому что довольно скоро нам потребуется обновить программы в рабочем центре данных либо новыми версиями пакета OfficeArrow, либо другими компонентами. Нам нужно будет тщательно тестировать новые конфигурации перед тем, как мы передадим их в эксплуатацию, поскольку доступность системы — это очень важный вопрос".

"Согласен, но, — продолжал настаивать на своем Раджеш, — почему нельзя использовать уменьшенную модель рабочей конфигурации".

"Она подойдет для функционального тестирования, — согласился Джамал. — Однако ошибки надежности, производительности, качества данных и совместимости обычно не проявляют себя в уменьшенных моделях так же, как в настоящих конфигурациях. Вот пример. Пусть я выполняю тестирование производительности в уменьшенной конфигурации сервера и обнаруживаю, что производительность падает до неприемлемой величины при обслуживании 100 пользователей. В эксплуатационной среде имеются процесс, работающий впятеро быстрее, диск, емкость которого в пять раз больше, и память, также больше в пять раз, чем в тестовой среде. При каком количестве пользователей производительность в рабочей среде снизится до неприемлемых величин? При 500? При 2500? Или при 12 500?"

Раджеш рассмеялся и ответил: "Я вряд ли смогу ответить на этот вопрос. Но я знаю, в компании есть умные технические специалисты, которые знают ответ, верно?" Он посмотрел на присутствующих.

"Да, — с неохотой вставила Дженни, — мы можем построить динамические статистические модели для производительности системы. Мы уже делали это для предыдущих версий. При проверке модели в идентичных конфигурациях среды мы могли получить результаты, довольно близкие к реальным. Эти модели постепенно приближались к идеалу, однако мы фактически никогда ими не пользовались. Проблема с моделированием такого сорта состоит в том, что нужно знать, где находятся узкие места, нужно уметь предвидеть экстремумы кривой производительности и выбирать подходящие статистические распределения (формулы плотности вероятности) для каждого ресурса... — голос Дженни затих, она погрузилась на минуту в проблемы. — В любом случае есть много деталей, и если не уследить за какой-либо из них, забыть хотя бы одну или неверно использовать в модели, вы получите не результаты тестирования, а мусор. Единственный способ проверить модель — это тестирование в настоящей среде".

"Но вы могли бы, — настаивал Раджеш, — будучи очень-очень аккуратными и очень-очень проницательными, построить модель, которая проверяет усеченную версию и позволяет прогнозировать производительность и тому подобное для центра данных, верно?"

Джамал сказал: "Да, Раджеш, это, конечно, возможно. Аэрокосмические и оборонные компании делают это постоянно. Они серьезно заняты моделированием, тестированием усеченных и макетных конфигураций, это верно, когда мы говорим о системе вооружения или летательном аппарате. Но представьте тестирование системы противоракетной обороны, о которой мы читаем время от времени или видим в новостях. Каж-

дый из тестов стоит очень и очень дорого, примерно 100 миллионов долларов. Я не сомневаюсь, что перед началом выполнения каждого из тестов проводится значительная работа по тщательному моделированию. Но все равно бывают неудачи. Когда мы говорим о сложной системе в динамической и непредсказуемой среде, там возможны непредвиденные последствия.

Дженни кивнула: "Джамал прав. Нет другого способа, кроме подробного тестирования, чтобы понять наверняка, что будет происходить в сложном центре данных".

"Хорошо, если вы не можете получить верные результаты при моделировании, что заставляет вас думать, что тестирование дает правильные результаты? Не являются ли ваши тесты просто моделью того, как пользователи применяют систему?" – возразил Раджеш.

"Раджеш, ты абсолютно прав, но выбор у нас не либо один, либо другой, а либо оба, либо только один. Мы могли бы использовать имитационную модель, а также статические модели, например таблицы, в обоих случаях, – ответил Джамал. – Однако без тестирования мы не можем проверить имитационную модель. При наличии требования постоянного доступа мы принимаем на себя риск "выставления компании на пари", что модели построены правильно, не проверив это заранее. У нас грамотные сотрудники, Раджеш, – сделал вывод Джамал, – но не настолько. Уверен, рано или поздно у нас будут большие проблемы. Это рулетка качества". Раджеш осмотрел сидевших в помещении: "Дженни, я думаю, у тебя такое же мнение? Кейт Эрнандес? Кейт Ли? Ясухиро? Что думают остальные?"

Все согласно кивали.

"Хорошо, – сказал Раджеш, откидываясь назад и скрещивая руки за головой. – Причина, по которой я так настойчиво продвигал это предложение, состоит в том, что я размышлял над этой темой вместе с Мухаммедом, и я поговорил с некоторыми клиентами. Это очень актуальная тема. Я также обсуждал этот вопрос на совете директоров и с некоторыми потенциальными инвесторами. Они действительно горят желанием получить что-то подобное, хотя инвесторы, естественно, не намерены тратить кучу денег прямо сейчас. Они рассматривают переход Software Cafeteria от продажи пакета к предоставлению услуг офисных приложений как очередной логический шаг в развитии компании". Раджеш посмотрел вокруг, встретившись с каждым из сидящих взглядом, и добавил: "И я тоже".

Джамал откинулся в кресле. "Однако, – думал он, – я высказал столько замечаний по поводу любимого проекта босса. Я думаю, что сейчас понимаю, что он действительно имел в виду, когда говорил о "желании услышать наши откровенные мнения" на совещании, посвященном запуску проекта".

"Но я знаю, у нас в компании много талантливых сотрудников, отлично подготовленных технически, и эти талантливые сотрудники выражают беспокойство, включая тебя, Джамал, – сказал Раджеш с поддержкой и уважением. – Я полагаюсь именно на такой честный и тщательный анализ, который ты подготовил по этому вопросу. И я думаю, что твоя работа фактически устанавливает стандарт для всех в группе управления. Я не за-

бываю ничьих соображений, и помню о сложности работы. Однако я обрисовал стратегическое направление для компании. Возможно, это неподходящее место, и я, вероятно, навлек на себя гнев Кейт по поводу повестки дня и правил работы группы".

Кейт, внимание которой было поглощено изменениями в стратегии компании, ответила не сразу после того, как услышала свое имя: "Нет, вовсе нет, я так увлеклась. Думаю, что мы можем выделить время на это обсуждение".

"Хорошо, я почти все сказал, — заявил Раджеш. — Есть предложение. Мухаммед и я определим два наиболее актуальных направления для провайдерских услуг. Через два месяца после завершения версии "Суматра" я бы хотел перейти к подготовке бета-версии SpeedyWriter ASP для небольшого, но функционального центра данных. Мы честно скажем нашим заказчикам, что мы изучаем, как лучше это сделать. Они могут реагировать как угодно, но мы не хотим терять данные. Гарольд и Джон, я хотел бы, чтобы вы подобрали в своих подразделениях квалифицированных людей, которые могли бы набросать план, как это сделать". Гарольд и Джон согласно кивнули. "Мы можем это сделать, не так ли?

"Если мы будем внимательны и осторожны, — сказала Дженни, — я думаю, мы сможем это сделать".

"Джамал?"

"Да, — ответил Джамал. — Для полной уверенности я должен поговорить со своими людьми, но как мне кажется, все это в пределах возможного".

"Тогда у нас есть план, — помпезно завершил обсуждение Раджеш. — Кейт, тебе слово".

№ этапа	Этап	Выполнено?
2.	Обсудите предлагаемые запросы в ходе регулярного или экстренного совещания группы управления изменениями, при помощи электронной почты или на телеконференции.	☑
2.A	Оцените связанные с запросами получаемые преимущества в свойствах, бюджете, сроках и качестве, затраты, проблемы и риски реализации, тестирования и выпуска версии. При необходимости отложите рассмотрение до очередного совещания и получите дополнительную проясняющую информацию.	☑
2.B	Присвойте приоритеты каждому отчету об ошибке или отвергните его.	☑
2.C	Идентифицируйте передаваемые материалы реализации, тестирования и интеграции версии; оцените даты завершения для каждого запроса.	☑

Совещание продолжилось. Участники перешли к завершению реализации, тестирования и интеграции изменений и исправлений дефектов, которые были ранее утверждены. Ссылаясь на одно из изменений, Джамал произнес с улыбкой: "Хотел бы всем напомнить после моей недавней обструкции о том, что я полностью поддержал это изменение и очень рад сообщить, что оно прошло функциональное тестирование и тестирование данных без проблем".

№ этапа	Этап	Выполнено?
3.	Спланируйте, реализуйте, протестируйте и интегрируйте изменение или исправление ошибки, отмечая новые затраты, преимущества, проблемы и риски.	☑
4.	Представьте результаты реализации, тестирования и интеграции версии и передаваемые материалы для окончательного утверждения.	☑
4.A	Оцените предстоящие затраты, преимущества, проблемы и риски, относящиеся к свойствам, бюджету, срокам и качеству.	☑
4.B	Сравните преимущества добавления изменения с затратами, проблемами и рисками.	☑
4.C	Примите окончательное решение: одобрить или отвергнуть изменение в соответствующей версии.	☑
5.	Если внесение изменения утверждено, включите новые или измененные компоненты системы, проектную документацию и другие передаваемые материалы в систему управления конфигурацией.	☑

Наконец, Раджеш и Косимо представили ошибки, которые, по их мнению, были отложены до следующей версии напрасно. Примерно десяток ошибок были открыты заново.

"Джамал, — сказал Раджеш, завершая обсуждение, — спасибо за то, что ты подал мне идею просмотреть эти ошибки. Десять — небольшое число, но некоторые из них действительно существенны".

"Рад помочь, Раджеш", — ответил Джамал.

"В целом, — подвел итог Раджеш, — с учетом примерно тысячи ошибок, обнаруженных к настоящему времени, я думаю, что мы отлично справляемся с динамикой этого проекта. Я вижу, что большим подспорьем являются и это совещание, и весь процесс управления изменениями. Я призываю всех вас поддерживать его".

Взаимосвязанный процесс

"Отлично, — скажете вы, — я понимаю, что процесс управления изменениями является ключевым, но является ли он *процессом тестирования?*" Это верное замечание. В отличие от одиннадцати процессов, описанных в этой книге, процесс управления изменениями относится в основном к ведению группы управления проектом. Другими словами, в процессе управления изменениями группа тестирования — это только один из участников. Группа тестирования не является ни основным, ни наиболее активным его участником.

Однако процесс управления изменениями оказывает сильное влияние на группу тестирования, а также на группы разработки и сборки версий. Процесс управления изменениями воздействует на все процессы тестирования. Процесс подготовки тестовых версий передает сборки с новым и исправленным содержимым. Процесс выполнения тестов проверяет новое и исправленное содержимое. Процесс подготовки отчетов об

ошибках выявляет конкретные проблемы в изменениях или исправлениях. Наконец, процесс подготовки отчетов о результатах тестирования передает информацию группе управления изменениями, чтобы ее участники могли принять решение, утвердить или отвергнуть изменение или исправление. Без эффективного управления изменениями все эти процессы работают менее эффективно и качественно.

В самом деле, в некоторых обстоятельствах процесс управления изменениями может отправить группу тестирования к начальным процессам, связанным с подготовкой тестирования. В нашем примере Джамал в ходе подготовки к совещанию проводит анализ рисков качества. В результате утверждения изменения в ближайшей версии Джамал вновь возвращается к процессам оценки затрат и планирования для того, чтобы рассмотреть введение услуг провайдера прикладных приложений для SpeedyWriter. Чтобы протестировать это предложение, группе тестирования, вероятно, понадобится спроектировать и разработать новые тесты. В случае если для выполнения этих тестов требуются специальные навыки или дополнительные тестировщики, у Джамала может возникнуть потребность принять на работу новых постоянных сотрудников или контрактников. В реальности для разработки услуг провайдера прикладных приложений может понадобиться отдельный процесс тестирования. Другими словами, версия провайдера прикладных приложений для Суматры будет отдельным проектом, а тестирование этой версии — подпроектом в рамках этого проекта.

Поэтому участие группы тестирования в процессе управления изменениями очень заметно, особенно когда тест-менеджер, такой как Джамал, привносит в него заслуживающую доверия конкретную информацию, чтобы прояснить тему рисков качества, связанных с изменениями. Процесс оказывает существенное влияние на способность группы тестирования приносить пользу проекту, и это влияние тем больше, чем больше число изменений, нуждающихся в управлении. Наконец, последствия неудачи этого процесса оказывают сильное воздействие на успех группы тестирования. Если нам неизвестно, в каком состоянии находится проект на заключительной стадии, то, очевидно, мы сделали что-то неверно в ходе тестирования. И наоборот, когда тест-менеджер, например Джамал, берет все под контроль, участие группы тестирования в этом процессе способствует успеху проекта[1].

Построение успешного процесса управления изменениями

Процесс управления изменениями довольно сложен. Однако грамотный процесс управления изменениями позволяет группе проекта сделать правильный выбор.

---

[1] Необходимо честно признать, что данная глава представляет взгляд на управление изменениями с позиции тестирования. Более широкий взгляд с позиции управления проектом должны принимать во внимание образованные тестировщики-профессионалы, поскольку он является частью понимания контекста проекта. Неплохой источник для изучения такого взгляда на процесс — книга Стива Макконела "*Software Project Survival Guide*", особенно глава 6.

Выбор нужных изменений в нужном порядке

Время идет и в жизни, и в проекте. Каждый день работы в проекте мы видим безжалостное движение времени к запланированной дате завершения проекта. Каждый день работы в проекте мы неизбежно истощаем его бюджет. Располагая меньшим количеством времени и денег, мы сталкиваемся с возрастающими рисками неудачи проекта, если рабочая группа проекта занимается не тем, чем нужно. Поэтому наиболее важный вопрос состоит в следующем: делаем ли мы сейчас то, что нужно, чтобы вести проект по направлению к наиболее успешному завершению? С точки зрения управления изменениями, вносим ли мы нужные изменения, корректируем ли курс в верном направлении, тратим ли наиболее ценные и невозобновляемые ресурсы проекта в нужном порядке?

Во многих проектах, в которых я работал, в периоды выполнения тестов существовал непрерывный цикл. Мы вносили изменения в определение проекта в ответ на новую информацию о свойствах, бюджете, сроках и качестве. Мы тестировали внесенные изменения, в результате получали еще информацию, в частности о качестве. Этот цикл продолжается до тех пор, пока группа управления проектом не примет решение о том, что критерии завершения фазы тестирования выполнены. На заключительной фазе тестирования, системного тестирования или приемо-сдаточных испытаний это решение также включает в себя решение о выпуске или поставке системы. Это тонкое и взвешенное решение. Превосходят ли риски, связанные с отсрочкой выпуска или поставки системы для внесения очередного исправления или усовершенствования и проведения тестового цикла, риски, связанные с поставкой системы с известными ошибками, без отложенных усовершенствований или незаконченных свойств?[1]

Поскольку недостаток времени отнимает у нас возможности для маневра, большое значение приобретает способность принятия решений. Читая эту главу, мой коллега тест-менеджер Харви Дойч сказал об этом так: "Не разрабатывай то, что ты не собираешься тестировать. Не тестируй то, что ты не собираешься выпускать. [Урезай] как можно раньше и чаще". Другими словами, в контексте полного тестирования, является ли то, над чем работает группа тестирования *прямо сейчас* наиболее важной совокупностью задач для достижения полного успеха проекта? Группа управления изменениями, составленная из специалистов разных направлений и использующая в качестве основы для своей работы результаты тестирования, может дать разумные рекомендации по рассматриваемому вопросу руководству компании и старшим менеджерам.

---

[1] В книге "*I Am a Bug!*" Роберт Сабурин пишет, касаясь вопроса оставшихся ошибок, рассматриваемых группой управления изменениями: "Мы считаем, что завершили работу, если оставшиеся ошибки – это ошибки, с которыми мы можем сосуществовать, по крайней мере, сейчас!

Уравновешивание мнений по поводу свойств, сроков, бюджета и качества

Группа, состоящая из специалистов разного профиля, идеальна для решения этого вопроса, поскольку сам вопрос, безусловно, охватывает различные сферы ответственности и должен рассматриваться с учетом многочисленных факторов. В этот перечень входят:

- Экономика, зависимости и задачи, связанные с реализацией, тестированием и интеграцией изменения или исправления ошибки в версию.
- Возможности и риски, затраты и преимущества предлагаемого изменения или исправления ошибки.
- Возможности и ограничения группы управления изменениями как с точки зрения трудозатрат, так и квалификации.
- Последнее, но не менее важное. Ключевые соображения бизнеса и технические проблемы, включая архитектуру системы, будущие требования и долгосрочное направление развития системы, группы проекта и компании в целом.

Слишком часто на совещаниях групп управления изменениями в ходе обсуждения исправления ошибок доминирует один вопрос. Страстные аргументы высказываются по поводу того, верно ли определена серьезность той или иной ошибки, в частности, серьезность 1 или серьезность 2. Обычный подтекст этой дискуссии состоит в том, что рассматриваются только ошибки серьезности 1. Но в сравнении с другими факторами, особенно стратегическими проблемами бизнеса, эти дискуссии не по существу. То же самое относится и к обсуждению того, стоит ли реализовывать предлагаемое изменение.

Принимая решение о том, исправлять ошибку или нет, реализовывать изменение или нет, проектные группы должны принимать к рассмотрению, как соответствующие состояния ошибок или отсутствие изменений повлияет на заказчиков и пользователей, поскольку они являются единственными арбитрами качества. И наоборот, группа тестирования должна учитывать, каким образом проявятся риски, связанные с внесением изменения или исправлением ошибки, но с точки зрения общего контекста проекта. В рассмотренной нами ситуации представление рисков Джамалом — это один из важнейших технических и экономических вопросов, относящихся к риску реализации изменения. Стратегическое направление развития компании, самые серьезные соображения бизнеса сделали это решение верным для бизнеса. Однако план Раджеша по продвижению в этом стратегическом направлении был немного приостановлен, а риск, присущий этому плану, смягчен, благодаря рассмотрению соображений Джамала.

В результате мы подошли к ключевому уроку, неявно вытекающему из всего примера: процесс должен предоставлять всем заинтересованным лицам и участникам возможность высказать свою точку зрения. В некоторых неудачных проектах процесс управления изменениями сопровождался вспышками раздражения. Одна-две фигуры руководителей эмоционально провозглашают, какое значение имеет предлагаемое изменение

или исправление ошибки, и игнорируют как обструкционистские так и, что еще хуже, реально существующие возражения и риски, представленные кем-либо еще.

Конечно, высшее руководство и старшие менеджеры имеют право, а в некоторых случаях просто обязаны отклонить решение группы управления изменениями. Старшие менеджеры и топ-менеджеры компании видят общую картину, что является их непосредственной обязанностью. Однако они обычно следуют грамотным советам, как дальше двигаться вперед с учетом смягчения, а не игнорирования реальных проблем и рисков. Поступать наоборот — это значит отказаться от вклада талантливых непосредственных исполнителей и линейных менеджеров проектной команды. Это равносильно вождению автомобиля ночью с выключенными фарами.

Важно, чтобы все участники, включая непосредственных исполнителей, имели более общее представление об организационном и стратегическом контексте, в который встроен проект. Если у группы управления изменениями отсутствует общая повестка дня, группа не представляет собой ничего иного, кроме политической организации, предназначенной для разрешения конфликтов между различными спорящими сторонами. Люди с большей вероятностью придут к схожему представлению об общем уровне риска, об оценке значения каждого риска и о преимуществах проекта в целом, если они имеют общий взгляд о направлениях развития проекта и компании. Другими словами, грамотный процесс управления изменениями устроен так, что его участники вырабатывают и обосновывают свои советы и решения в пределах общего видения будущего.

Решение проблем

В процессе управления изменениями существует ряд проблем, присущих проекту тестирования в целом. Рассмотрим те из них, которые относятся непосредственно к управлению изменениями.

Сложная природа воздействия изменений

Поскольку не все изменения и исправления ошибок равноправны по уровню риска, сопровождающего каждый из четырех элементов проекта, не все изменения и исправления ошибок оказывают одинаковое воздействие на различные группы. Изменение, в результате которого исправлена единственная строка кода, может потребовать от прилежного программиста потратить часы и даже дни, чтобы удостовериться, что изменение не вредит ничему другому в системе. Изменение, которое требует серьезной работы программиста, может быть тривиально для тестирования и для поддержки рабочей среды, но может оказаться сложным и требующим серьезных затрат времени от группы сборки версий для того, чтобы интегрировать изменение в сборку, а также для технических писателей, чтобы документировать изменения. Возможны и другие варианты. Даже при грубой оценке затрат на изменение, скажем, при разбиении затрат на небольшие, средние и крупные для каждой из пяти групп: разра-

ботки, тестирования, операций, сборки версий и технической документации, — существует 243 различных комбинации возможного распределения затрат.

Так, в проекте по разработке Интернет-приложения несколько раз упоминалась разработка комплекта тестовых сценариев для ручного тестирования, которые, среди прочего, тестировали последовательность действий с экранами. Экраны были реализованы на HTML для просмотра в браузере, что являлось развитой архитектурой для клиентских машин. Тестирование с помощью эксплуатационных тестов удобства использования показало, что пользовательский интерфейс неудобен и неэстетичен. Поэтому разработчики пользовательского интерфейса пришли в один из выходных и полностью переработали его. В понедельник утром мы получили новую тестовую версию, в которой был абсолютно другой пользовательский интерфейс. Когда я объяснил разработчикам интерфейса, что их изменения, несмотря на хорошее качество, могут вынудить нас переработать многие из тестов, они не поверили. "Почему? — спросил один из инженеров. — Это же просто ссылки!"

Как правило, группы управления изменениями ошибочно предполагают, что изменение, не оказавшее сильного воздействия на одну из групп, по всей вероятности, не окажет сильного воздействия и на все остальные группы. Мой опыт показывает, что за точку отсчета обычно принимают влияние изменения на группу разработки. Если изменение требует серьезных усилий от программистов, руководство склонно поддержать соответственно большие затраты на тестирование. Если затраты на разработку малы, руководство скептически относится к идее значительного увеличения сроков или бюджета на тестирование изменения.

Однако не относится ли это допущение линейной зависимости между затратами, необходимыми программистам, и затратами остальных групп, работающих над проектом, к старым заблуждениям, связанным с соотношением тестировщиков и разработчиков, упомянутым в главе 3? Если мы предположим, что затраты y человеко-часов на тестирование соответствуют затратам x человеко-часов на разработку, то получим следующую формулу:

$$y = Rx,$$

где R — постоянная, например равная

$$\frac{2\ РВжРирЛвшиНа}{5\ раБраЖЛР\ иНЬЛв}$$

В результате мы снова вернемся к использованию отношения между числом тестировщиков и разработчиков в качестве метода оценки затрат. Как уже отмечалось в главе 3, эти отношения могут быть полезны для дополнительной проверки декомпозиции работ, но не в качестве основного метода.

На самом деле, в этой ситуации такие соотношения еще менее полезны. Почему?

В большом проекте закон больших чисел сглаживает все крайности. Когда мы говорим о небольшом изменении или об исправлении ошибки, разность в измерениях затрат тестирования и разработки часто весьма значительная.

Эта проблема еще раз указывает на необходимость включения в процесс управления изменениями специалистов разных направлений, а также их участия на этапах процесса, цель которых дать реалистичную оценку воздействия изменения на каждую группу, занятую в проекте. Грамотный процесс управления изменениями должен предусматривать тщательное рассмотрение влияния изменения на каждую группу до принятия решения.

Волнообразный эффект изменений

Помимо необходимости переработки тестовых сценариев (см. выше), другим существенным источником нелинейной зависимости между затратами на тестирование и разработку является волнообразное влияние изменений на систему. В некоторых случаях изменение затрагивает компонент системы, который является центральным для всех остальных элементов системы.

Под эту категорию обычно подпадают данные и метаданные в центральной базе данных. В предыдущей главе упоминался проект, в котором одна группа разработки утверждала, что созданный ею компонент системы обладает всеми запланированными свойствами. Однако от версии к версии они продолжали изменять схему мастер-таблицы, в которой хранилась информация о пользователях. Это изменение оказывало такое же влияние на тестирование, как добавление или удаление свойств, поскольку каждое поле было привязано к изменениям пользовательского интерфейса и к модификациям в последовательности действий и транзакций, которые, в свою очередь, оказывали воздействие на агентов центра обработки вызовов. Кроме того, поскольку четыре из семи других подсистем использовали данные из этой базы, изменение воздействовало и на них.

Волнообразный эффект представляет собой отдельную проблему, если система, предназначенная для тестирования, является *системой систем*. Изменения и другие нелинейные воздействия могут волнообразно проходить через организационные границы. Другими словами, одна группа программистов может посчитать, что сделать изменение в их подсистеме несложно. Но другие группы программистов, группа тестирования и другие участники проекта обнаружат, что предлагаемое изменение перевернет все с ног на голову.

Вновь отмечу, что мы обязаны привлекать специалистов разных направлений. Группа управления изменениями может выйти за рамки проекта и вовлечь в свою работу представителей, не входящих в проектную группу. В случае волнообразного эффекта изменений и исправлений ошибок в стратегических, ответственных и взаимосвязанных системах систем может возникнуть необходимость привлечения к участию в группе управления изменениями топ- менеджеров компании.

Действительное или кажущееся препятствие?

"Ужасно, – возможно, вы думаете сейчас, – какие неправдоподобные накладные расходы. Не станет ли процесс управления изменениями огромным препятствием на пути прогресса? Как мы сможем исправлять ошибки и вносить изменения, если мы должны получить одобрение от двух- или трехуровневых групп управления изменениями, состоящих из десятков участников, каждый из которых может вполне быть заинтересован в сохранении статус-кво и минимизации объема работ, возложенного на уже перегруженные группы?"

Это верное соображение, и на него нет простого ответа. Управление изменениями служит не для создания препятствий изменениям, а для того, чтобы убедиться, что для каждого предлагаемого изменения проведен необходимый анализ перед тем, как запустить его в работу. Любое изменение связано с затратами и рисками, а также с новыми возможностями и прибылью. Поэтому все зависит от расположенности компании к риску и ее стремления к экономии.

Компания, которая принципиально отказывается от рисков и стремится к экономии, вероятно, нуждается в более осторожном процессе управления изменениями. Компания, которая больше думает о возможностях, а не о рисках, и рассматривает затраты, связанные с изменениями, как потенциально большие прибыли, обуславливаемые завоеванием большей доли рынка, отнесутся к осторожному процессу управления изменениями как к препятствию. Таким образом, организационный контекст компании имеет значение при определении подходящего уровня формализации и тенденции к отсрочке или отклонению изменений.

В главном примере книги Джамал концентрируется на рисках и затратах. Раджеш Гупта прежде всего смотрит на возможности и прибыли. Однако Джамал привлек внимание Раджеша к политике группы управления изменениями, которая привела более чем к десятку неверно отложенных ошибок. В этом случае Раджеш принял аргументы Джамала, что затраты и риски будут слишком высокими. Участники группы управления изменениями будут до некоторой степени не согласны с политикой в отношении изменений. Время от времени группа управления изменениями отступает то в одну, то в другую сторону от принятых правил отношения к рискам и экономии компании. В самом деле, такие ситуации неизбежны в группе, состоящей из сотрудников разных направлений. Попытки устранить несогласие и отклонения от политики в отношении изменений обычно приводят к невозможности проведения анализа частью группы управления изменениями.

Однако последовательные ошибки или перекос в одном или другом направлении, например утверждение чересчур рискованных изменений или отклонение изменений, которые соответствуют корпоративным целям, указывают на проблемы в группе управления изменениями. Как упоминалось ранее, разделяемое всеми представление о будущем является ключевым индикатором грамотного процесса управления изменениями. Если участники группы управления изменениями не имеют общего представления или не разделяют представления о ситуации с руководством компании, группа управления изменениями становится препятствием.

Возможно, что группа управления изменениями, все руководители и все непосредственные исполнители разделяют общее видение ситуации, но группа управления изменениями все еще рассматривается как препятствие. Если процесс управления изменениями является слишком бюрократизированным и формальным для корпоративной культуры даже при условии, что расстановка приоритетов, критерии отклонения или принятия изменения четко соответствуют потребностям компании, сотрудники могут рассматривать группу как инструмент, предназначенный в основном для отклонения или откладывания изменений и исправлений ошибок. Вообще говоря, бюрократизм и формальные процессы имеют репутацию тирании, которая замедляет деятельность компании, независимо от того, на благо или нет. Мы вернемся к вопросу о формализации процессов в главе 17.

Внедрение усовершенствований

Как тестировщик и как тест-менеджер я часто ощущаю, что изменение выходит из-под контроля. Как я отмечал ранее, очень важно сохранить правильный баланс. Каким образом можно направить процесс управления изменениями в сторону более удачного баланса, сохранив при этом достаточную гибкость процесса?

1. Максимально объективно с токи зрения компании проанализируйте затраты и прибыли, риски и возможности, присущие имеющемуся процессу управления изменениями. С точки зрения тестирования естественно, что в первую очередь обращают внимание на риски[1]. На самом деле, с точки зрения непосредственных исполнителей, любой проект обычно можно сделать проще, допуская минимум изменений (естественно, исходящих извне проекта), пока проект находится в процессе разработки. Однако это редко является правильным направлением действий.
2. Если затраты и риски перевешивают прибыли и возможности, рассмотрите способы изменения процесса, чтобы снизить затраты и риски, как можно меньше наступая на прибыли и возможности. Как правило, наблюдается тенденция энергичной реакции на неуправляемые и недостаточно управляемые изменения в проекте, но учет динамики плюсов и минусов между теми, чьи интересы связаны с более слабым управлением изменениями, и теми, чьи интересы связаны с более плотным управлением изменениями, не имеет ничего общего с улучшением процесса.
3. Если существует позитивный и актуальный бизнес-план улучшения процесса, задайте вопросы себе как тестировщику или тест-менеджеру: "Являетесь ли вы подходящей фигурой для защиты изменения? Подходящее ли сейчас время для внесения изменения?" Если ответы на оба вопроса отрицательные, найдите человека, ко-

[1] Методы определения затрат и рисков, относящихся к изменениям в проекте, изложены в главе 6 книги "*Managing the Testing Process, Second Edition*".

торый с наибольшей вероятностью добьется успеха, и убедите его в этом. Если текущий момент не является подходящим, например, до завершения проекта осталось две недели и любое изменение должно быть просто заблокировано, лучше подождать, а не инициировать сеющие распри эмоциональные обсуждения того, что является совершенно посторонним для приоритетов бизнеса в настоящий момент.

4. Учитывайте политические соображения. Любые изменения процессов являются политическими. Есть сторонники изменений, и есть сторонники сохранения статус-кво. В случае с управлением изменениями различия могут быть полными. Чем менее функционален текущий процесс управления изменениями, тем более заметны различия между двумя противостоящими сторонами. Какими методами вы сумеете преодолеть возражения сторонников статус-кво?

Последнее замечание возвращает нас к более общей картине. Усовершенствование процесса должно соответствовать общему представлению о ситуации, что становится тем более важным, чем более взаимосвязанным и взаимопроникающим является сам процесс. В следующей главе мы вернемся к большой картине, чтобы посмотреть, как определить хороший процесс тестирования, как выстроить такой процесс и как разумным способом внедрить усовершенствования в процесс тестирования.

ГЛАВА 17

Снова к общей картине: совершенствование процесса тестирования

В первой главе мы рассмотрели процесс тестирования в целом, общую картину. В последующих главах мы исследовали все ключевые процессы, составляющие общий процесс. В заключительной главе мы снова возвращаемся к процессу тестирования в целом. Для напоминания позвольте мне воспроизвести еще раз Процесс 1 из главы 1.

Как выявить успешный процесс тестирования? Как справиться с трудностями? Как перейти от привычного ведения процесса к лучшему способу выполнения работ?

Определение грамотного процесса тестирования

На протяжении этой книги мы рассматривали, как работают группа тестирования и другие группы проекта, когда они следуют грамотно выстроенным процессам тестирования, а также чего они при этом добиваются и что приобретают. Что можно сказать о группах, следующих грамотно выстроенному процессу тестирования в целом?

Предоставление полезных и недорогих услуг

Многие заинтересованные стороны, включая старших менеджеров и руководство компании, безразлично относятся к процессу тестирования как к таковому. Их интересует лишь его влияние на затраты и его результат. С позиции стороннего наблюдателя, группа тестирования получает ресурсы, использует их для получения важной информации, а затем вовремя, четко и надежно передает полученные сведения с учетом разнообразных потребностей заинтересованных сторон. С точки зрения группы тестирования, мы предоставляем все эти полезные и недорогие услуги за

счет эффективного и производительного тестирования, удовлетворяющего требованиям сторон, заинтересованных в тестировании[1].

Процесс 1. Процесс тестирования

№ этапа	Этап	Выполнено?
1.	Планирование: понимание места тестирования.	☐
1.A	Определить функциональный (система, проект и процесс) и организационный контексты, в рамках которых выполняется тестирование.	☐
1.B	Определить и расставить приоритеты рисков качества системы, достичь консенсуса заинтересованных сторон проекта относительно возможностей тестирования по смягчению этих рисков.	☐
1.C	Оценить временные, ресурсные и финансовые затраты, необходимые для выполнения тестирования и согласованные с пунктом 1.B, получить поддержку оценки у руководства.	☐
1.D	Запланировать задачи, зависимости и участников, которые необходимы для смягчения рисков качества системы, и получить поддержку плана у заинтересованных сторон проекта.	☐
2.	Подготовка: сбор людей и тестов.	☐
2.A	Создать команду специалистов-тестировщиков, обладающих соответствующими умениями, навыками, отношением к делу и мотивацией, посредством поиска персонала и его обучения.	☐
2.B	Спроектировать, разработать, получить и верифицировать систему тестов, которую группа тестирования будет использовать для оценки качества системы, предназначенной для тестирования.	☐
3.	Проведение: выполнение тестов и сбор результатов.	☐
3.A	Получить и установить тестовую версию, состоящую из всех или нескольких компонентов системы, предназначенной для тестирования.	☐
3.B	Определить, отслеживать и управлять комплектом тестов, с помощью которого планируется проверять каждую из тестовых версий.	☐
4.	Совершенствование: предоставление информации о проекте и его улучшение.	☐
4.A	Документировать ошибки, обнаруженные в ходе выполнения тестов.	☐
4.B	Обсудить результаты тестирования с ключевыми заинтересованными лицами.	☐
4.C	Управлять изменениями и уточнить процесс тестирования.	☐

[1] Роберт Сабурин написал прекрасную статью "At Your Service", впервые опубликованную в журнале "*Software Testing and Quality Engineering*", Volume 3, Issue 3 (May/Jun 2001). В настоящее время ее можно найти на сайте www.stickyminds.com. В этой статье автор рассматривает тестирование и работу других групп разработки и проектирования программного обеспечения как организаций, предоставляющих услуги.

Эффективность означает получение желаемого и полезного результата. В этой книге я делаю упор на четыре способа, при помощи которых тестировщики достигают желаемых и полезных результатов.

1. Обнаружение ошибок, которые будут устранены, или даже предотвращение их внесения.
2. Обнаружение ошибок, которые не будут устранены, но о которых становится известно.
3. Проведение тестов, которые снижают (потенциально затратные) риски.
4. Обеспечение проекта своевременной, точной и заслуживающей доверия информацией.

Кроме того, эффективность означает, что заинтересованные в тестировании и в обеспечении качества стороны удовлетворены тем, каким образом мы предоставляем результаты. Это вытекает из определения качества, данного Джураном: наличие удовлетворенности и отсутствие неудовлетворенности.

Порой изменения, необходимые для достижения истинной эффективности, на самом деле влияют не на сам процесс тестирования, а на то, как он воспринимается со стороны. Часто для увеличения степени удовлетворенности заинтересованных сторон в процессе тестирования нужно добиться реалистичных ожиданий. Применение навыков убеждения и дипломатии для формирования ожиданий — еще одна полезная услуга, которую могут оказать тестировщики.

Эффективность означает не только достижение желательных результатов, но и потребление ресурсов в минимально возможном объеме. Имеются в виду не только фиксированные ресурсы (такие как зарплата, затраты на инструменты и базисные системы), но и время. Отсрочка в завершении проекта ведет к тому, что экономисты называют *скрытыми издержками*. На рынке массового потребления мы теряем возможность продажи нашей системы клиентам, готовым за нее заплатить. Работая по контракту, мы упускаем возможность вести работы по следующему выгодному проекту. Что касается внутренних IT-проектов, мы теряем шансы своевременно получить преимущества от применения нашей системы.

Проникновение в проект

Процесс тестирования вовсе не предполагает использование труда небольшого количества сотрудников, изолированных в тестовых лабораториях, в конце проекта. Процесс тестирования состоит из сложных, взаимозависимых совокупностей видов деятельности. Продуманный процесс тестирования включает в себя взаимодействие широкого круга людей на протяжении всего проекта. Успешные процессы тестирования пронизывают весь проект разработки или сопровождения.

- *Всепроникающее тестирование ведется параллельно с другими работами.* Чем раньше в проекте стартует процесс тестирования, тем более полезен он будет для проекта. Обнаруживая дефекты в документах,

описывающих требования и архитектуру, мы предотвращаем внесение ошибок в программу. Мы реалистично оцениваем план проекта, особенно с точки зрения времени и ресурсов, отведенных на тестирование. Мы фокусируем внимание и определяем предстоящие работы за счет тщательного планирования и анализа. Мы создаем надежные системы тестов, пригодные к повторному использованию. Мы применяем эти системы для обдуманного и тщательного выполнения тестов.

- *Всепроникающее тестирование является совместной деятельностью.* Группа тестирования в соответствии с процессом тестирования оценивает и анализирует материалы, передаваемые другими группами. Сотрудники отдела продаж, службы маркетинга, бизнес-аналитики и пользователи определяют требования к продукту. Группа разработки проектирует и создает систему, подлежащую тестированию. Группа управления проектом и руководство компании утверждают плановые графики, бюджет и людские ресурсы. Часто некоторая внешняя по отношению к группе тестирования команда специалистов встраивает компоненты в тестопригодную систему. При наличии сложной тестовой среды помощь группе тестирования в ее установке, конфигурировании и сопровождении может оказать группа сетевого и системного администрирования. Когда эти группы выполняют договоренности в соответствии с графиком и обещаниями, данными группе тестирования, процесс тестирования протекает гладко.
- *Всепроникающее тестирование требует участия специалистов разного профиля.* Сотрудники компании, участвующие во всем цикле разработки или сопровождения, также выполняют часть задач, входящих в процесс тестирования. Основные заинтересованные стороны участвуют в анализе риска качества. Сотрудники групп поддержки заказчиков и операций помогают тестировщикам освоить реально существующие пользовательские сценарии. Разработчики проводят адекватное компонентное и модульное тестирование. Независимая группа тестирования проводит комплексное и системное тестирование системы в целом, охватывающее все компоненты аппаратного и программного обеспечения. Эти компоненты могут быть разработаны различными группами данной компании или даже сторонними организациями. В ряде случаев бета-тестирование, приемо-сдаточные испытания или пилотное тестирование проводят бизнес-аналитики или специалисты, занимающиеся изучением рынка сбыта.
- *Всепроникающее тестирование "перекрестно опыляет" действия тестирования и разработки.* Тестировщики применяют знание предметной области приложения, для того чтобы помочь специалистам по продажам, маркетингу, бизнес-аналитикам и пользователям определить лучшую систему. Используя технические навыки, тестировщики помогают разработчикам лучше спроектировать систему. Тестировщики могут предложить усовершенствования. И тестировщики, и разработчики занимаются такими компонентами сис-

темного тестирования, как заглушки, драйверы, оболочки, генераторы загрузки и автоматизированные скрипты для графического интерфейса пользователя. Инструменты тестирования также имеют значение для групп разработки и эксплуатации. Отчеты об ошибках и описание обходных путей включаются в базу знаний, необходимых специалистам по поддержке пользователей.

- *Всепроникающее тестирование означает обмен информацией.* Группа тестирования анализирует результаты каждого из процессов тестирования, кратко формулирует и докладывает о них различным заинтересованным сторонам. Руководство применяет результаты тестирования в качестве части инструментальной панели руководителей.

Для всепроникающего тестирования необходимо соблюдение ряда условий. Заинтересованные стороны должны реалистично оценивать, какие услуги группа тестирования в состоянии предоставить, и ценить эти услуги. Процесс всепроникающего тестирования связан со многими другими процессами проекта, эти связи подразумевают четкое распределение ролей, ответственности и передачи материалов. И, наконец, общий организационный контекст должен поддерживать и подпитывать процесс тестирования.

Однажды я работал в небольшой недавно основанной компании. Я приступил к работе через месяц после того, как было принято решение о создании группы тестирования и о выстраивании процесса. Я тесно сотрудничал с маркетологами, программистами и высшим руководством, чтобы определить грамотный процесс, нацеленный на смягчение конкретных рисков качества в условиях финансовых и временных ресурсов, которые компания могла себе позволить. Разработчики инструментария тестирования совместно с программистами создали средства для генерации пользовательской загрузки, которые позволили эффективно провести нагрузочное тестирование. Тестировщики совместно с группой, отвечающей за производительность системы, анализировали, моделировали и тестировали серверный кластер, настраивая работу соответствующей пользовательской загрузки в соответствии с прогнозом продаж, предоставленным группой маркетинга. Вместе с ведущими тестировщиками я принимал участие в совещаниях группы управления проектом. Мы готовили и поддерживали в актуальном состоянии инструментальные панели — основное средство измерения хода работ. Руководство, в свою очередь, поддерживало нашу работу. Другие непосредственные исполнители — программисты и инженеры сборок — трудились с нами плечом к плечу для обеспечения гладкой передачи материалов. Процесс тестирования был рациональным и эффективным.

Чем это было обусловлено? Благодарить за это нужно не только тестировщиков, а всю рабочую группу проекта, включая каждого разработчика, менеджера, маркетолога-аналитика и специалиста по поддержке заказчика. Каждый, не жалея сил, делал все возможное, чтобы тестирование внесло существенный вклад в успех проекта.

Использование зрелого процесса

Всепроникающее тестирование часто является (или становится) зрелым. Под зрелым процессом я понимаю процесс, к которому применимы следующие определения:

1. *Эффективный.* Группа тестирования, состоящая из квалифицированных специалистов и оснащенная хорошей системой тестирования, получает полезную информацию на постоянной основе.
2. *Рациональный.* Группа тестирования выполняет каждое задание с первого раза и должным образом, поэтому работу не приходится переделывать. Группа не тратит время на решение задач, не входящих в ее обязанности.
3. *Понятный.* Тестировщики имеют четкое представление о том, чем они занимаются и чем должны заниматься. При этом заинтересованные в проведении тестирования и в обеспечении качества стороны понимают, над чем работает группа тестирования.
4. *Принятый.* Тестировщики и заинтересованные стороны пришли к согласию по важным вопросам.
5. *Организованный.* Каждый тестировщик по отдельности, группа тестирования в целом, группа тестирования совместно с другими командами работают по определенному распорядку, с четкими передачами материалов.
6. *Оптимизирующий.* Тестировщики и другие сотрудники компании собирают информацию, использующуюся для совершенствования процесса тестирования.

Эти определения взаимосвязаны и подпитывают друг друга. Мы долго обсуждали их в данной книге, особенно первые пять.

Оптимизация также имеет большое значение, особенно в долгосрочной перспективе. Даже зрелый процесс по причине прогресса со временем лишается своей зрелости. В области программного и аппаратного обеспечения, разработки систем, так же как и в медицине, естественных науках и других отраслях высоких технологий, прогресс происходит в постоянно ускоряющемся темпе. Либо мы будем шагать с ним в ногу, либо начнем отставать.

Приведу пример, не связанный с вычислительной техникой. Процессы изготовления стали и алюминия дали возможность аэрокосмическим технологиям сделать резкий рывок. Братья Райт для испытания первой удачной модели самолета втащили его на холмы Китти-Хок, запустили двигатель и понаблюдали за происходящим. Сработает ли подобный процесс тестирования для Конкорда или Боинга 777? Точно так же, по мере эволюции технологий, с которыми мы имеем дело, нам, тестировщикам систем, необходимо адаптировать и совершенствовать свои инструменты, применяемые технологии и процессы.

Применение подходящего уровня формализации и стандартизации

Чтобы достичь зрелости процесса, некоторые компании применяют формальные модели процессов и стандарты. Формализация может означать создание формальной модели работы процесса.

Стандартизация может означать следование установленным отраслевым стандартам для процессов и документов. В качестве примеров можно привести руководство по подготовке документов тестирования Института инженеров по электротехнике и электронике (IEEE-стандарт 829) и требования к разработке документации процесса разработки модели зрелости процессов разработки программного обеспечения Института программной инженерии (Software Engineering Institute's Capability Maturity Model, SEI CMM) или Международной организации стандартизации (International Standards Organization, ISO)[1].

Кроме того, стандартизация может подразумевать использование при оценке зрелости процесса тестирования стандартных для отрасли моделей, таких как:

- Совершенствование процесса тестирования (Test Process Improvement) Коомена и Пола
- Модель зрелости тестопригодности (Testability Maturity Model) Гельперина и Хияши
- Модель развития функциональных возможностей тестирования (Testing Capability Maturity Model) Бургесса и Драбика
- Модель зрелости тестирования Бернштейна, Карлсона и Суваннасарта[2]

В некоторых случаях формальные модели процессов, стандартные стратегии тестирования и шаблоны документации образуют хорошие стандартные блоки для успешных проектов тестирования. Роджер Драбик создал модель высокой степени формализации для всего процесса тестирования: с самого начала и до послепроектного анализа и совершенствования. Данная модель, минимальная детализация которой представлена на рис. 17.1, прибегает к диаграммам "ввод-обработка-вывод" для определения всех "существенных элементов формальной программы тестирования проекта среднего (более 20 000 исходных строк кода) или крупного (свыше 100 000 срок кода) масштаба". Драбик не определяет никаких конкретных стандартов документации, но упоминает, что стандарты IEEE 829 вполне пригодны[3].

---

[1] Для получения более полной информации о данных стандартах см. "*IEEE Software Engineering Standards Collection, 1997 Edition*", Paulk и др. "*The Capability Maturity Model*" и Schmauch "ISO 9000 for Software Developers".

[2] Более полную информацию, касающуюся методологии совершенствования процесса тестирования, см. в книге Коомена и Пола "*Test Process Improvement*". Краткий обзор и сравнение этих моделей зрелости см. в статье Драбика "Growth of Maturity in the Testing Process" на сайте www.softtest.org/articles/rdrabick3.htm.

[3] Более подробную информацию о полной модели Драбика вы можете найти в его книге "*Best Practices for Software Testing*" и на www.softtest.org/sigs/material/rdrabick1.htm.

Обучающая ситуация "Влияние зрелости процессов на примере слияния групп медицинского программного обеспечения" предоставлена Робертом Сабуриным, ведущим консультантом компании AmiBug

Я создавал эффективную, ориентированную на предоставление услуг, группу по анализу процессов разработки программного обеспечения (Software Engineering Team, SET) для одной монреальской компании, специализирующейся на выпуске медицинского программного обеспечения. Группа SET предлагала услуги по комплексному и системному тестированию, анализу тестов и тенденций, управлению конфигурацией, анализу измерений и формальной инспекции кода. Заказчиками данных услуг были проекты по созданию программных продуктов. Члены группы SET работали над множеством различных проектов; группа, в которую входило около 10 специалистов, работала каждый месяц над 20 различными проектами.

Наша компания приобрела более мелкую компанию по разработке административного медицинского программного обеспечения. Руководство намеревалось объединить производственные мощности и увеличить долю на местном рынке сбыта. Стратегия интеграции была проста: новая группа разработки станет новой рабочей группой проекта, получающей услуги, общие для всех групп, включая, среди прочих, и услуги нашей группы SET.

Эта новая группа гордилась своими программными продуктами, создаваемыми двумя ведущими разработчиками, работающими непосредственно с клиентами. Документации было мало, процесс разработки велся не систематически, методы тестирования подбирались под текущие задачи, однако программы были удачными. Они применяли жесткий, ориентированный на заказчика процесс: кодирование — исправление — обновление.

Даже несмотря на то, что ориентированная на оказание услуг группа SET стремилась к удовлетворению потребностей внутренних заказчиков из всех рабочих групп проекта, которые прибегали к ее услугам, оказалось сложным делом ввести всего лишь понятие тестовых версий, которые должны передаваться этой новой группой независимой группе тестирования. Отчеты об ошибках превратились в политический вопрос.

В попытках организовать взаимодействие этих по-разному функционирующих групп я потратил много времени, но результат был скромным. Ведущие разработчики самовольно вносили изменения, а затем создавали видимость управления изменениями или регулярной работы над расстановкой приоритетов ошибок.

Тем временем другие начатые проекты, пусть и далекие от совершенства, соблюдали определенный ритм и велись на одном уровне зрелости процесса. Основные этапы были понятными. Код проектировался и создавался. Существовали спецификации требований. Был организован процесс определения приоритетов. Основные взаимодействия между группами велись в благожелательном ключе.

Члены группы SET не хотели работать в новых проектах по созданию программного обеспечения администрирования в области медицины. Им бы пришлось тратить слишком много времени на мифические ошибки, проблемы, связанные с процессом, беспорядочные контакты с разработчи-

ками и на прочие бесперспективные занятия. В свою очередь, сотрудники группы создания программ медицинского администрирования тоже не желали работать с группами, ориентированными на предоставление услуг, заявляя, что им приходится снижать производительность ради ненужных и невыгодных процессов.

Я принял решение не вовлекать группу SET в работы по первому основному проекту медицинского администрирования, а привлечь по контракту внешнюю компанию для параллельного проведения тестирования. Мы не делали измерений. Единственное установленное мною правило по процессу состояло в том, что мы должны быть в состоянии восстановить все, что когда-либо было передано заказчику. Более того, у нас даже не было средств контроля исходного кода.

Проект подошел к концу, и продукт работал довольно хорошо, так что его можно было выпускать. Однако ему требовался более высокий уровень сопровождения. В ближайшие полгода предстояло ликвидировать множество ошибок и поставить много "заплаток". По моим обычным стандартам проект провалился. Но недавно пришедшая к нам команда считала этот подход чутко реагирующим на нужды клиентов. А сами они считали себя героями, которые экономили день каждый раз, когда им удавалось создать новую заплатку для программы или устранить дефект!

На полную интеграцию семейств продукта и подходов к разработке потребовалось три года. Смешанная группа, ориентированная на предоставление услуг, должна вводиться постепенно, по одной услуге за раз, при этом особое внимание должно уделяться зрелости процесса, связанной с этой услугой, с тем, чтобы она не противоречила другим проектам.

В чем урок этой обучающей ситуации? Прибегая к модели оказания услуг, постарайтесь добиться, чтобы все проекты велись на примерно одинаковом уровне зрелости процессов!

Другой формальный процесс тестирования описан в книге Рика Крэйга и Стефана Яскиля "*Systematic Software Testing*". На рис. 17.2 приведена диаграмма этого процесса с минимальной детализацией. Подобно модели Драбика, она охватывает диапазон от первичного знакомства с вопросом до анализа, следующего за выпуском версии. По ходу дела тестирование в ней разбивается на следующие составляющие:

- Уровни, обычно модульное тестирование, комплексное тестирование, системное тестирование и приемо-сдаточные испытания (каждый из них представлен на рис. 17.2 как "Уровень j")
- Фазы в рамках каждого уровня (например, планирование, получение и измерение)
- Действия, выполняемые на этих фазах (например, анализ, проектирование, реализация)
- Задачи, решаемые непосредственными исполнителями для выполнения действий

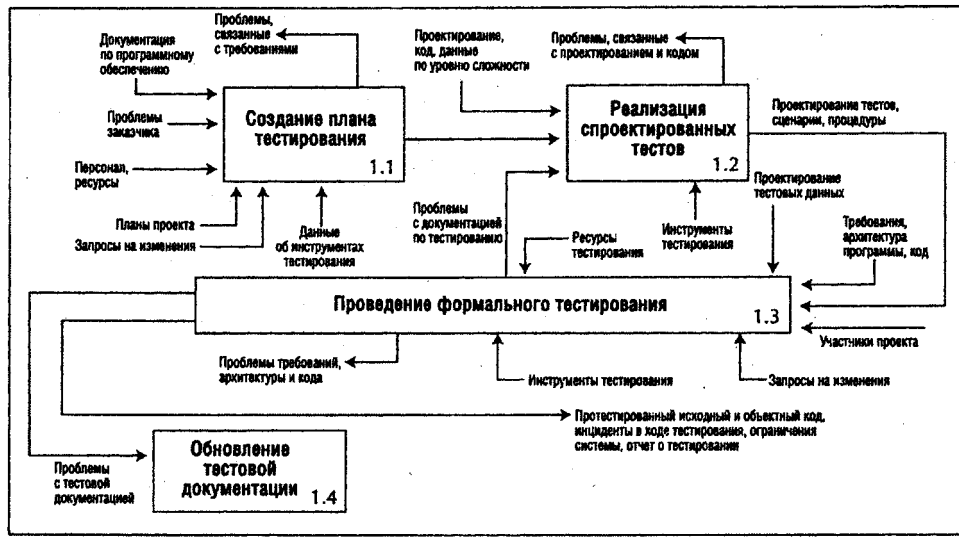


Рис. 17.1. Уровень 1 модели процесса тестирования "ввод-обработка-вывод" по Драбику (Copyright с 1995 by Rodger Drabick. Приводится с разрешения автора.)

В данном процессе тестовая документация ведется по стандарту IEEE 829[1].

Применение формальных моделей в тестировании влияет не только на подход к проведению тестирования и его документирования, но и на способ передачи результатов тестирования. Иногда мы мимоходом рассказываем о результатах тестирования, обсуждая их с заинтересованными сторонами и внутренними заказчиками по электронной почте, оставляя сообщение на автоответчике, в телефонных разговорах один на один и на телеконференциях, останавливаясь поговорить при встречах в коридоре или на проектных совещаниях. При использовании других подходов мы сообщаем о полученных результатах в более структурированной форме, прибегая к письменным отчетам, количественному анализу, графикам, таблицам, измерениям, формальным моделям и структурированным документам. Внедрение формальных методов работы может поставить группу тестирования перед необходимостью более частого предоставления результатов в структурированном виде.

Чтобы принять решение об организации формализованных методов работы и о стандартизации, следует оценить все плюсы и минусы. С одной стороны, есть значительные потенциальные преимущества. Форма-

[1] См. книгу Крэйга и Яскиля "*Systematic Software Testing*". В книге Майкла Шмидта (Michael Schmidt) "*Implementing the IEEE Software Engineering Standards*" описан полный процесс создания и сопровождения программного обеспечения, дополненный обширным подмножеством стандартов IEEE.

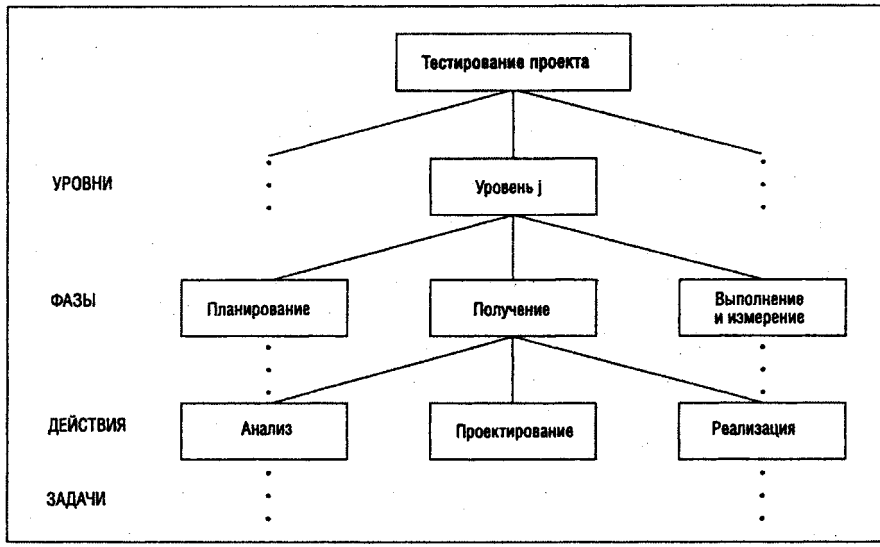


Рис. 17.2. Этапы архитектуры SQE (Copyright c 200 Software Quality Engineering, Inc. Приводится с разрешения.)

лизация и стандартизация процесса тестирования позволяют тестировщикам и тест-менеджерам:

- Снизить беспорядочность и непредсказуемость работ по тестированию.
- Обеспечить быстрое, систематическое обучение новых сотрудников группы тестирования; в некоторых случаях это позволит новичкам в тестировании выполнять те работы, которые в противном случае легли бы на плечи квалифицированного тестировщика.
- Создать и оптимизировать шаблоны и компоненты системного тестирования, пригодные для повторного использования.
- Обеспечить твердую основу для непрерывного совершенствования процесса.
- Прояснить предварительные условия, входные данные, зависимости и выходные данные, необходимые для эффективного и результативного тестирования.
- Подготовить документацию, которая может выступать в качестве юридического доказательства, что тестирование проводилось согласно общепринятой практике.

С точки зрения получения и передачи информации, принятие стандартов и формализация процесса могут:

- Создать полный и непротиворечивый архив данных по проектам для оценки затрат последующих проектов.

- Создать общепринятые формы для отчетов о результатах тестирования, позволяющие всей рабочей группе проекта и даже компании в целом общаться на одном языке.
- Снизить риск того, что не будет собрана информация, которая является критичной для тестирования.

Еще одним преимуществом для некоторых групп тестирования является соответствие определенным стандартам. В ряде случаев при разработке системы служба внутреннего контроля, заказчики, правительственные или другие полномочные органы устанавливают определенный стиль ведения документации и модель процесса, которые не подлежат обсуждению. Для того чтобы получить разрешение на выпуск или поставку системы, рабочая группа проекта должна следовать определенным правилам. Например, компании, продающие медицинские системы в США, должны работать в соответствии с положениями Федерального Управления по контролю над продуктами и лекарствами правительства США. Иногда предприятия, приобретающие системы или компоненты систем, накладывают на своих поставщиков определенные требования, что приводит к формализации или стандартизации процесса тестирования. Приведу два примера: чтобы иметь возможность заключать оборонные контракты с армией США, компания должна достичь уровня зрелости, оцениваемого по модели развития функциональных возможностей (CMM), тогда как многие крупные компании, приобретающие компьютеры на базе Intel, настаивают на сертификации по стандартам Microsoft. И, наконец, в некоторых случаях стремление к соответствию определенным стандартам диктуется скорее желанием получить преимущества на рынке, нежели соображениями соответствия. Например, оффшорные компании получают сертификат CMM или ISO 9000, чтобы потенциальные заказчики могли быть уверены в качестве предлагаемых услуг.

С другой стороны, существуют серьезные подводные камни, связанные с переходом к работе по стандартам и формальным процессам. Во-первых, экстенсивная формализация и стандартизация, как правило, требуют больших финансовых вложений. Время, проведенное в обсуждении шаблонов, разработке формальных моделей процессов тестирования, исследовании и применении стандартов, спорах по поводу темпов преобразования и подходов к изменению процесса и т.п. — это время, не используемое на собственно тестирование. В некоторых случаях подобные инвестиции могут со временем окупиться, но для многих компаний в рамках одного проекта подобные непроизводительные расходы просто недопустимы. Несложные методики, например использование списков контрольных вопросов, в состоянии помочь снизить затраты.

Помимо первоначальных затрат, слишком жесткая стандартизация или формализация может снизить производительность. Люди порой слепо следуют процессам, и в результате работа ведется непродуктивным или даже антипродуктивным образом. Тестировщики могут тратить много времени на заполнение пустых строк в шаблонах документов, не задаваясь вопросом: каким образом документирование сведений способствует лучшему проведению тестирования? Часто рассказывают притчу о сыне, который спрашивает свою мать, почему она всегда отрезает часть от куска свинины, прежде чем начать ее готовить. Мама отвечает: "Не знаю, так меня научила

твоя бабушка". Сын спрашивает у бабушки, и та ему рассказывает: "О, когда твоя мама была маленькой, у нас не было такой большой жаровни, чтобы зажарить все мясо целиком, вот мне и приходилось отрезать кусочек, чтобы оно поместилось". Необходимо иметь гибкость и знать, как выявлять исключения из правил и обходиться с ними.

Особенно важны гибкость и продуманный подход в случае, если вы адаптируете под свои нужды модели процессов, стандарты и шаблоны, созданные другими людьми. У разных людей разный стиль работы, и эти различные подходы незаметно влияют на процессы и стандарты. В хорошо работающих компаниях, например в тех, что уже применяют зрелые процессы на протяжении всего жизненного цикла системы, можно ожидать минимальных усилий при обращении к стандартному процессу тестирования, совместимому с уже имеющимся процессом разработки. Однако я несколько раз работал в таких компаниях, где попытка одним махом внедрить формальные методы и стандартную документацию привела бы к полной неспособности тестировщиков приносить какую-либо пользу. Изучая методы, используемые другими, основательно обдумайте допущения и предпосылки.

Поступая подобным образом, вы, кроме того, морально подготовитесь к наиболее разрушительной опасности на пути эффективного внедрения моделей формальных процессов, стандартов или вообще любых усилий по совершенствованию процесса: эффекта серебряной пули. Некоторые полагают, что шаблоны и функциональные схемы решают все проблемы, связанные с тестированием. Например, в ходе написания этой главы я получил кое-какую литературу по маркетингу от одной компании, занимающейся консалтингом и обучением, с описанием их процесса тестирования. Процесс представлял собой патентованную смесь модели SEI CMM и процессов и шаблонов IEEE. Они утверждали, что если тестировщики будут следовать их формальному подходу, то действия тестирования будут удовлетворять потребности любого проекта по разработке программного обеспечения или превосходить эти нужды.

Подобные заявления и подобный образ мысли опасны.

Можно усовершенствовать существующий процесс тестирования, изучив имеющиеся ключевые процессы тестирования и среду, в которой происходит работа. Можно разумно применять идеи из этой или другой книги, статей и документов. Формализация и стандартизация в состоянии принести значительную выгоду, уровень же усилий и объем документации при этом часто бывает вполне приемлемым.

Однако единственно верного процесса не существует. Вы не должны непременно применять громоздкие диаграммы "ввод-обработка-вывод" или делать отчеты, совместимые со стандартом IEEE 829. Возможно, будет достаточно всего лишь составления простых списков контрольных вопросов для каждого ключевого процесса и приведения к единообразному виду измерений, графиков и отчетов всех тестировщиков компании. Поступайте так, как в вашей ситуации представляется разумным.

Использование подходящих стратегий

В этой книге я пропагандирую стратегию тестирования, основанную на рисках, и описываю, как я ее использую. Некоторые тестировщики счита-

ют, что системные тесты должны строиться исключительно на спецификациях требований и архитектуры, а сами спецификации должны однозначно отвечать на любой вопрос по поводу корректности поведения системы. Документы с описанием требований и архитектуры и, конечно, тестовые сценарии являются полезными моделями того, как должна работать система. Однако в конечном счете это всего лишь модели -- полезные, но упрощенные. Поэтому не будем заставлять себя и своих тестировщиков ограничиваться этими моделями и строго следовать им.

В главе 15 упоминалось, что разработка или сопровождение проекта имеет успех, когда соединяются четыре составляющие:

- *Свойства.* Создаем ли мы необходимое множество функций, которые решают проблемы пользователей и заказчиков?
- *Сроки.* Можем ли мы передать в заданные сроки систему, решающую эти проблемы.
- *Бюджет.* Можем ли мы передать систему способом, который является привлекательным с финансовой точки зрения для заказчиков, пользователей, спонсоров проекта и создателей системы?
- *Качество.* Можем ли мы передать свойства в состоянии, которое подходит для применения пользователями и заказчиками?

Если вы можете утвердительно ответить на все четыре вопроса, проект будет успешным.

На самом же деле мы узнаем ответы на эти вопросы только после того, как по окончании проекта пройдет некоторое время. До или в ходе проекта мы можем говорить лишь об уровне уверенности или вероятности успеха. Другими словами, на протяжении всего жизненного цикла проекта мы рискуем тем, насколько успешным он окажется. Потому мудрые менеджеры проекта применяют управление рисками.

Тестирование является частью разумной стратегии по управлению рисками проекта. На протяжении этой книги я делал акцент на способах увязывания тестирования с рисками и со снижением рисков. Были показаны различные инструменты и методики предоставления услуг и информации, призванные помогать рабочей группе проекта в управлении рисками, связанными с качеством системы.

Не все прибегают к стратегии, основанной на рисках. Ниже приводится краткий список некоторых других стратегий тестирования.

- *Исчерпывающая.* Тестируйте все, как можно полнее.
- *Вынужденная.* Тестируйте что угодно и где угодно; распределяйте работы по тестированию на весь продукт.
- *Охота за ошибками.* Используйте профили, систематизацию ошибок и интуицию, чтобы нацеливать тестирование на те места, где водятся ошибки.
- *Список контрольных вопросов.* При тестировании руководствуйтесь списками контрольных вопросов, созданными в течение определенного времени.
- *Управляемая программистами.* Тестируйте согласно указаниям программистов и рассматривайте их как высший авторитет на предмет правильных действий.

- *Управляемая заказчиками или пользователями.* Тестируйте в соответствии с тем, что, по мнению людей, оплачивающих систему, или экспертов в данной отрасли, которые будут использовать систему, нуждается в тестировании.
- *Управляемая реализацией.* Изучайте архитектуру, код, схемы системы и т.п., чтобы определить, на чем сконцентрировать внимание при тестировании.
- *Аналитическая.* Соберите всю имеющуюся информацию, изучите структуру системы, установки и цели; проводите тестирование на основе этих обширных и глубоких знаний.
- *Интуитивная.* Тестируйте в соответствии с тем, что говорят коллективный опыт, мудрость и интуиция сотрудников группы тестирования на предмет того, что следует тестировать.
- *Исследовательская.* Одновременно изучайте систему, тестируйте, находите ошибки и совершенствуйте подход к тестированию, корректируйте его направленность.
- *Быстрая.* Тестируйте, применяя облегченные процессы, в поиске вероятных ошибок ориентируйтесь на готовность реагировать на недавно внесенные изменения.
- *Функциональная.* Тестируйте каждую функцию системы по очереди.
- *Основанная на требованиях или претензиях.* Тестируйте каждое требование или претензию к продукту.
- *Автоматизированная регрессия.* Тестируйте, применяя исключительно автоматизированные тесты, которые можно повторить при необходимости проверки на наличие регрессии.
- *На основе состояний системы.* Тестируйте различные состояния и переходы из одного состояния в другое, которые могут иметь место.
- *На основе сценариев использования системы.* Тестируйте в соответствии с различными возможными в реальной жизни сценариями по всем областям функциональности системы.
- *На основе областей системы.* Анализируйте различные области входных и выходных данных и обработки данных, которые есть в системе, проверяйте наиболее репрезентативных представителей каждой области.
- *Массовая случайная.* Тестируйте огромное количество совокупностей данных, выбранных случайным образом, при помощи автоматизированных инструментов.
- *На основе качества.* Тестируйте, ориентируясь на важные свойства системы, такие как функциональность, удобство и простота использования, надежность, производительность, масштабируемость и т.д.[1]

---

[1] Этот список появился в результате онлайнового обсуждения вопроса с Россом Колладом, Семом Канером, Кэти Айберли и Камешем Пеммараджу. Благодарю их за содействие.

Возможно, вы заметили, что многие из этих стратегий были показаны в представленных мной процессах. Несмотря на то, что основная движущая сила моего подхода к тестированию основана на рисках, все другие стратегии являются полезными инструментами, применяемыми по мере необходимости. Кроме того, невзирая на то, что подход к работе, основанный на рисках, во многих случаях эффективен, вам следует продумать, не будет ли другая стратегия более подходящей в ваших условиях.

Успешное внедрение инструментов

Разработка инструментов тестирования — бурно растущий бизнес, однако часто попытки внедрить их оказываются неудачными. Почему? Цитирую высказывание Фьюстера и Грэм: "Автоматизация неразберихи всего лишь быстрее приведет вас к неразберихе". Наличие основательного, повторяемого, хорошо понятного процесса становится залогом успеха использования инструментов автоматизированного тестирования[1].

Это особенно важно по мере нарастания сложности проекта. На рис. 17.3 показано, каким образом в проектах с низким уровнем сложности квалифицированные инженеры (включая тестировщиков) при помощи верно выбранных методов могут самостоятельно справиться с трудностями проекта и "пробить проекту дорогу", чтобы предоставить систему пользователям и заказчикам. На рис. 17.4 показано, что по мере нарастания сложности необходимо поддерживать "мост" между пользователями и рабочей группой проекта, чтобы предотвратить его обрушение под давлением, оказываемым на него проектом, и этой поддержкой начинает заниматься процесс. И, наконец, из рис. 17.5 видно, что мы можем укрепить процесс разработки, сопровождения или приобретения, в том числе про-

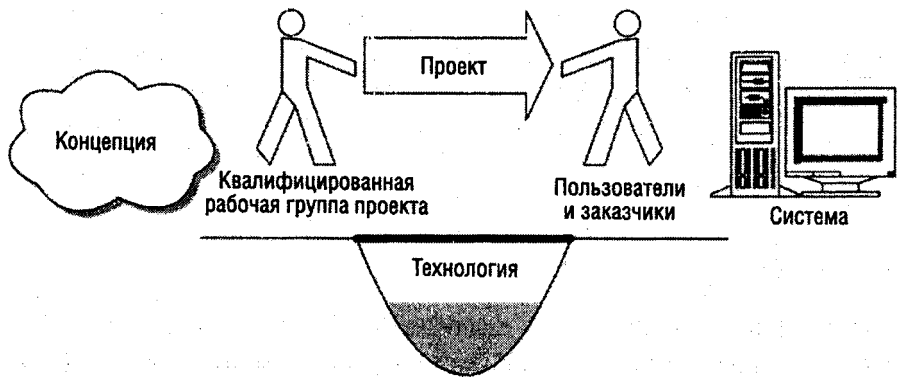


Рис. 17.3. Навыки и методы возводят мост в простых проектах

[1] См. книгу Марка Фьюстера и Дороти Грэм "*Автоматизация тестирования программного обеспечения*".

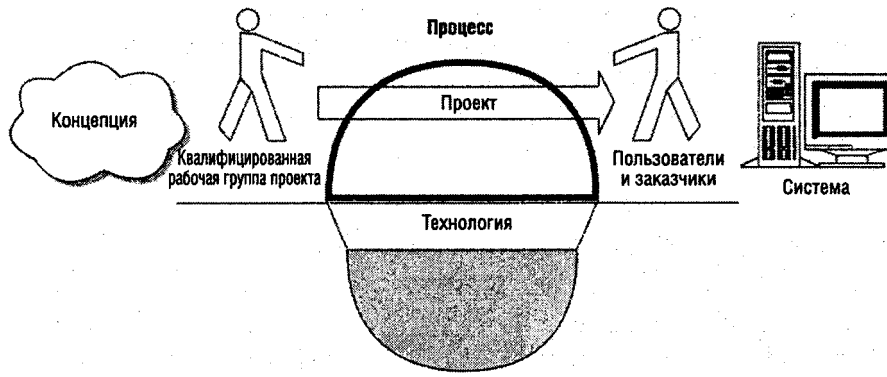


Рис. 17.4. По мере нарастания сложности процесс должен усиливать навыки и методы

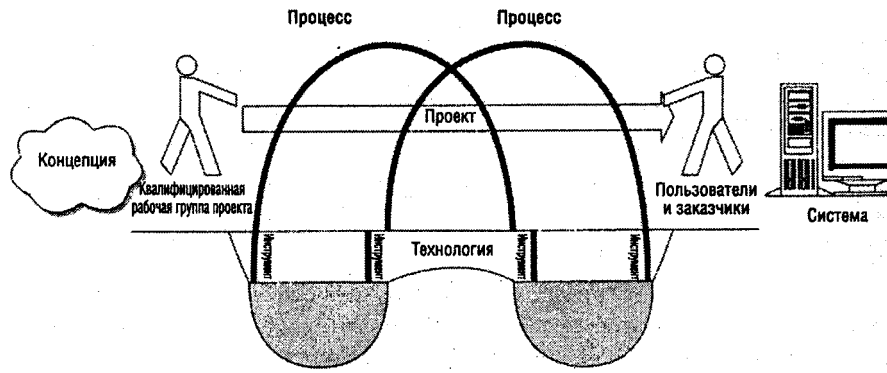


Рис. 17.5. Грамотный процесс поддерживает успешное внедрение инструмента

цесс тестирования, инструментами, чтобы справиться с большими трудностями в очень сложных проектах. Разумное сочетание квалифицированных и увлеченных работников, использования передовой, но устойчивой технологии, повторяемых и понятных процессов и должным образом применяемых инструментов позволит нам довести сложные проекты до успешного завершения.

Получайте удовольствие от вдохновения и от возможности непрерывно учиться

Один из этапов процесса роста команды, обсуждаемых в книге, включает в себя программу повышения квалификации, подразумевающую обуче-

ние и последующее применение изученных знаний в выполнении задач тестирования. В нашем примере Лин Цу должна была участвовать в конференции, изучать управление документацией и затем на основе полученных знаний разработать несколько тестов. Это одна из форм обучения: начать что-то делать, приобрести некоторые новые навыки и закрепить полученные знания, применив их.

Кроме того, мы можем осваивать новое, изучая то, что сделано другими. Инспекции, рецензирование документации, кода и тестов являются не только методами контроля качества, но также дают их участникам отличную возможность узнать что-то новое. Программирование и тестирование, проводимое совместно двумя сотрудниками, находящееся пока на стадии эксперимента, столь же многообещающе в плане обучения напарников друг друга.

Когда мы учимся делать что-то самостоятельно, это еще одна форма обучения. Отец научил меня игре в шахматы, играя вместе со мной. Так же я научился кататься на велосипеде. Я не изучал теорию шахматной игры или езды на велосипеде. Я не пытался понять, как другие играют в шахматы или катаются. Я просто начал играть в шахматы и ездить на велосипеде, удачи и поражения помогли мне в учебе.

Порой нам в голову приходят удачные идеи. Однажды, работая над системой автоматизированных тестов, я пытался выбрать верный синтаксис для набора тестов и для определения тестовых сценариев. В это же время я готовил документ в формате HTML. Этот документ не относился к проекту. Он просто входил незначительным пунктом в список того "что нужно сделать", и я занимался этими вопросами, ожидая, пока в голову придет хорошее решение главной проблемы. И, конечно, когда я взглянул на тэги (например, и — начало и окончание полужирного шрифта) в файле HTML, я понял, что могу использовать подобные метки. Так, я выбрал <Action> и </Action> для того, чтобы отметить начало и конец фрагмента, показывающего инструменту тестирования его часть действия в тестовом сценарии. Такие моменты открытий иногда происходят, когда мы отвлекаемся от поиска решения своей проблемы, как в случае с Архимедом, пожелавшим принять ванну.

Обучение может быть беспорядочным. Обучением иногда трудно управлять. Но оно всегда в радость. Что еще важнее, оно побуждает к действию. Как часто, проводя собеседования, я слышал вариации на тему "чего я больше всего хочу от этой работы, так это возможности продолжать узнавать что-то новое". Люди с мотивацией и жаждой знаний лучше работают. Более того, с каждым днем их труд становится изящнее и эффективнее. Поэтому в роли менеджера я способствую обучению и поощряю творческое начало. Я поощряю вдохновение. Я работаю над тем, чтобы сделать обучение, творческий подход и вдохновение — и неизбежные ошибки, которые с этим связаны, — безопасными. Я выяснил, что результатом является приносящая положительные эмоции, мотивирующая и продуктивная среда для работы

Решение проблем

Мы рассмотрели ряд сложных проблем, возникающих в конкретных процессах тестирования. Общий процесс тестирования также сопровождается многочисленными проблемами[1].

Как измерить то, чему мы не в силах дать определение?

Вернемся к определению качества, которое я позаимствовал у Джурана: наличие поведения, удовлетворяющего пользователя, и отсутствие поведения, не удовлетворяющего пользователя. Используя это определение, нам, возможно, следовало бы измерить ряд полезных свойств и ряд ошибок, найденных пользователями. Может быть, эти два показателя измеряют качество? Я отношусь к этому скептически по двум причинам.

Во-первых, для качества системы в равной степени важно, какие ошибки остались и сколько их осталось. Знаем ли мы о качестве все необходимое, если единственное, что нам известно, — это то, что к моменту передачи продукта мы устранили 90% ошибок? Уверены ли мы, что система удовлетворит потребности 90% пользователей и заказчиков? Может быть и так, если все оставшиеся дефекты влияют на одних и тех же пользователей и заказчиков и их 10% от общего числа. Но предположим, что у нас имеется один дефект, оказывающий воздействие на 20% заказчиков. Подобные рассуждения превращают результаты измерения ряда полезных характеристик в далекие от совершенства и неполные показатели. Поэтому, хотя эти цифры могут быть параметрами в некоторых уравнениях качества (очевидно, предпочтительнее иметь больше полезных свойств и меньше ошибок), они не являются единственными и не могут выступать в роли двух основных параметров.

Второе возражение основано на том, что не все согласны с определением Джурана. Журнал "Quality Progress", принадлежащий American Society of Quality, опубликовал статью Роберта Хойера и Бруки Хойер под названием "Что такое качество?". В ней рассматривались восемь различных определений, которые дали восемь гуру в области качества. Приведу некоторые из определений, упомянутые в этой статье:

- Фил Кросби: "Соответствие требованиям".
- У. Э. Деминг: "Качество [существует] в группах посредника [заинтересованного лица] ..."
- Геничи Тагучи: Качество — это "потеря ... [нанесенная] обществу" продуктом или системой.

Р. Хойер и Б. Хойер делят эти разнообразные определения на две группы. Определения, отнесенные ими к уровню 1, провозглашают, что качество — это предоставление продуктов и услуг, "измеримые характеристики которых удовлетворяют фиксированному набору [численно]

---

[1] Трудности, связанные с общим процессом тестирования, удачно отражены в книге Билла Пэрри (Bill Perry) и Рэнди Райса (Randy Rice) "*Surviving the Top Ten Challenges of Software Testing*".

определенных спецификаций." Определения уровня 2 характеризуют качество как "независимое от ... измеряемых свойств" и берущее начало в "[удовлетворенных] ожиданиях заказчика"[1].

Если мы выбираем подход уровня 1, мы можем измерить характеристики и затем разработать показатели качества, таблицы или модель, основанные на этих измерениях. Эти показатели, таблицы или модель могут быть неточными и неопределенными, однако, возможно, окажутся полезными в качестве ориентира.

Если мы соглашаемся с подходом уровня 2, то можем допустить, что подобные измерения вредят достижению основной цели обеспечения качества — удовлетворению пользователей и заказчиков. Выбор данного подхода не означает, что мы должны прекратить измерение качества. Он означает, что мы должны тестировать эти показатели на предмет соответствия стандартам удовлетворенности заказчика и пользователя.

Незрелые процессы разработки и сопровождения

Ранее я рассматривал проникновение процесса в проект и его зрелость как взаимосвязанные свойства успешного процесса тестирования. Успешный процесс тестирования действительно будет охватывать весь проект, однако обратите внимание, что проект всегда проникает в процесс тестирования. Когда проект в целом ведется в условиях недостаточной зрелости процесса, процесс тестирования становится беспорядочным и непредсказуемым, с пониженной эффективностью и производительностью.

Я люблю самостоятельно оценивать проект, пользуясь моделью CMM. После изучения ключевых областей процесса вам, возможно, потребуется не более часа, чтобы понять, ведется ли процесс на Первоначальном, Повторяемом, Описанном, Управляемом или Оптимизирующем уровне. (Будьте тактичны в применении сведений, полученных вами таким образом.) Низкий уровень зрелости проекта часто указывает на то, что с процессом тестирования будут сложности.

Тестировщики, работающие в проектах с низким уровнем зрелости процессов, сталкиваются с конкретным риском, связанным с восприятием качества, а значит, и усилий, потраченных на его оценку и обеспечение, как предмета роскоши. Компании не могут создавать программы без программистов, работать с сетью без операторов сети, выпускать продукт без инженеров сборок или оказывать помощь без сотрудников службы технической поддержки, но компании *могут* (а некоторые так и *делают*) продавать и использовать системы, не тестируя их. В тяжелые времена именно от предметов роскоши отказываются в первую очередь. По мнению авторов "*The Capability Maturity Model*", в незрелых компаниях "плохо понимают, каким образом конкретные этапы процесса создания программного обеспечения влияют на качество, и качество продукта трудно прогнозировать. Более того, виды деятельности, нацеленные на повышение качества, такие как рецензирование и тестирование, часто сокращаются или исключаются, когда проект выпадает из запланированного графика"[2].

R.W.Hoyer and Brooke Hoyer "What is Quality?", *Quality Progress*, July 2001.

Mark Paulk и др. "*The Capability Maturity Model*", с. 7.

Неэффективность работы, являющаяся следствием низкой зрелости проекта, в сочетании с восприятием со стороны руководства тестирования как излишества делает проекты с низким уровнем зрелости опасными для тестировщиков. В таких случаях больше говорите с вашими и другими менеджерами об их взглядах на тестирование и ожиданиях от его проведения, и обратите внимание на признаки колебаний, когда дела пойдут неважно.

Зная, какое влияние могут оказать незрелые процессы разработки на группу тестирования, квалифицированные тестировщики порой пытаются руководить совершенствованием процесса разработки. Это опасно. Скорейший и наиболее эффективный из известных мне способов потерять уважение и сжечь политический капитал — стать назойливым тестировщиком, который бегает по компании и указывает другим, как им делать свое дело. Если к подразделению, отвечающему за тестирование, не относятся как к первой скрипке, то вы будете выглядеть ханжой, которому не следует совать нос не в свое дело. Если тестирование в компании представляет собой хорошо отлаженный механизм, то вас, вероятно, будут считать надоедливым всезнайкой. Если вы чувствуете, что необходимо изменить процесс в рамках всей компании, возможно, вы сумеете найти единомышленников на уровне топ-менеджеров, которые возглавят это дело.

Нереалистичные ожидания

Даже в компаниях с высоким уровнем зрелости часто имеется неверное представление о тестировании. Поговорив со своими менеджерами о том, что они ожидают от тестирования, вы, возможно, выясните, что в вашей компании тоже отсутствует верное понимание тестирования. В запущенных случаях менеджеры полагают, что тестирование представляет собой распыление на систему некоего волшебного порошка качества, с помощью которого можно избавиться от ошибок (подобно пестицидам, уничтожающим тлю).

Я знаком с одним тест-менеджером, получившим плохую оценку по результатам ежегодной аттестации и лишенным премии, поскольку его компания выпустила полный ошибок продукт. В свое время он докладывал о наличии большинства этих ошибок, они были отражены в базе дефектов, но ему отвечали, что эти ошибки не существенны. Вскоре после этого он успешно нашел новую работу, где люди охотно расставались со своими нереалистичными ожиданиями по поводу тестирования. Менеджер одного из проектов, которым я занимался, рассчитывал, что мы найдем все ошибки за один раунд тестирования, затем программисты за несколько дней устранят их, и следующий раунд тестирования подтвердит, что все найденные ошибки устранены, и не выявит новых.

Нереалистичные ожидания могут принимать много форм. Одно из распространенных обличий нереалистичных ожиданий связано с тем, что тестирование воспринимается как в некотором роде тривиальная задача. Работая над этой книгой, я получил предложение внедрить в компании процесс тестирования, включая полностью автоматизированное регрессионное тестирование. Я ответил, что вместе со своими ассистентами справлюсь с этой задачей. Тогда они добавили, что хотят, чтобы эту рабо-

ту провел одни тестировщик средней квалификации за шесть недель. Когда же я ответил, что мы, возможно, сумеем провести грамотный анализ рисков, дать реалистичную оценку положения дел и составить продуманный план действий в два раза быстрее, но внедрение займет месяцы работы, они решили обратиться к кому-нибудь другому.

Целое семейство нереалистичных ожиданий обуславливается неверным представлением, что тестирование должно быть в состоянии получить полное покрытие при использовании минимальных ресурсов. Я, например, не знаю, как выявить проблемы производительности иначе, чем провести тестирование в среде, аналогичной среде пользователя, что иногда дорого обходится. Если компания считает, что не стоит так тратиться на то, чтобы заранее узнать об ошибках производительности, ей не следует проводить подобное тестирование. Однако менеджерам, принимающим это решение, следует понимать, что они согласились на увеличение риска качества за счет снижения риска не уложиться в рамки бюджета, и что, возможно, придется жалеть о таком компромиссе.

Другое нереалистичное ожидание в отношении группы тестирования, с которым я сталкивался, заключено в "древней" заповеди руководителей: "Не говорите мне о проблеме, если не можете предложить, как с ней справиться". Эта тактика может быть разумной применительно к некоторым менеджерам и командам, поскольку она позволяет предотвратить бесконечные попытки избежать ответственности. Однако группа тестирования — особенная: мы существуем лишь для того, чтобы предоставлять беспристрастную, точную информацию о текущем состоянии качества системы. Часто в системе, которую мы тестируем, на самом деле много ошибок. Существуют как разумный, так и неразумный способы сообщить об этом, однако со стороны менеджеров нереалистично было бы ожидать от тестировщиков готового решения по поводу каждой найденной ошибки. Это всего лишь еще одна ситуация, где под личиной стремления поддержать командный дух скрывается ожидание волшебного порошка качества. Если тестировщикам не дают возможности заявлять о проблемах в тех случаях, когда им не известны пути устранения этих проблем, проект лишается большинства преимуществ наличия в нем независимой группы тестирования.

Часть проблем, связанных с ожиданиями, возникает из-за смешения функций тестирования и контроля качества. Настоящая группа по контролю качества имеет многообразные обязанности и полномочия, позволяющие ей добиться того, чтобы процессы разработки и сопровождения завершились созданием качественной системы. Группа тестирования оценивает качество. Это понятные роли и задачи, однако мне приходилось сталкиваться с тем, что от группы тестирования ожидали *обеспечения качества.*

(Если вы настоящий, добросовестный менеджер по контролю качества, в ваши обязанности входят процессы, не описанные в этой книге. Кроме того, процесс подбора персонала будет отличаться от процесса, представленного в главах 8 и 9. И, наконец, сфера вашей компетенции должна быть не такой, как у типичного тест-менеджера. Обеспечение качества подразумевает понимание всего процесса разработки программного обеспечения, углубленные навыки в программировании и другие техни-

ческие знания, обучение управлению качеством и знание формальных методик по совершенствованию процесса, таких как ISO 9000, CMM и тотальное управление качеством.)

Между компанией и группой тестирования должно существовать своего рода соглашение по поводу того, какие услуги предоставляет группа тестирования. Если ваша компания определила это соглашение с помощью официально сформулированных целей и задач, используйте их. Если компания прибегает к управлению в соответствии с целями (management-by-objective) на уровне тест-менеджеров, значит, ориентируйтесь на это. Ваша задача — избежать неверной трактовки вашей роли, особенно со стороны старших менеджеров и руководства компании.

За годы работы я сталкивался со многими менеджерами проекта, которым требовалась помощь в понимании истинных возможностей тестирования и его потенциального вклада в общее дело. Большинство из них с готовностью выслушивали меня. Некоторые не были готовы к восприятию моей точки зрения. Если вы сталкиваетесь с нереалистичными ожиданиями, попытайтесь понять мотивацию людей, их высказывающих. Что для них представляет ценность? О чем они беспокоятся? Чем они занимаются в свободное время, и можно ли использовать сравнения, чтобы облечь вашу точку зрения в близкую для них форму. Общайтесь на понятном им языке.

Если это не действует, если кто-то упрямо цепляется за свои нереалистичные ожидания, тогда уже вам придется реально оценить собственные возможности. Вы можете продолжать работать с этим человеком, зная, что время от времени вас будут в чем-то несправедливо обвинять. Или же вы можете найти другую работу[1].

Взаимоотношения

Слишком часто я слышал и сталкивался с ситуациями, в которых роль группы тестирования рассматривалась, по сути, в виде противостояния остальным группам проекта. Тем не менее, группы тестирования часто являются подразделениями, предоставляющими услуги. Сервисные подразделения либо удовлетворяют потребности заказчиков, либо выходят из игры. Очевидно, что отношения, построенные на соперничестве, указывают на отсутствие удовлетворенности. Хотя вопрос о том, как в согласии работать с другими людьми, выходит за рамки данной книги, я могу предложить несколько коротких советов по поводу этих проблем применительно к специфике работы тестировщика.

Прежде всего, убедитесь в том, что вы контролируете все возможные нереалистичные ожидания. Многие из проблем во взаимоотношениях с руководителями берут начало в нереалистичных ожиданиях. Во-вторых, совместно со своими коллегами и руководителями проясните все роли, отношения и передачи материалов, влияющие на процесс тестиро-

---

[1] См. мою статью "Stop Destroying My Team with Bad MBOs: A Manager's New Year's Resolution" на www.stickyminds.com, где приведены размышления по поводу того, как нереалистичные ожидания превращаются в измеряемые цели производительности, и показано, почему это плохо.

вания. В-третьих, подумайте о том, кто является вашими внутренними заказчиками и заинтересованными сторонами. Вот несколько примеров:

- Разработчики, нуждающиеся в помощи в тестировании и отладке
- Сотрудники служб маркетинга и продаж, стремящиеся продвинуть на рынок и продать качественные системы
- Бизнес-аналитики и заказчики, которые хотят получить качественные системы
- Сотрудники службы технической поддержки, не желающие иметь дело с сердитыми заказчиками
- Администраторы сети, которые хотят, чтобы серверы работали надежно и были масштабируемы

Вам удастся выстроить надежные взаимоотношения с этими людьми, если вы сможете предложить им конкретные услуги.

И последнее. Займитесь созданием крепких взаимоотношений на ранних этапах проекта, прежде чем проявятся паника и стресс, так часто сопутствующие конечным стадиям проекта. Такие взаимоотношения будут приносить отдачу на протяжении всего проекта, не только делая вашу работу менее нервной, но и превращая процесс тестирования в более эффективный и производительный как с объективной точки зрения, так и в глазах ваших коллег.

"Привычка тестировать"

Некоторые компании организуют независимые группы тестирования, пытаясь справиться с особенно тяжелым (и дорогим) проектом или обеспечить составляющую работ по совершенствованию процесса разработки программных продуктов, направленного на улучшение качества. Как правило, новая группа тестирования призвана концентрировать усилия на поведенческом тестировании, часто на этапах комплексного и системного тестирования, а также во время приемо-сдаточных испытаний, тогда как программисты по-прежнему отвечают за структурное тестирование и за большую часть работ на этапах компонентного и модульного тестирования.

Однако иногда лишь первая часть данного мероприятия "пускает корни", в то время как тестирование, которое должны проводить программисты, сходит на нет. Программисты, ранее считавшие себя ответственными за все тестирование собственного программного продукта, перестают что-либо делать, размышляя так: "Ага! Мне больше не нужно тестировать, это работа группы тестирования".

Мой коллега Рэн Макнари говорит о такой ситуации, что программисты "привыкли тестировать". Но когда они прекращают этим заниматься, качество системы, поступающей тестировщикам, становится более низким, чем до организации независимого тестирования, когда система поступала к пользователям или заказчикам непосредственно от группы разработчиков. В результате возникает последовательность из трех чрезвычайно опасных ситуаций, одна за другой ударяющих по проекту и, разумеется, по группе тестирования.

Во-первых, интеграция разнообразных компонентов после их завершения становится сложной. Поскольку сами компоненты содержат проблемы, на то, чтобы "заставить" их работать вместе, требуется много времени, и в ходе подобных попыток выявляется много проблем. Эти проблемы сложно отделить друг от друга, что порождает споры, в которых программисты, создавшие "недружественные" компоненты, обвиняют друг друга в ошибках. И лишь благодаря титаническим усилиям со стороны руководства интеграция продолжается. Вследствие того что затраты на интеграцию компонентов в любом случае часто недооцениваются в планах-графиках проекта, это может привести к существенным и непредвиденным отставаниям в середине проекта.

Во-вторых, когда с интеграцией системы справляются, ее передают независимой группе тестирования в чисто функциональном виде, и она переполнена критическими ошибками. Из-за отставания по срокам задерживается начало комплексного и системного тестирования, причем соответствующего сдвига в окончании работ по тестированию часто не происходит, поскольку руководство объявляет дату поставки проекта или выпуска системы неприкосновенной. Появляется надежда, что "во время тестирования мы наконец получим передышку", но на самом деле тестирование занимает значительно больше времени, чем первоначально планировалось. Это происходит не только потому, что, так же как интеграция, тестирование недооценивается, но и потому, что продукт оказывается хуже ожиданий, а значит и тестирование идет значительно медленнее, чем могло бы. Критические блокирующие ошибки приводят к невозможности нормального прохождения тестов согласно плану. Новые тестовые сборки показывают отсутствие ключевых функций, в некоторых случаях сборки даже не могут быть установлены. Нередко именно с этими проблемами связан полный крах того или иного проекта, когда терпение руководства, финансовые возможности или даже жизнеспособность компании в целом исчерпывается.

В-третьих, когда необходимо сплотиться вокруг проекта и развернуть или выпустить систему, результаты нередко бывают неутешительными. Группа тестирования тратит большую часть времени и сил на поиск ошибок при помощи поведенческих методик, в то время как те же ошибки можно было выявить раньше и с меньшими денежными затратами, если бы программисты провели тестирование структурными методами. Таким образом, они находят меньше ошибок, что означает, что больше ошибок останется в программе. Когда к этим трем ситуациям добавляются нереалистичные ожидания по поводу группы тестирования, руководство иногда возлагает всю вину на тестировщиков[1].

Чтобы этого не произошло, начните с устранения каких бы то ни было нереалистичных ожиданий, сложившихся в головах ваших коллег отно-

---

[1] Существует показатель, который вы, будучи тестировщиком, можете применять для количественной оценки эффекта чрезмерной веры в тестирование на поздних этапах, он называется "*процент выявления дефектов*". Узнать об этом показателе более подробно можно из моей книги "*Managing the Testing Process, Second Edition*". В статье Элизабет Хендриксон (Elisabeth Hendrickson) "More Testing, Worse Quality» на сайте www.qualitytree.com вы найдете учебную ситуацию, посвященную "привычке тестировать". См. также статью Ли Коплэнда "When Helping Doesn't Help» на www.stickyminds.com.

сительно того, что группа тестирования в действительности в состоянии сделать. Убедитесь в том, что группа разработки знает о необходимости продолжать нести ответственность за структурное тестирование. Последовательно требуйте от руководства проектом выполнения своей части обязательств по поводу ответственности за этапы модульного или компонентного тестирования (в соответствии со сроками, планом и адекватными ресурсами), которые ясно оговорены и активно отслеживаются по графикам проекта. И, наконец, используйте приемочное тестирование и вводите входные критерии для измерения и определения уровней качества системы, необходимых для перехода к комплексному и системному тестированию.

Постепенное совершенствование процесса

Возможно, сейчас вы думаете: "Ну, знаете, наши процессы достаточно хороши. Я не желаю делать какие-либо коренные преобразования. Я лишь хочу, чтобы мы делали свою работу чуть лучше". С этой целью позвольте мне предложить несколько подходов общего назначения, применяемых мной для оптимизации процессов тестирования. Эти подходы относятся к одной из двух категорий. Первая — точная регулировка больших процессов. Второй — выявление и попытка устранения препятствий в больших процессах.

Точная регулировка больших процессов

Прежде всего, начните с изучения тех процессов, которые занимают большую часть времени, сил или ресурсов. Вот простой метод оптимизации процессов, который я узнал, изучая тотальное управление качеством. Я применял этот подход в тех компаниях, где имелся работающий процесс тестирования, но была потребность в его совершенствовании[1].

Начните с рассмотрения двенадцати ключевых процессов тестирования. Выберите два или три из них, те, что занимают у группы тестирования две трети или более времени, сил или ресурсов. К примеру, неудивительно, если вы выясните, что составление отчетов об ошибках и процесс проведения тестов составляют две трети от работ группы тестирования.

Для этих ключевых процессов преобразуйте списки контрольных вопросов, представленные в данной книге, таким образом, чтобы они подходили для ваших текущих потребностей и давали возможность собирать информацию о продолжительности, затратах ресурсов, и, если можно, о стоимости каждого этапа. (Возможно, у вас уже имеются какие-то данные по итогам выполнения тестовых сценариев, если вы начали применять таблицы отслеживания тестовых сценариев, представленные в главе 13.) По мере того как вы и ваша команда будете заниматься каждым

---

[1] Это обучение являлось частью сертификационной программы по тотальному управлению качеством в Калифорнийском университете в Лос- Анжелесе. С точки зрения постепенного совершенствования процесса, самыми полезными были занятия, проводимые корпоративной службой обучения компании Xerox "*Лидерство благодаря качеству*", в особенности "Теоретические основы качества", "Процесс улучшения качества" и "Процесс решения проблем".

процессом, собирайте данные о продолжительности, затратах ресурсов и стоимости каждого этапа работ. Для мелких процессов длительность и затраты следует собирать с точностью до минут.

К процессу сбора этих данных применимы все обычные предостережения относительно использования измерений. Если у работников компании сложится впечатление, что руководство станет использовать данную информацию против них, они исказят данные таким образом, чтобы создавалось благоприятное впечатление об их работе. Если сотрудники решат, что эти данные нужны руководству в качестве показателей для ежегодной аттестации, они будут подтасовывать информацию. Если ваши коллеги сочтут, что руководство будет применять эти данные, чтобы сравнить одних специалистов с другими, ранжируя их по эффективности, может даже начаться взаимная подрывная деятельность, ведущая к разрушению сплоченности команды. Вам вряд ли захочется во имя совершенствования процессов разрушать свою команду.

После того как у вас наберутся данные об исполнении процесса каждым членом группы тестирования примерно в десятке случаев, проанализируйте их. Каковы средние величины по каждому этапу и для всего процесса в целом? Каковы колебания для каждого этапа и для всего процесса? Классическая теория совершенствования качества гласит, что для улучшения производительности процесса нужно снизить колебания вокруг средней величины, а затем изменить ее в желаемом направлении. Однако в отношении процесса тестирования эту аксиому нужно применять с осторожностью, поскольку "быстрее" не обязательно означает "лучше", и для широкой амплитуды колебаний могут существовать вполне разумные причины.

В ходе анализа полученных данных разделите имеющуюся информацию по показательным важным факторам. Рассмотрите данные как по каждому тестировщику, так и по всей команде. Каков уровень индивидуальных колебаний? В случае с отчетами об ошибках изучите данные об ошибках разных уровней приоритетности и серьезности, поскольку предполагается, что на составление отчетов о более важных проблемах требуется больше времени. При исследовании выполнения тестов контроль колебаний зависит от количества ошибок, обнаруженных при помощи теста, а также от времени настройки и удаления теста. Как изменяются данные для автоматизированного тестирования по сравнению с данными, полученными для тестирования вручную? В статистический анализ данных по процессу очень легко могут вкрасться ошибки, если вы смешаете данные по процессам, которые по веским причинам проходят по- разному.

Проведя данный анализ, займитесь поиском основных причин. У кого-то из вашей команды трудности с локализацией ошибок? Возможно, ему нужно пройти курс обучения. У всех членов группы проблемы с локализацией? Может быть, у вас нет достаточного количества альтернативных конфигураций системы, и следует собрать библиотеку образов дисков для ускорения изменения конфигураций. Каждый понедельник после установки новой сборки процесс предоставления отчетов об ошибках замедляется? Возможно, чрезмерно высокая насыщенность ошибками каждой

новой сборки приводит к путанице по поводу того, кто какую ошибку отслеживает.

Может статься, что для проверки ваших гипотез относительно основных причин задержек вам придется собрать дополнительные данные, однако избегайте ситуации, когда анализ парализует ваши дальнейшие действия. Если интуиция подсказывает вам, что можно внести простые изменения, которые имеют низкий риск (т.е вероятность того, что они срикошетят и еще более усугубят ситуацию, крайне мала), то можно сделать это без сбора дополнительной информации. С другой стороны, рискованные и дорогие преобразования требуют дополнительного анализа. Если вы хотите приобрести аппаратное обеспечение на $100 000 для того, чтобы быстрее проводить локализацию ошибок, необходимо собрать информацию, свидетельствующую о том, что нехватка аппаратного обеспечения является основной причиной продолжительной локализации дефектов, и разработать обстоятельный бизнес-план, окупающий инвестиции.

Внеся изменения в процесс, время от времени рассматривайте затраты ресурсов, чтобы со временем перейти к их сокращению. Сразу процесс не улучшится. Собирайте данные и наблюдайте за тем, как протекает совершенствование процесса. Если улучшения ситуации не происходит, прежде чем переходить к дальнейшей оптимизации, выясните, почему ваши преобразования не оказали желаемого влияния. Если же улучшения имели место, дождитесь, когда этот новый процесс стабилизируется, и только после этого переходите к дальнейшей оптимизации. Повторяйте процесс оптимизации снова и снова.

Не все процессы, на выполнение которых требуется больше времени, усилий или затрат, непременно самые худшие. В случае с отчетами об ошибках может оказаться, что один из ваших сотрудников всегда тратит много времени на их составление, однако его отчеты непременно самые лучшие. Вы вряд ли захотите оптимизировать время, затраты сил и ресурсов, но при этом игнорировать основное предназначение процесса. Вводите лишь те усовершенствования, которые позволяют повысить производительность, не понижая эффективность вашей команды. На самом деле, улучшения следует организовывать таким образом, чтобы они, по возможности, оптимизировали и эффективность. Совершенствуя процесс, продолжаете его проверять на предмет качества результатов, используя разделы данной книги, посвященные построению успешных процессов.

Выявление и устранение крупных препятствий для процессов

Распространенной причиной неэффективности процесса являются бесполезная трата ресурсов и переделывание работы. Методика оптимизации процесса, применяемая мной, заключается в отслеживании распространенных предпосылок для бесполезной траты времени, сил и других ресурсов. Пусть сотрудники вашей группы каждую неделю докладывают вам о любых случаях, в результате которых потеряно более одного часа прописанного в графике времени, одного часа затрат (личное время) или ресурсов на сумму свыше $100, если подобных трат можно было избежать. (Если вас не устраивают эти пороговые величины, выберите другие, бо-

лее подходящие в ваших условиях.) В качестве примеров каждого из видов потерь можно привести: находящийся в режиме оффлайн сервер, который используется для обычного сопровождения в период, отведенный по графику на тестирование; прогон тестов по старой тестовой версии; ланч для группы тестирования в пятницу по причине того, что она не может продолжать работу из-за проблем с материально-техническим обеспечением соответственно.

После того как вы соберете эти данные, разбейте их на категории и постройте таблицы для сбора дальнейшей информации. Целью подобных мероприятий является уже не сбор статистических данных по поводу самих процессов тестирования (мы только что проделали это для точной регулировки процессов, см. выше), а выявление помех в процессе тестирования. События, отнесенные вами к напрасной трате времени, не должны включаться в перечень задач, отражаемых в списках контрольных вопросов по процессам. Эти события — неожиданные препятствия, которые мешают вашей группе тестирования работать эффективно и производительно.

Еще раз повторяю, не забывайте о распространенном отношении к количественным показателям. Если тестировщики считают, что вы используете информацию против них, они исказят данные в свою пользу. Если люди, не работающие в вашей компании, решат, что вы собираете эти данные в качестве оружия против них, возникнут огромные трудности. Руководство ненавидит напрасное использование ресурсов и переделывание работы, поэтому имеющаяся у вас информация потенциально взрывоопасна, если попадет не в те руки. Во многих компаниях вам придется быть очень осмотрительным на предмет того, кто будет в курсе, что вы собираете подобную информацию.

Собирая данные, вы со временем столкнетесь с действием правила 80/20 или принципа Парето. Важными будут два или три источника напрасной траты ресурсов, которые обуславливают две трети или более подобных случаев. Остальные будут менее значимыми. В отношении каждого важного источника бесполезного расходования ресурсов составьте план по улучшению процесса, направленный на устранение ненужных затрат. Например, если постоянным источником проблем являются плохие тестовые версии, перечитайте главу 12 и затем введите для решения данного вопроса, скажем, приемочное тестирование[1].

Разрабатывая план улучшения процесса, продолжайте собирать данные. Возможно, вы увидите, что в действительности, прежде чем ситуация начнет улучшаться, проблем становится еще больше (изменение процесса означает непроизводственные издержки и способно смутить вас). Не предпринимайте сразу ответных действий. По прошествии некоторого времени (нескольких месяцев или нескольких проектов) вы увидите, что новый процесс стал привычным для вашей группы. Когда это произойдет, напрасная трата времени, связанная с целевым источником (или источниками) должна снизиться. Как только источники подобной траты стабилизируют-

---

[1] Отличным способом сбора подобной информации может служить база данных управления изменениями, описанная в главе 6 книги "*Managing the Testing Process, Second Edition*" и включенная вместе с другими шаблонами в приложение к этой книге.

Процесс боевых искусств для улучшения процесса
Представлен Директором разработки процессов Филиппом Крюштеном, Rational Software

Шу Ха Ри. Эти три японских иероглифа в некоторых переводах означают "держать, сделать паузу, отпустить", в других — "подражать, уклоняться, разделять". Они также являются принципом айкидо, будо и других японских боевых искусств, относящимся к этапам обучения. Представляется разумным применить этот принцип к знаниям о преобразовании процесса.

Шу. Во-первых, подражайте или просто следуйте основным правилам до тех пор, пока полностью их не усвоите.

Ха. Во-вторых, изучите, когда и каким образом следует делать исключения из этих правил.

Ри. В-третьих, выйдите за пределы основных правил и изобретите свои собственные.

Если вы проигнорируете *Шу* и слишком быстро приступите к *Ха*, то получится путаница, поскольку основные правила и принципы еще толком не поняты. Подобная ситуация возникает, если попробовать чересчур быстро перейти от *Ха* к *Ри*, так как, возможно, в результате вы изобретете колесо. Если лучшие из уже имеющихся методов идеально вам подходят, то, придумывая собственные, вы можете потратить впустую много времени.

Применим этот принцип к знаниям о преобразовании процесса. *Шу* — следуйте простому процессу. Попытайтесь понять и усвоить его основные идеи и найти свой подход к данному процессу. *Ха* — преобразуйте процесс под свои конкретные нужды. *Ри* — изобретите собственный процесс для тех случаев, для которых существующие и даже преобразованные вами процессы представляются неудовлетворительными.

ся, вновь приступайте к сбору данных для улучшения процесса. Каковы теперь два или три источника двух третей напрасной траты ресурсов? Пора составлять новый план по улучшению процесса.

Внедрение усовершенствований

Итак, возможно, вы хотите внедрить крупные изменения в процесс тестирования в вашей компании. Каковы идеи по поводу серьезных преобразований процесса тестирования?

1. Выявите пробелы в ваших текущих процессах тестирования. Эта книга и, например, книги "*Test Process Improvement*" Коомена и Пола и "*The Testing Practitioner*" ван Веенендааля должны обогатить вас идеями. Продолжайте концентрировать внимание на процессах тестирования, находящихся в вашей компетенции, поскольку вы действительно можете изменить их. Используйте такие источники данных, как системы по отслеживанию ошибок, системы сетевой поддержки, данные об отказах в период эксплуатации и результаты проведен-

ных ранее тестов, но будьте также готовы прислушаться к более субъективным, но обоснованным интуитивным соображениям.

2. Выявите преимущества и недостатки изменения процесса для каждого слабого места. Является ли потребность в преобразованиях непосредственной и срочной или долгосрочной и существенной? Каковы риски и затраты, связанные с текущим стилем работы? В чем заключаются возможности и преимущества изменения процесса? Существуют ли личные, политические или практические причины для преобразований?
3. Выберите идеальные изменения процесса, к которым следует приступить. Делая выбор, не забывайте, что на него могут оказать влияние личные, политические и практические воздействия. Вам потребуется дополнить эти воздействия четким бизнес-планом.
4. Планируйте изменения. Выявите заинтересованные стороны, на которых отразятся изменения. Некоторые заинтересованные стороны могут ратовать за сохранение имеющихся процессов, значит, вам придется каким-то образом убедить их. Внедрение ограниченных изменений процесса как элемента отдельно взятого проекта обычно менее рискованно в политическом плане и при этом мудро с точки зрения содержания, а следовательно, это спокойнее воспринимается. Изменить внутренние процессы проще, чем процессы, в которые вовлечено много сторон. Менее контекстно-зависимые процессы менять легче, чем более контекстно-зависимые.
5. Убедите коллег в разумности вашего плана по изменениям процесса. Поймите сами и разъясните другим, каковы риски запланированных изменений и в чем заключаются их преимущества.
6. Приступайте к изменениям, тщательно фиксируйте преимущества, как только они начнут проявляться. Будучи менеджером, выработайте соответствующие изменения в своем поведении для поддержки изменений, внесенных в процесс[1]. Если вносимые изменения существенны, ситуация может сначала ухудшиться. Как и в случае с тестированием, расходы на изменение процесса увеличиваются, прежде чем появляются преимущества, потому важно возобновить аргументацию причин, вызвавших необходимость преобразований, и напомнить о конечной выгоде.
7. Закрепляйте изменения. Порой происходят изменения к лучшему, ситуация исправляется, а затем компания возвращается к старому, дисфункциональному поведению. Важно продолжать работать в выбранном направлении. Добейтесь того, чтобы общая культура группы тестирования или организации работ по созданию программного продукта изменилась таким образом, чтобы новый процесс стал привычным.

---

[1] Более подробную информацию о трудностях и препятствиях, которые руководство в состоянии создать на пути внедрения изменений в процесс, можно найти в статье Джоанны Ротман "Manager Heal Thyself" на www.jrothman.com.

8. Вернитесь к первому этапу и проделайте все сначала. Компании, добившиеся успеха на ниве совершенствования, имеют обыкновение превращать это занятие в привычку. Всегда можно что-то сделать лучше. Всегда найдется то, что можно усовершенствовать. Именно эта привычка, заключающаяся в постоянном улучшении процессов, и отличает действительно выдающиеся компании по разработке программного обеспечения, наиболее зрелые организации, которые являются лидерами в своей области.

Нередко, чем крупнее преобразования, тем медленнее происходит прогресс в изменении процесса. Для внедрения нового способа составления отчетов об ошибках в группе тестирования, состоящей из десяти человек, тест-менеджеру может потребоваться квартал. Часто получение преимуществ начинается с момента введения изменения. Вице-президент по разработке программного обеспечения, внедряющий формальный процесс тестирования, с самого начала может рассчитывать на то, что успех придет через год работы над этим изменением. В силу того, что многие компании, специализирующиеся на тестировании, оказываются не у дел в течение первых двух лет со дня основания, важно, чтобы изменения, проводящиеся в это время, были осторожными, тщательно обдуманными и направленными на получение максимальной пользы.

Джамал Браун оглядывается назад и смотрит вперед

31 марта после обеда Джамал сидел у себя в офисе за закрытыми дверями и размышлял над успехами и неудачами проекта "Суматра". Заключительное совещание прошло 28 марта, всего лишь с недельным отставанием. Задержка была вызвана необходимостью завершить один заключительный полный раунд тестирования после получения в каком-то смысле рискованной тестовой версии, основная ошибка в которой была устранена 14 марта. Кандидат в золотую сборку благополучно прошел тестирование. По крайней мере, существенных ошибок не было выявлено. Поэтому в пятницу вечером было решено выпустить систему. На этом совещании Джамал представил последний отчет о тестировании, дав общее заключение, в котором говорилось, что как известные, так и неизвестные риски качества невысоки и более не могут служить основанием для задержки выпуска версии. Сразу по окончании данного собрания Джамал, с разрешения менеджера, уехал за город на выходные, вернувшись на работу в понедельник в 11 часов утра.

Следующей должна была стать работа над внедрением возможностей поставщика услуг по аренде приложений (ASP) в SpeedyWriter 3.1 — так теперь официально называли Суматру. Но прежде всего Джамал должен был подготовиться к собранию, посвященному подведению итогов по проекту "Суматра". Собрание было намечено на утро следующего дня.

Джамал готовил свою часть тезисов по итогам проекта. Он на минуту задумался, позвонил кому-то из рабочей группы проекта и поговорил с ним, с минуту набирал текст, открыл книгу, внимательно прочитал элек-

тронные письма по проекту, взглянул на отчет о ходе тестирования и затем повторил эти действия. Несколько часов спустя все было готово. Он распечатал документ для всех участников совещания и пошел домой, оставив на рабочем столе аккуратную стопку копий документа, содержащего тезисы. Выключая свет, Джамал взглянул на свой стол, улыбнулся и подумал: "В этот раз мы почти все сделали правильно".

№ этапа	Этап	Выполнено?
4.	Совершенствование: предоставление информации о проекте и его улучшение.	☑
4.A	Документировать ошибки, обнаруженные в ходе выполнения тестов.	☑
4.B	Обсудить результаты тестирования с ключевыми заинтересованными лицами.	☑
4.C	Управлять изменениями и уточнить процесс тестирования	☑

Итоги проекта "Суматра
Отчет группы тестирования

Что протекало успешно

В этом проекте столько всего шло хорошо, что трудно выбрать, но полагаю, наибольший вклад в успех проекта внесли следующие элементы.

Работа в команде. Группа тестирования и рабочая группа проекта работали хорошо. Оценка индивидуального вклада варьируется от хорошей до отличной. Сплоченность команды была на очень высоком уровне. Уровень конфликтности минимален. Единый взгляд на проект и стремление к его успешному осуществлению наблюдалось на протяжении всего срока работ, вплоть до самого конца. Предпосылки успеха:

- Грамотное управление командой, включающее минимизацию сверхурочной работы, чрезмерного давления и непродуктивных совещаний
- Реалистичные ожидания со стороны высшего руководства, менеджеров и команды
- Неизменность целей руководства

Стратегия тестирования на основе рисков. Метод анализа рисков качества — анализ видов ошибок и их влияния — двигал работы по тестированию в наиболее значимых направлениях. Предпосылки успеха:

- Участие всех заинтересованных сторон с квалификацией в разных областях
- Реалистичные ожидания со стороны заинтересованных сторон
- Неизменность цели в том, что касается следования совокупности приоритетов, определенных при анализе риска качества

Отчеты о результатах тестирования. Результаты работы группы тестирования рассматривались как своевременная, важная и надежная информация. Очевидно, что существенное снижение расходов на внешние отказы, следующие за выпуском версии, достигнуто благодаря высокой доле устраненных ошибок. Предпосылки успеха:

- Своевременное достижение консенсуса по поводу инструментальной панели тестирования и ее понимания
- Командный дух во взаимоотношениях тестировщиков и программистов, в результате чего отчеты об ошибках никогда не рассматривались как враждебные выпады
- Группа тестирования сосредоточила силы на предоставлении высококачественной информации

Что было неудачным

Проблем было мало и, по моему мнению, они в большей части относились к таким ситуациям, с которыми практически всегда возникают сложности. Как бы то ни было, к замеченным возможностям по совершенствованию процесса относятся следующие.

Разбор ошибок. Группа управления изменениями отложила слишком много ошибок, и эти ошибки пришлось открыть заново после вмешательства высшего руководства. Но открытие ошибок заново является повторным выполнением работы (а следовательно, элементом неэффективной работы) и увеличивает риск, связанный с сокращением времени на разрешение других проблем. Возможности по исправлению ситуации:

- Возможно, имеет место недостаточный надзор со стороны руководства за группой по управлению изменениями. Увеличить долю участия высшего руководства.
- Имеющийся в настоящее время процесс не в состоянии эффективно эскалировать дефекты "переходного характера", которые, возможно, с точки зрения руководства относятся к подлежащим обязательному устранению в выпускаемой версии.

Кадровые вопросы. Нам очень повезло, что в последний момент удалось найти устраивающего нас специалиста по автоматизации тестирования. Если бы этого не произошло, проведенное нами тестирование было бы значительно менее эффективным, особенно для первых нескольких инкрементов. Возможности по улучшению ситуации:

- В настоящее время отношения компании с контрактниками строятся в основном, исходя из стоимости рабочей силы, и ведутся с компаниями, специализирующимися исключительно на экспертизе в программировании. Необходимо определить возможных поставщиков услуг по аутсорсингу высококвалифицированных специалистов и подбору контрактников-тестировщиков, выстроить стратегические отношения с ними, чтобы заполнять внезапно возникающие вакантные позиции.
- Навыки членов группы тестирования слишком узко специализированы, что опасно при необходимости взаимозаменяемости. Увеличить упор на обучение и создать гармоничную группу тестирования, что подразумевает рост расходов на обучение, заложенный в бюджет.

Планы на будущее

Наряду с работой над вопросами, указанными выше в качестве возможностей по улучшению ситуации, мною планируется следующее:

Оптимизация процессов. Я проведу анализ измерений процесса тестирования в проекте "Суматра" и направлю силы на поиск возможностей по совершенствованию процесса с целью повышения эффективности и производительности. Кроме того, чтобы ответить на вопросы, связанные с эффективностью и производительностью, которые не в состоянии разрешить уже имеющиеся показатели, я продумаю шаги по усовершенствованию процесса сбора измерений в предстоящих проектах.

Устранение препятствий. При работе над последующими проектами я введу новые методы сбора данных, которые помогут отследить и структурировать повторно проводимые работы, блокирование работ, время простоя и прочие препятствия на пути к эффективной работе процесса.

Снижение затрат в расчете на дефект, подлежащий обязательному устранению. Наши затраты в расчете на дефект, подлежащий обязательному устранению, составляют 500$. Эта сумма остается неизменной на протяжении последних трех проектов. Однако большая часть бюджета на тестирование отводится на ручное повторное проведение тестов в рамках регрессионного тестирования, которое является не слишком эффективным. Если при тестировании вручную мы сосредоточимся на проведении новых тестов и выполнении исследовательского тестирования, то средние расходы на дефект, подлежащий обязательному устранению, снизятся, поскольку при поиске ошибок, которые должны быть непременно исправлены, подобные тесты, как правило, дают лучший результат, чем регрессионные. В виду того, что ключевым моментом является управление риском регрессии, нам следует значительно увеличить долю автоматизации регрессионного тестирования, а также автоматизировать этот процесс на ранних фазах жизненного цикла. Для решения этого вопроса я предлагаю создать оперативную группу, включающую специалистов по комплексному и системному тестированию, сборке версий, разработке новой версии, по поддержке системы и системной архитектуре. Предлагаю поставить перед этой группой задачу по планированию и внедрению полностью автоматизированной и интегрированной оболочки для регрессионного тестирования, работающей как на уровне пользовательского интерфейса, так и на уровне слоев бизнес-логики архитектуры системы.

Успех в этих трех направлениях приложения сил способен, по моему мнению, увеличить окупаемость инвестиций в тестирование не менее чем на 50%, а также сократить время циклов тестирования и повысить процент ошибок, обнаруженных до передачи версии заказчику.

Заключение

В этой книге содержатся идеи по совершенствованию процессов тестирования. Представленные процессы описывают мой собственный подход к тестированию и управлению им. Учебная ситуация является живым

примером работы данных методов. Изучая документы обучающего примера, вы получите примеры и шаблоны, которые могут облегчить процесс "перекройки" моих методов под ваши потребности, если у вас возникнет желание применить полученные знания. И, наконец, разделы, посвященные построению успешных процессов, решению проблем и внедрению усовершенствований, призваны помочь вам понять, как продвигаться в этих направлениях. Упорное, продуманное совершенствование двенадцати ключевых процессов тестирования поможет вам эффективно и качественно готовить и передавать информацию по управлению рисками качества и оказывать услуги в рамках выстроенных вами процессов тестирования.

Тем не менее, позвольте мне закончить книгу предостережением и сделать признание, что сам по себе процесс не может творить чудеса. Прежде чем профессиональный сленг управления качеством привел нас к беседе о процессах, мы рассмотрели способы решения проблем как *системы*. Наличие систематического подхода при выполнении некоторой работы означает, что имеется зафиксированный и повторяемый процесс. Русский классик Иван Тургенев писал: "Системами дорожат только те, которым вся правда в руки не даётся, которые хотят её за хвост поймать; система — точно хвост правды, но правда — как ящерица; оставит хвост в руке, а сама убежит: она знает, что у нее в скором времени другой вырастет".

Иван Тургенев прав, поскольку процесс может оказаться сдерживающим. И если им пользоваться неверно, так и происходит. Однако если вы направите свои силы на то, что оказывает в конкретной ситуации существенное влияние на эффективность и качество тестирования, процесс может стать полезным руководством к действию. Грамотный процесс окажет вам услугу, освободив от необходимости каждый раз вновь придумывать, как добиться успеха. Процесс может выступить в роли контрольного списка вопросов и направить вашу творческую энергию на решение более существенных проблем, нежели просто заботиться о том, чтобы ничего не упустить из виду. Удачи вам в работе с ключевыми процессами тестирования и всеми другими важными составляющими успеха в тестировании!

ГЛОССАРИЙ

Приемо-сдаточные испытания, пользовательское приемочное тестирование (Acceptance Testing, User Acceptance Testing). Фаза тестирования, целью которой является проверка того, что система удовлетворяет требованиям. Название "пользовательское приемочное тестирование" может указывать на то, что тестирование проводится силами пользователя.

Активность, действие (Activity). Логически выделенная часть фазы, состоящая из набора задач, выполнение которых означает получение одного существенного передаваемого результата.

Альфа-тестирование, бета-тестирование (Alpha Testing, Beta Testing). Фазы тестирования, в ходе которых система подвергается проверке со стороны внутренних (альфа) или внешних (бета) заказчиков и пользователей в реальных условиях, на настоящих данных и в реальной среде.

Амортизация (Amortize). Распределение затрат и прибыли от инвестиций на серии временных периодов (например, ежемесячно) либо на совокупность проектов. Показывает средние затраты и среднюю прибыль от инвестиций на период времени или на проект.

Навыки в предметной области приложения, экспертиза в предметной области (Application Domain Skills, Subject Matter Expertise). Навыки или знания, относящиеся к какой-либо области или к решаемым задачам в программном или аппаратном обеспечении; другими словами, понимание того, как заказчики и пользователи будут применять создаваемую систему.

Выполнение контрольных заданий (Audition Interview). Часть процесса интервьюирования при приеме на работу, которая посвящена показу умений и навыков, необходимых для занятия вакансии, например написанию тестового сценария или подготовке отчета об ошибке.

Система сбалансированных показателей (Balanced Scorecard). См. *Инструментальная панель* (Dashboard)

Тестирование поведения системы, тестирование методами "черного ящика" (Behavioral Tests, Black-Box Tests). Тестирование на основе внешнего поведения системы, часто апеллирующее к требованиям и спецификациям архитектуры высокого уровня.

Бета-тестирование (Beta Testing). См. *Альфа-тестирование* (Alpha Testing)

Тестирование методами "черного ящика". См. *Тестирование поведения системы* (Behavioral Tests)

Ошибка, дефект (Bug, Defect). Проблема, которая вызывает или может вызвать неадекватное выполнение системой одного или нескольких обоснованных ожиданий качества заказчика/пользователя.

Отчет об ошибке (Bug Report). Технический документ, описывающий симптомы ошибки для 1) информирования о влиянии и обстоятельствах возникновения проблемы; 2) назначения ошибке приоритета для исправления; 3) помощи программисту в локализации дефекта и его исправления.

Сборка, версия, тестовая версия (Build, Release, Test Release). Устанавливаемый компонент аппаратно-программного обеспечения или системы либо система целиком, передаваемая группе тестировщиков для тестирования.

Номер сборки, имя сборки, версия, номер версии (Build ID, Release Name, Revision, Revision Number, Version, Version Number). Уникальная последовательность цифр и/или букв, однозначно определяющая сборку.

Build Manager. См. *Инженер сборки* (Release Engineer)

План сборок (Build Plan, Release Engineering Plan). План, определяющий, каким образом будут строиться сборки для тестирования в реальных условиях.

Кандидат (Candidate). Человек, подавший заявление на занятие вакансии и находящийся на рассмотрении, но еще не принятый официально.

Кодируй и исправляй (Code and Fix). Подход к разработке, исключающий большинство действий по планированию, сбору требований и проектированию, начинающийся непосредственно с кодирования системы с последующим исправлением обнаруженных ошибок. Обнаружение ошибок ведется посредством формального тестирования независимой группой тестировщиков, либо при помощи неформального (ad hoc) тестирования, либо только при отладке. Данная модель может быть удобна для небольших, несложных проектов разработки или поддержки с невысоким уровнем рисков. Модель не подходит для больших, сложных проектов и проектов с высоким уровнем рисков.

Компонентное тестирование, тестирование подсистем (Component Testing, Subsystem Testing). Фаза тестирования, ориентированная на проверку компонентов системы и/или ее подсистем по отдельности.

Критическая сеть (Critical Network). См. *Критический путь* (Critical Path).

Критический путь, критическая сеть (Critical Path, Critical Network). Совокупность задач (в терминах длительности действий и задач) в структурной декомпозиции проектных работ (с учетом всех межзадачных взаимозависимостей), которые должны завершиться к определенному моменту для того, чтобы проект был завершен в срок.

Заказчики, спонсоры (Customers, Sponsors). Люди, финансирующие создание системы любым из следующих способов: непосредственная оплата разработки системы, участие в финансировании системы, финансовая поддержка производителя системы, приобретение продукта или службы, содержащей или использующей систему, оплата налогов, прочие прямые и косвенные способы финансирования. Заказчик — это организация, приобретающая и/или поставляющая систему для пользователей, являющихся сотрудниками организации, или частное лицо, приобретающее систему для личного пользования.

Инструментальная панель, система сбалансированных показателей (Dashboard, Balanced Scorecard). Это набор метрик, которые обычно представлены в виде небольшого числа диаграмм или графиков. Они дают представление о текущем статусе некоторой деятельности. Аналогией здесь является приборная доска автомобиля, на которой набор из двух-трех измерительных устройств и предупреждающие светодиоды обеспечивают водителя важной информацией о состоянии транспортного средства, не отвлекая его от наблюдения за дорогой и движением. Термин "система сбалансированных показателей" означает тот факт,

что при удачном подборе диаграммы или графики такого рода дополняют друг друга. Иными словами, любое, отдельно взятое измерение может склонить к неверным выводам, тогда как кумулятивный эффект всех метрик должен образовать контекст для других измерений.

Дефект (Defect). См. *Ошибка* (Bug)

План разработки (Development Plan). План, определяющий, каким образом будут создаваться компоненты, составляющие систему, включая последовательность разработки компонентов.

Оценка (Estimation). Прогноз графика работ и затрат в проекте. В случае оценки тестирования это прогноз графика работ и затрат подпроекта тестирования.

Ожидание качества (Expectation of Quality). Убежденность заказчика/пользователя в уровне качества, который должна обеспечивать система. В идеале заказчики и пользователи должны иметь обоснованные ожидания качества системы.

Необоснованные ожидания обычно вызывают остановку таких действий, как сбор требований, бизнес-анализ и управление изменениями. Часто невозможно и, как правило, нежелательно пытаться следовать необоснованным ожиданиям качества при проведении тестирования.

Опыт качества (Experience of Quality). Мнения о качестве системы вместе с общим уровнем удовлетворенности пользователей/заказчиков по мере получения опыта работы с системой. Когда уровень опыта качества совпадает с ожиданием качества, пользователь/заказчик обычно удовлетворен. Если опыт качества превышает ожидания, пользователь/заказчик восхищается. Если опыт качества ниже ожиданий, пользователь/заказчик разочарован.

Дополнительные смены (Extended Shifts). Любая работа, выполняемая сотрудниками вне основного рабочего времени (примерно с 8.00 до 17.00). Обычно это *вечерняя смена* (примерно с 16.00 до 1.00), *ночная смена* (примерно с 0.00 до 9.00), *работа в выходные и праздничные дни* (любая из указанных смен, но не в рабочие дни). Сравните с *переработкой* (overtime) — работа свыше 40 часов в неделю, может включать в себя работу в любые смены.

Вид ошибки (Failure Mode). Определенный способ возникновения или тип ошибки в системе, т.е. категория ошибок или сбоев с определенными отличительными общими признаками. Одному виду ошибки могут соответствовать ноль, одна и более отдельно взятых ошибок или неисправностей. При рассмотрении вида ошибки в любом варианте анализа рисков качества необходимо тщательно выверять степень детализации категорий видов ошибки, т.е. расширенные категории, состоящие из большого числа ошибок, или узкие категории, состоящие лишь из нескольких ошибок.

Полные затраты на штатную единицу (Fully Burdened Staff Rate). Затраты на одну штатную единицу, включая заработную плату, премии, затраты на инфраструктуру и прочие связанные затраты на персонал.

Золотая сборка, мастер-сборка, сборка общего доступа (Golden Build, Gold Master, GA Build). Сборка или версия, поставляемая заказчикам или устанавливаемая пользователям. Тестовую сборку, предшествующую "золотой" сборке, часто называют *"золотым" кандидатом* (Golden Candidate). Если "золотой" кандидат оказывается качественным, он становится *"золотой сборкой"*. (GA означает General Availability (общий доступ), т.е. доступ всем возможным заказчикам и пользователям. Limited Availability (ограниченный доступ) означает бета- или раннюю версию для специальных заказчиков.)

"Золотой" кандидат (Golden Candidate). Плановая финальная тестовая версия, которая при условии, что не будет обнаружено ошибок, требующих исправления, становится версией системы, поставляемой пользователям, т.е. *"золотой" сборкой*.

Отдел по работе с персоналом (Human Resources Department, Personnel Department). Группа, отвечающая за 1) поиск сотрудников, контрактников и консультантов; 2) выработку политики и процедур управления персоналом; 3) планирование и управление изменениями в заработной плате и карьерном росте сотрудников.

Жизненный цикл инкрементной системы разработки (Incremental System Development Lifecycle). Модель разработки, при которой большой проект разбивается на последовательность шагов; в результате каждого из шагов появляется очередная порция функциональности общего перечня требований, зафиксированных в проекте. Требования расставляются в порядке приоритетов и поставляются в указанном порядке по итогам соответствующего этапа наращивания функциональности. В некоторых (не во всех) версиях этой модели разработки каждый подпроект выполняется в соответствии с мини-V-моделью со своими фазами проектирования, кодирования и тестирования. Как бы ни был устроен процесс каждого этапа приращения функциональности, тенденцией является раннее начало тестирования с поставкой первой инкрементной тестовой сборки, и далее тестирование продолжается в полном объеме до момента поставки системы или первой установки у пользователя.

Независимая группа тестирования (Independent Test Team). Группа людей, отличная от разработчиков системы, предназначенная для тестирования; в сферу ее ответственности входят эффективная проверка и своевременное информирование о качестве тестируемой системы. Обычно я рассматриваю эту группу как поставщика услуг для проектной команды.

План интеграции (Integration Plan). План, определяющий, каким образом различные компоненты будут собираться в некоторую логическую многоэтапную последовательность до тех пор, пока не будет получена система целиком.

Комплексное тестирование, тестирование продукта (Integration Testing, Product Testing). Фаза тестирования, в ходе которой проверяются взаимосвязи и интерфейсы между парами и группами компонентов или подсистем системы.

Ключевые лица, заинтересованные в качестве и тестировании, ключевые заинтересованные лица (Key Quality and Testing Stakeholders, Key Stakeholders). Сотрудники компании, непосредственно контактирующие с заказчиками, пользователями или участвующие в процессе тестирования, а также сотрудники, которые хорошо знакомы с техническими деталями системы.

Библиотека (Library). См. *Репозиторий* (Repository)

Новый сотрудник (New Hire). Лицо, получившее письмо с предложением о занятии вакантного места и ответившее на него устно или письменно согласием занять предложенную вакансию.

Письмо с предложением о занятии вакантного места (Offer Letter). Письмо из компании успешному кандидату с предложением занять вакантное место, включая детальное описание должностных обязанностей, указание фактического размера заработной платы и даты начала работы.

Фаза (Phase). Логически выделенная часть проекта или подпроекта, состоящая из совокупности действий, которые завершаются выпуском продуктов, поставляемых заказчику.

Пилотное тестирование (Pilot Testing). Фаза тестирования, на которой система подвергается опытной эксплуатации под тщательным наблюдением. Пилотные тесты аппаратной части показывают возможности сборочной линии по массовому производству системы. Пилотные тесты программной части показывают возможности системы по обработке типичных операций реальных пользователей на реальной аппаратуре.

Тестирование продукта (Product Testing). См. *Комплексное тестирование* (Integration Testing)

Проект (Project). Временное предприятие, целью которого является создание или поставка некоторого уникального продукта, системы или услуги. Это предприятие состоит из некоторой последовательности задач, выполняемых группой людей при определенных ресурсных и временных ограничениях для передачи продукта, системы или услуги некоторому множеству заказчиков или пользователей. Диапазон, в котором в течение проекта могут изменяться группа, бюджет, график, требования и целевые заказчики и пользователи, может существенно отличаться.

Качество (Quality). 1. Пригодность для использования. Наличие свойств, атрибутов и функций, удовлетворяющих ожиданиям заказчиков и пользователей, и отсутствие свойств, атрибутов и функций, неудовлетворяющих ожиданиям заказчиков и пользователей.

2. Соответствие требованиям. Наличие свойств, атрибутов и функций, удовлетворяющих всем установленным требованиям, и отсутствие свойств, атрибутов и функций, отклоняющихся от требований. (Данное определение применимо в конкретном контексте, в особенности для проектов, выполняемых по контракту.)

Оценка качества (в отличие от тестирования) (Quality Assurance contrasted with Testing). В соответствии со стандартом IEEE 610.12-1990 стандартный глоссарий терминологии по программированию ("IEEE Standard Glossary of Software Engineering Terminology") говорит: оценка качества (QA) включает в себя "всякую деятельность, необходимую для обеспечения достаточной уверенности" в качестве системы и оценку "процесса, с помощью которого [система] разрабатывалась", тогда как тестирование состоит из действий, необходимых для "обнаружения различий между существующими и требуемыми условиями (ошибок) и оценки... свойств". Другими словами, оценка качества фокусируется на корректности процесса, а тестирование — на установлении качества.

Риск качества (Quality Risk). Потенциальный вид ошибки, способ поведения системы, при котором она, вероятно, не соответствует обоснованным ожиданиям качества системы, имеющимся у пользователя или заказчика. Заметим, что риск качества — это потенциальный, а не обязательный результат с вероятностью больше нуля и меньше единицы.

Рассматривают категории рисков качества. Поскольку виды ошибок также разбиваются на категории, категории рисков качества — это категории категорий потенциальных дефектов.

Эталонная система (Reference System). Особый тип *тестового оракула*, когда существующая система моделирует некоторые или все ожидаемые варианты поведения системы, предназначенной для тестирования.

Регрессия (Regression). Система обладает свойством регрессии, если в результате изменения новая версия системы S_{n+1} содержит ошибку в ранее корректно работавшей функции, т.е. этой ошибки не было в предыдущих версиях, начиная с S_1 и заканчивая S_n.

Границы регрессионного тестирования (Regression Test Gap). Это разность между тестовым покрытием всего комплекта тестов и тестовым покрытием, которое достигается при помощи реально выполненных тестов для данного изменения или данной версии. Для версии, предоставляемой заказчику, границы регрессионного тестирования могут рассматриваться как область, которая в определенной степени не подвергалась проверке в ходе тестирования данной версии с применением всего комплекта тестов.

Регрессионное тестирование (Regression Testing). Тестирование, предназначенное для поиска дефектов, вызывающих регрессию.

Версия (Release). См. *Сборка* (Build)

Менеджер сборки, инженер сборки (Release Engineer, Build Manager, Source Code Control Engineer, Release Manager). Специалист, отвечающий за создание тестовой версии из исходного кода, данных, метаданных, файлов помощи, web-экранов и прочих объектов, необходимых для сборки системы.

Менеджер сборки (Release Manager). См. *Инженер сборки* (Release Engineer)

Имя версии (Release Name). См. *Идентификатор сборки* (Build ID)

Репозиторий, библиотека, система управления исходным кодом (Repository, Library, Source Code Control System). Программное приложение, обычно работающее на общедоступном сервере. Хранит исходный код, встроенную систему оперативной помощи (online help), документы HTML, аудио- и прочие файлы таким образом, что после компиляции и ассемблирования создается инсталлируемый программный продукт. Кроме того, хранит связанные документы, в частности планы тестирования, анализ рисков, спецификации требований.

Коэффициент окупаемости инвестиций (КОИ) (Return on Investment, ROI). Величина, полученная в результате инвестирования в некоторый проект. Коэффициент окупаемости инвестиций — это финансовая прибыль от инвестиции за вычетом начальных затрат, деленная на начальные затраты:

$$КОИ = (Прибыль - Затраты)/Затраты$$

КОИ обычно умножают на 100% и приводят в процентах.

Редакция (Revision). См. *Идентификатор сборки* (Build ID)

Номер редакции (Revision Number). См. *Идентификатор сборки* (Build ID)

Тестирование на исправность (Sanity Test). См. *Приемочное тестирование* (Smoke Test)

Приемочное тестирование, тестирование на исправность (Smoke Test, Sanity Test). Тестирование, проверяющее, насколько стабильна предлагаемая на тестирование версия, чтобы можно было начинать штатное тестирование. Обычно это подмножество всего комплекта тестов, как правило, автоматизированное, затрагивающее все части системы, по крайней мере, поверхностно. Качественные приемочные тесты обычно довольно долго проверяют работу системы, чтобы проявились серьезные проблемы надежности и работоспособности. Термин "smoke test" (тест на задымленность) взят из электротехники: когда инженер включает цепь, первичный тест проверяет, не дымятся ли компоненты.

Инженер исходного кода (Source Code Control Engineer). См. *Инженер сборки* (Release Engineer)

Система управления исходным кодом (Source Code Control System). См. *Репозиторий* (Repository)

Спонсоры (Sponsors). См. *Заказчики* (Customers)

Заинтересованные (в системе) стороны (Stakeholders (in the system)). Лица как внутри компании, так и вне ее, на которых влияют в настоящем, повлияют или могут повлиять в будущем ход проекта, разрабатываемая система и ее качество.

Структурное тестирование, тестирование методами "белого ящика" (Structural Tests, White-Box Tests). Тестирование, основанное на внутренних особенностях работы системы, обычно апеллирующее к деталям архитектуры, логике и данным.

Заглушка (Stub). Обычно небольшой фрагмент программы, который вставляется в код либо заменяет компонент системы, предназначенный для тестирования,

чтобы аппроксимировать работу ряда заменяемых с его помощью функций для выполнения определенных тестов.

Экспертиза в предметной области (Subject Matter Expertise). См. *Навыки в предметной области приложения* (Application Domain Skills)

Подпроект (Subproject). Проект в рамках основного проекта, предоставляющий продукты и услуги для основного проекта. Согласно этому определению, тестирование, обсуждаемое в этой книге, обычно является подпроектом. Выпуск бухгалтерией чеков для оплаты работы участников проекта не является подпроектом, поскольку это выходит за пределы контекста проекта. Аудит соответствия документированному процессу, проводимый усилиями сотрудников отдела качества, не является ни подпроектом, ни проектом, поскольку он выполняется для нужд компании.

Тестирование подсистем (Subsystem Testing). См. *Компонентное тестирование* (Component Testing)

Система, система, предназначенная для тестирования, разрабатываемая система, система на сопровождении (System, System under Test, System under Development, System Being Maintained). Цельная система (программное обеспечение, аппаратное обеспечение, аппаратно-программное обеспечение и т.д.) с точки зрения тестирования. Часто состоит не только из очевидных компонентов. Например, в случае офисных приложений в дополнение к исполняемым модулям на рабочем месте пользователя могут прибавиться прикладные части, выполняемые на сервере, которые могут иметь разнообразные интерфейсы с продуктами третьих сторон, а также процессы документирования, поставки и обновления, онлайновые базы данных и средства поддержки, компоненты аппаратного обеспечения, встроенные программы и даже инфраструктура, включая LAN, WAN и Интернет. Слово "продукт" до некоторой степени более общий, однако может быть некорректно понят как исключительно прикладная программа, предназначенная для массового рынка.

Система может быть также *системой систем*, где каждая система взаимодействует с другими системами и зависит от одной и более систем, и все системы существенны для работы системы систем. Например, приложение на основе браузера вместе с клиентскими программами, web-сервером, сервером приложений, сервером баз данных есть система систем.

Наконец, система может представлять собой *семейство систем*, где каждая система взаимодействует с одной или несколькими системами, но зависимости ограничены, и система может применяться как самостоятельно, так и в интеграции с другими системами. Например, Microsoft Office — это семейство систем программного обеспечения.

Система на сопровождении (System Being Maintained). См. *Система* (System)

Жизненный цикл разработки системы (System Development Lifecycle). Часть жизненного цикла системы, связанная с начальной разработкой системы, начиная со спецификации и проектирования и заканчивая реализацией, тестированием и первичной поставкой.

Жизненный цикл системы (System Lifecycle). Все множество действий, которые имеют место в ходе разработки, поставки, использования и устаревания системы, начиная с исходной спецификации и проектирования, далее включая реализацию, тестирование, поддержку, сопровождение и заканчивая остановкой и выводом системы из эксплуатации.

Системное тестирование (System Testing). Фаза тестирования, на которой проверяется поведение системы как в целом, так и в частностях, и ее функциональность.

Разрабатываемая система (System under Development). См. *Система* (System)

Система, предназначенная для тестирования (System under Test). См. *Система* (System)

Задача (Task). Рабочие элементы активности, которые являются логически последовательными множествами действий, включающих в себя соответствующие наборы умений и навыков.

Технические умения и навыки (Technical Skills). Способность применять определенные методы и инструменты, которые планируется использовать при создании системы, например, языки программирования или серверную и сетевую архитектуру.

Артефакты тестирования (Test Artifacts). Поведение, которое возникает из-за искусственной природы тестовой среды или из-за того, что процесс тестирования отклоняется от потенциальных действий системы в реальных условиях; неверное поведение или некорректные результаты, полученные с помощью системы тестирования.

Тестовый сценарий (Test Case). Последовательность шагов, состоящих из действий, которые необходимо выполнить при помощи системы, предназначенной для тестирования. (Эти шаги обычно называют *тестовой процедурой* или *тестовым скриптом*.) Такого рода действия обычно увязывают с некоторым набором данных (заранее загруженным или вводимым в процессе выполнения сценария). Комбинация выполняемых действий и данных, передаваемых системе в процессе выполнения теста, приводит систему к некоторому тестовому состоянию. В тестовом состоянии система выдает результаты, которые можно сравнить с ожидаемыми, т.е. оценить качество в данном тестовом состоянии. Такие действия могут выполняться последовательно, параллельно или в различных вариациях.

Тестовое состояние (Test Condition). Состояние системы, выходные данные, результат или условие, порождаемые выполнением тестового сценария, преимущественно интересные с точки зрения оценки качества системы.

Тестовое покрытие (Test Coverage). 1. Структурное: степень покрытия тестами структуры — кода, подсистем или компонентов — в системе, предназначенной для тестирования.

2. Поведенческое: степень покрытия тестами поведения — рисков качества, режимов работы, функций, активностей и прочего применения — системы, предназначенной для тестирования.

Тестовый цикл (Test Cycle). Выполнение некоторого подмножества тестов для одной, определенной тестовой сборки.

Тестовые данные (Test Data). Набор входных данных, который используется в тестовом сценарии. Данные вводятся либо при помощи пользовательского интерфейса, либо из хранимого файла, либо из другой взаимодействующей системы.

Тестировщик, инженер тестирования (Test Engineer). Сотрудник группы тестирования, обученный, обладающий навыками и опытом во всех областях тестирования (планирование, проектирование, реализация и выполнение), а также обладающий определенными навыками в проведении технической экспертизы и знанием прикладной области. (В некоторых штатах и странах лица, чей титул включает в себя слово "инженер", должны иметь профессиональную лицензию (диплом).)

Тестовая среда (Test Environment). Аппаратное и программное обеспечение, оборудование и помещения, необходимые для проведения тестирования.

Пропуск при тестировании (Test Escape). Любая ошибка, которая могла быть обнаружена при тестировании, но была пропущена. Термин часто применяют к ошибкам, обнаруженным на фазах,

следующих за тестированием, хотя они должны были быть найдены на предыдущей фазе.

Процесс выполнения тестов (Test Execution Process). Методы, при помощи которых тестировщики и другие ключевые участники процесса тестирования используют тестовые программы в тестовой среде.

Степень детализации при тестировании (Test Granularity). Точность нацеливания теста. Точные (fine-grained) тесты позволяют тестировщику проверить низкоуровневые компоненты, особенно реализации; статические и структурные тесты являются точными. Грубые (coarse-grained) тесты предоставляют тестировщику информацию об общей работе системы и требованиях; поведенческие тесты и тесты для опытной эксплуатации являются грубыми.

Уровень тестирования (Test Level). См. *Фаза тестирования* (Test Phase).

Инструменты управления тестированием (Test Management Tools). Инструменты, управляющие объектами систем тестирования и их результатами. Классическими примерами являются системы управления дефектами и тестами. Более продвинутые системы предоставляют полномасштабные средства для проектирования, реализации и выполнения тестов. Такие системы обычно интегрированы с инструментами тестирования, хотя иногда эти дополнительные возможности ограничивают полезность инструментария для тестов, поддерживаемых инструментом тестирования.

Тестовый оракул (Test Oracle). Система, метод или методика для предсказания или оценки корректности поведения системы, предназначенной для тестирования, в определенных условиях.

Тестовый раунд (Test Pass). Выполнение тестов, определенных для данной фазы тестирования.

Фаза тестирования, уровень тестирования (Test Phase, Test Level). Выделяемая совокупность действий тестирования, адресованная фиксированной группе рисков качества, например: компонентное тестирование, комплексное тестирование, системное тестирование, приемо-сдаточные испытания. (Я предпочитаю использовать слово "фаза", тогда как другие говорят "уровень".)

Специалист-тестировщик (Test Professional). См. *Тестировщик* (Tester)

Тестовая версия (Test Release). См. *Сборка* (Build)

Тестовый скрипт (Test Script). Описание того, как выполнять один и более тестовых сценариев при проведении тестирования вручную. Часто называют тестовой процедурой. При автоматизированном тестировании говорят о соответствующих последовательностях команд, выполняемых компьютером.

Комплект тестов (Test Set). Все наборы тестов, которые группа тестирования планирует выполнить в течение данной фазы (или уровня) тестирования.

Младший тестировщик (Test Specialist). См. *Лаборант-тестировщик* (Test Technician)

Стратегия тестирования (Test Strategy). Совокупность выбранных подходов и решений, вытекающих из целей и задач подпроекта тестирования и группы тестирования. Целью обычно является эффективное и качественное тестирование, а стратегия — это общие правила и принципы, способствующие достижению цели. Тактика тестирования — это частные правила, методики, процессы и способы проведения тестирования.

Подпроект тестирования (Test Subproject). Совокупность задач, необходимых для предоставления услуг тестирования и информирования о качестве и ходе тестирования заинтересованных лиц проекта, а также содействие проектной команде в управлении рисками качества.

Набор тестов (Test Suite). Совокупность логически связанных тестовых сценариев. Те или иные тестовые сценарии могут содержаться в нескольких наборах тестов.

Лаборант-тестировщик, младший тестировщик (Test Technician, Test Specialist). Специалист по тестированию, менее квалифицированный, менее образованный и менее опытный, чем инженер-тестировщик, хотя бы в одном из критических направлений (тестирование, предметная область или технологии). Лаборанты-тестировщики обычно занимаются выполнением тестовых скриптов вручную и другими очевидными задачами своего уровня. См. также *Тестировщик* (Tester)

Тестировщик, специалист-тестировщик (Tester, Test Professional). Лицо, выполняющее задачи, относящиеся к тестированию. Человек, работающий в группе тестирования постоянно или временно в качестве сотрудника, контрактника или консультанта. Я использую этот термин как аналог понятия "разработчик" или "программист" в группе разработки.

Тестирование (Testing). Функционально это процесс оценки качества системы, предоставления услуг и информации, позволяющий компании управлять рисками качества системы. Организационно это группа специалистов, предоставляющая данные услуги и информационные продукты для проекта, в котором она занята.

Умения и навыки тестирования (Testing Skills). Квалификационные способности, относящиеся к выполнению задач тестирования, например к разработке качественных тестовых сценариев и написанию подробных отчетов о дефектах.

Инструменты тестирования (Testing Tools). Любая система, способствующая выполнению задач тестирования, таких как проектирование, реализация, верификация, выполнение, измерение и интерпретация тестовых сценариев, тестовых данных и результатов тестирования. В качестве инструментов могут выступать таблицы, шаблоны, базы данных, комплексные заказные утилиты собственной разработки и коммерческие системы. Инструменты бывают комбинированные и интегрированные. Некоторые инструменты, в частности системы захвата/воспроизведения и утилиты покрытия кода, разрабатываются сразу как инструменты тестирования, другие — отладчики, языки скриптов — лишь используются в этом качестве.

Средства тестирования (Testware). Тестовые сценарии, тестовые данные, тестовые скрипты, инструменты тестирования, протоколы тестирования, отчеты о тестировании, а также соответствующие руководства пользователя и документация.

Модульное тестирование (Unit Testing). Тестирование небольшого элемента или модуля системы.

Пользовательское приемочное тестирование (User Acceptance Testing). См. *Приемо-сдаточные испытания* (Acceptance Testing)

Пользователи (Users). Люди, которые работали и, возможно, будут работать с системой, а также люди, применяющие результаты работы системы. Иногда пользователи являются заказчиками системы, но не всегда.

V-модель (V Model). Вариация каскадной модели, в которой задачи разработки идут сверху вниз по левой стороне буквы V, а задачи тестирования — вверх по правой стороне буквы V. Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из фаз разработки влияют на развитие системы тестирования на каждой из фаз тестирования. Модель базируется на том, что приемо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование — на требованиях и архитектуре, комплексное тестирование — на требованиях, архитектуре и интерфейсах, а компонентное тестирование — на требованиях, архитектуре, интерфейсах и алгоритмах.

Версия (Version). См. *Идентификатор сборки* (Build ID)

Номер версии (Version Number). См. *Идентификатор сборки* (Build ID)

Жизненный цикл каскадной модели разработки системы (Waterfall System Development Lifecycle). Модель разработки, в которой группа разработки проходит серию последовательных фаз в процессе создания системы. Сначала группа определяет требования, затем проектирует систему и наконец реализует ее в коде. Как только код системы готов, система проходит через последовательность фаз тестирования, начиная с модульного или компонентного тестирования, затем комплексное тестирование и наконец системное тестирование. (В случае контрактной разработки или разработки для внутреннего применения часто после системного тестирования проводятся приемо-сдаточные испытания.) Вариации этого подхода допускают наложение фаз, не требуя завершения каждой из фаз перед началом следующей.

Тестирование методами "белого ящика" (White-Box Tests). См. *Структурное тестирование* (Structural Tests)

БИБЛИОГРАФИЯ

Adams, Douglas. *The Hitchhiker's Guide to the Galaxy.* New York: Balantine, 1995.

Allison, Anna. "Meaningful Metrics". *Software Testing and Quality Engineering,* Volume 3, Issue 3 (May/June 2001): pages 38 – 45.

Bach, James. "Exploratory Testing and the Planning Myth". www.stickyminds.com.

Bach, James. "Test Automation Snake-oil". www.satisfice.com.

Bartlett, Robert. "Power Testing". *Proceedings of the Fifth International Quality Week Europe.*

Beizer, Boris. *Black-Box Testing.* New York: Wiley, 1995. Есть русский перевод: *Бейзер Б.* Тестирование черного ящика: Технологии функционального тестирования программного обеспечения и систем. - СПб: Питер, 2004.

Beizer, Boris. "Software is Different". Software Research's Quality Techniques Newsletter archives. www.soft.com/News/QTN-Online/qtnapr01.html, www.soft.com/News/QTN-Online/qtnmay01.html, www.soft.com/News/ QTN-Online/qtnjun01.html.

Beizer, Boris. *Software System Testing and Quality Assurance.* New York: International Thomson Computer Press, 1996.

Beizer, Boris. *Software Testing Techniques, Second Edition.* New York: Van Nostrand Reinhold, 1990.

Bernstein, Peter. *Against the Gods.* New York: Wiley, 1996.

Binder, Robert. *Testing Object-Oriented Software.* Boston: Addison-Wesley, 1999.

Black, Rex. "Critical Software Testing Processes". *Journal of Software Testing Professionals,* Volume 1, Issue 1 (Mar 2000).

Black, Rex. "Effective Test Status Reporting". *Software Testing and Quality Engineering,* Volume 2, Issue 2 (Mar/Apr 2000).

Black, Rex. "Factors That Influence Test Estimation". www.stickyminds.com.

Black, Rex. "Four Ways Tasting Adds Value". *Proceedings of the EuroSTAR 2002 Conference.*

Black, Rex. *Managing the Testing Process, Second Edition.* New York: Wiley, 2002.

Black, Rex. "Stop Destroying My Team with Bad MBOs: A Manager's New Year's Resolution". www.stickyminds.com.

Black, Rex. "Test Estimation". *Software Testing and Quality Engineering,* Volume 4, Issue 6 (Nov/Dec 2002).

Black, Rex. "The Test Execution Process". *Journal of Software Testing Professionals*, Volume 1, Issue 3 (Sep 2000).

Black, Rex. "Test Release Process". *Journal of Software Testing Professionals*, Volume 1, Issue 2 (June 2000).

Black, Rex, and Greg Kubaczkowski. "Mission Made Possible". *Software Testing and Quality Engineering*, Volume 4, Issue 4 (July/Aug 2002).

Boehm, Barry. *Software Engineering Economics.* Prentice-Hall PTR, 1982.

Brooks, Frederick P., Jr. *The Mythical Man-Month.* Reading, Mass.: Addison- Wesley, 1975.

Brooks, Frederick P., Jr. *The Mythical Man-Month, Second Edition.* Reading, Mass.: Addison-Wesley, 1995. Есть русский перевод: *Брукс Ф.* Мифический человеко-месяц, или Как создаются программные системы. – М.: Символ-Плюс, 2001.

Bullock, James. "Calculating the Value of Testing". *Software Testing and Quality Engineering*, Volume 2, Issue 3 (May/June 2000): pages 56 – 62.

Buwalda, Hans. "Soap Opera Testing". *Proceedings of the STAR East 2000 Conference.*

Campanella, Jack, ed. *Principles of Quality Costs.* Milwaukee, Wisc.: ASQ Quality Press, 1999.

Cervantes, Miguel. *Don Quixote.* New York: Penguin, 2003.

Cohn, Mike. "A Measured Response". *Software Testing and Quality Engineering*, Volume 4, Issue 5 (Sep/Oct 2002).

Collard, Ross. "Test Design: Developing Test Cases from Use Cases". *Software Testing and Quality Engineering*, Volume 1, Issue 4 (July/Aug 1999).

Copeland, Lee. "*Exploratory Planning*". www.stickyminds.com.

Copeland, Lee. *Mastering Software Test Design.* Forthcoming.

Copeland, Lee. "When Helping Doesn't Help". www.stickyminds.com.

Craig, Rick, and Stefan Jaskiel. *Systematic Software Testing.* New York: Artech House, 2002.

Crosby, Philip. *Quality Is Free: The Art of Making Quality Certain.* New York: McGraw-Hill, 1979.

De Jaeger, Peter. "The Risk in Risk Management". *Software Testing and Quality Engineering*, Volume 3, Issue 4 (July/Aug 2001): pages 63 – 64.

DeMarco, Tom. *The Deadline.* New York: Dorset House, 1998.

DeMarco, Tom, and Tim Lister. *Peopleware, Second Edition.* New York: Dorset House, 1999.

DeNardis, Chris. "Perspectives from a Test Manager". *Software Testing and Quality Engineering*, Volume 2, Issue 5 (Sep/Oct2000).

Deming, W. E. *Out of Crisis.* Boston: MIT Press, 2000.

Dobbins, James. *Software Quality Assurance and Evaluation.* Milwaukee, Wisc: ASQ Quality Press, 1990.

Drabick, Rodger. *Best Practices for Software Testing.* New York: Dorset House, 2003.

Drabick, Rodger. "Growth of Maturity in the Testing Process". www.softtest.org/articles/rdrabick3.htm.

Drabick, Rodger. "Modeling the Formal the Testing Process". *Software QA Magazine*, Volume 2, Number 2: pages 20 – 38.

Drabick, Rodger. "On-Track Requirements". *Software Testing and Quality Engineering*, Volume 1, Issue 3. Available at www.stickyminds.com.

Drabick, Rodger. "A Process Model of Software Quality Assurance/Software Quality Engineering". *Software Quality Professional*, Volume 2, Issue 4 (Sep 2000): pages 20 – 38.

Dustin, Elfriede. "Orthogonally Speaking". *Software Testing and Quality Engineering*, Volume 3, Issue 5 (Sep/Oct 2001).

Dustin, Elfriede, Jeff Rashka, and John Paul. *Automated Software Testing.* Boston: Addison-Wesley, 1999. Есть русский перевод: *Элфрид Дастин, Джефф Рэшка, Джон Пол.* Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. – М.: Лори, 2003.

Dyes, Tim. "Tracking Severity". *Software Testing and Quality Engineering*, Volume 1, Issue 2 (Mar/Apr 1999).

Ensworth, Patricia. *The Accidental Project Manager.* New York: Wiley, 2001.

Fewster, Mark, and Dorothy Graham. *Software Test Automation.* Harlow, UK: Addison-Wesley, 1999.

Gardner, Robert. "Resolving the Process Paradox". *Quality Progress*, Volume 34, Issue 3 (Mar 2001): pages 51 – 59.

Gilbreath, Robert. *Winning at Project Management.* New York: Wiley, 1986.

The Guide to the Project Management Body of Knowledge, 2000 Edition. Newtown Square, Penn.: Project Management Institute, 2000.

Hadden, Rita. "Credible Estimation for Small Projects". *Software Quality Professional*, Volume 3, Issue 2 (Mar 2001): pages 7 – 11.

Hendrickson, Elisabeth. "More Testing, Worse Quality". www.qualitytree.com.

Hetzel, Bill. *The Complete Guide to Software Testing.* New York: Wiley-QED, 1988.

Hoffman, Douglas. "Using Oracles in Testing Automation". www.softwarequality-methods.com.

Hoyer, Robert, and Brooke Hoyer. "What Is Quality?" *Quality Progress*, Volume 34, Issue 7 (July 2001): pages 52 – 62.

Humphrey, Watts. *Introduction to the Personal Software Process.* Reading, Mass.: Addison-Wesley, 1997.

Iberle, Kathleen, and Susan Bartlett. "Estimating Tester to Developer Ratios (or Not)". *Proceedings of the STAR West Conference 2001.* Orange Park, Fla: Software Quality Engineering, 2001.

International Standards Organization. *ISO/IEC 9126:1991 (E).* Geneva, Switzerland: ISO/IEC Copyright Office, 1991.

Ishikawa, Kaoru. *Guide to Quality Control.* Tokyo: Asian Productivity Organization, 1982.

Jalote, Pankoj. *CMM in Practice.* Boston: Addison-Wesley, 1999.

Johnson, Karen. "Delivering Unwelcome News to Developers". *Software Testing and Quality Engineering*, Volume 4, Issue 5 (Sep/Oct 2002).

Johnson, Paul. *Intellectuals.* New York: Harper and Row, 1988.

Jones, T. Capers. *Estimating Software Costs.* New York: McGraw-Hill, 1995.

Juran, J. M. *Juran on Planning for Quality.* New York: Free Press, 1988.

Juran, J. M., and Frank Gryna, eds. *Quality Control Handbook, Fourth Edition.* New York: McGraw-Hill, 1988.

Kan, Stephen. *Metrics and Models in Software Quality Engineering, Second Edition.* Boston: Addison-Wesley, 2003.

Kaner, Cem. "Architectures of Test Automation". www.kaner.com.

Kaner, Cem. "Cem Kaner on Rethinking Software Metrics". www.stickyminds.com.

Kaner, Cem. "Measurement of the Extent of Testing". www.kaner.com.

Kaner, Cem. "Quality Cost Analysis: Benefits and Risks". www.kaner.com.

Kaner, Cem. "Recruiting Software Testers". www.sdmagazine.com/documents/s=751/sdm9912f/9912f.htm.

Kaner, Cem, James Bach, and Bret Pettichord. *Lessons Learned in Software Testing.* New York: Wiley, 2001.

Kaner, Cem, Jack Falk, and Hung Quoc Nguyen. *Testing Computer Software, Second Edition.* New York: International Thomson Computer Press, 1993. Есть русский перевод: Канер Сэм и др. Тестирование программного обеспечения. – Киев: Диасофт, 2000.

Kaner, Cem, and David Pels. *Bad Software.* New York: Wiley, 1998.

Kasse, Tim, and Patricia McQuaid. "Software Configuration Management for Project Leaders". *Software Quality Professional,* Volume 2, Issue 4 (Sep 2000): pages 8 – 19.

Kit, Ed. *Software Testing in the Real World.* Harlow, UK; Addison-Wesley, 1995.

Koomen, Tim, and Martin Pol. *Test Process Improvement.* Harlow, UK: Addison-Wesley, 1999.

Marciniak, John, ed. *Encyclopedia of Software Engineering, Second Edition.* New York: Wiley, 2001.

Marick, Brian. "Classic Testing Mistakes". www.testing.com.

Marick, Brian. *Craft of Software Testing.* Upper Saddle River, NJ: Prentice Hall, 1995.

Marick, Brian. "New Models for Test Development". www.testing.com.

McClintic, Ron. "Load Performance and Scalability Testing Challenges in E-Commerce". *Software Quality,* Number 2 (Winter 2001): pages 1 and 3 – 5.

McConnell, Steve. *After the Gold Rush.* Redmond, Wash.: Microsoft Press, 1999.

McConnell, Steve. *Software Project Survival Guide.* Redmond, Wash.: Microsoft Press, 1998.

McDermott, Robin, Raymond Mikulak, and Michael Beauregard. *The Basics of FMEA.* Portland, Ore: Productivity, 1996.

Mendonca, John, and Robert Lineberger. "Methodology as Road Kill: The Decade-Long Assault on Quality Assurance". *Software Quality,* Number 3 (Spring 2001), pages 1, 3, and 4.

Moore, Geoffrey. *Crossing the Chasm, Second Edition.* New York: Harper Collins, 1999.

Myers, Glenford. *The Art of Software Testing.* New York: Wiley, 1979. Есть русский перевод: *Майерс Г.* Искусство тестирования программ. – М., Финансы и статистика, 1981.

Neumann, Peter. *Computer-Related Risks.* Reading, MA: Addison-Wesley, 1995.

Nguyen, Hung. *Testing Applications on the Web.* New York: Wiley, 2000.

Nielsen, Jakob. *Usability Engineering.* San Francisco: Morgan Kaufmann Publishers, 1993.

Nishi Yasuharu. "Resource Path Testing: A Framework for Design of System Testing". *Software Quality Professional,* Volume 3, Issue 3 (June 2001).

Nyman, Noel. "Using Monkey Test Tools". *Software Testing and Quality Engineering,* Volume 2, Issue 1 (Jan/Feb 2000).

Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The Capability Maturity Model.* Reading, Mass.: Addison-Wesley, 1995.

Payne, Jeff. "Quality Meets the CEO". *Software Testing and Quality Engineering*, Volume 1, Issue 3 (May/June 1999).

Perry, Dewayne, and Gail Kaiser. "Object-Oriented Programs and Testing". citeseer.nj.nec.com/perry90objectoriented.html.

Perry, William, and Randall Rice. *Surviving the Top Ten Challenges of Software Testing.* New York: Dorset House, 1997.

Petersen, Erik. "Smarter Testing Using the 80/20 Rule". *Proceedings of the STAR West 2002 Conference.*

Plato. *The Trial and Death of Socrates, Third Edition.* New York: Hackett Publishing, 2001.

Pol, Martin, et al. *Software Testing: A Guide to the T-MAP Approach.* Boston: Addison-Wesley, 2001.

Pressman, Roger. *Software Engineering: A Practitioner's Approach, Fourth Edition.* New York: McGraw-Hill, 1997.

Quentin, Geoff. "People Skills, Including the Review as a Test Technique". *Professional Tester*, Volume 2, Issue 2 (June 2001): pages 42 – 43.

Rashka, Jeff. "Test Effort Estimation". *Journal of Software Testing Professionals*, Volume 2, Issue 1 (Mar 2001): pages 3 – 7 and 38.

Rosenberg, Linda, and Lawrence Hyatt. "Developing a Successful Metrics Program". satc.gsfc.nasa.gov/support/index.html.

Rothman, Johanna. "Hiring Technical People: A Guide to Hiring the Right People for the Job". www.jrothman.com.

Rothman, Johanna. "The Influential Test Manager". *Software Testing and Quality Engineering*, Volume 2, Issue 2 (Mar/Apr 2000).

Rothman, Johanna. "It Depends: Deciding on the Correct Ratio of Developers to Testers". www.jrothman.com.

Rothman, Johanna. "Manager Heal Thyself". www.jrothman.com.

Rothman, Johanna. "Release Criteria". *Software Testing and Quality Engineering*, Volume 4, Issue 2 (Mar/Apr 2002).

Rothman, Johanna. "Testing in the Dark". *Software Testing and Quality Engineering*, Volume 1, Issue 2 (Mar/Apr 1999).

Ruhl, Janet. *The Computer Consultant's Guide.* New York: Wiley, 1997.

Sabourin, Robert. "At Your Service". *Software Testing and Quality Engineering*, Volume 3, Issue 3 (May/June 2001): pages 46 – 52.

Sabourin, Robert. *I Am a Bug.* Self-published, 1999.

Sabourin, Robert, and Kim Davis. "Exploring, Discovering and Exterminating Bug Clusters in Web Applications". www.amibug.com.

Schmauch, Charles. *ISO 9000 for Software Developers.* Milwaukee, Wisc.: ASQ Quality Press, 1994.

Schmidt, Michael E. C. *Implementing the IEEE Software Engineering Standards.* Indianapolis, Ind.: SAMS Publishing, 2000.

Simmons, Erik. "The Human Side of Risk". *Proceedings of the Fifteenth International Internet and Software Quality Week 2002.*

Slaughter, Sandra, David Harter, and Mayuram Krishnan. "Evaluating the Cost of Software Quality". *Communications of the ACM*, Volume 41, Number 8 (Aug 1998): pages 67 – 73.

Splaine, Steven. *Testing Web Security.* New York: Wiley, 2002.

Splaine, Steven, et al. *The Web Testing Handbook.* Orange Park, Fla: Software Quality Engineering Publishing, 2001.

Stamatis, D. H. *Failure Mode and Effect Analysis.* Milwaukee, Wisc.: ASQ Quality Press, 1995.

Taguchi, Genichi, and Rajesh Jugulum. "Principles of Taguchi Methods for Software Testing". *Proceedings of the Second World Congress for Software Quality.* Tokyo: Union of Japanese Scientists and Engineers, 2000.

Taylor, Frederick Winslow. *The Principles of Scientific Management.* 1911.

Tolstoy, Leo. *Anna Karenina.* New York: Modern Library, 2000.

Tripp, Leonard, ed. *IEEE Software Engineering Standards Collection, 1997 Edition.* New York: IEEE, 1997.

Tufte, Edward. *Envisioning Information.* Cheshire, Conn.: Graphics Press, 1990.

Tufte, Edward. *The Visual Display of Quantitative Information, Second Edition.* Cheshire, Conn.: Graphics Press, 2001.

Tufte, Edward. *Visual Explanations.* Cheshire, Conn.: Graphics Press, 1997.

Tzu, Sun. *The Art of War.* New York: Delacorte Press, 1983.

VanDoren, Edmond. "Cyclomatic Complexity". Software Engineering Institute's Software Technology Review. www.sei.cmu.edu/str/descriptions/ cyclomatic_ body.html.

Van Veenendaal, Erik. *The Testing Practitioner.* Den Bosch, Netherlands: UTN Publishers, 2002.

Van Veenendaal, Erik, Rob Hendriks, and Robert van Vonderen. "Measuring Software Product Quality during Testing". *Professional Tester,* Volume 2, Issue 1 (Mar 2001): pages 26 – 30.

Walsh, James. *True Odds.* Santa Monica, Calif.: Merritt Publishing, 1996.

Walton, Mary. *The Deming Management Method.* New York: Putnam Publishing Group, 1986.

Weinberg, Gerald M. *The Psychology of Computer Programming.* New York: Van Nostrand Reinhold, 1971.

Weinberg, Gerald M. *The Secrets of Consulting.* New York: Dorset House, 1985.

Wert, Jeffrey. *Custer.* New York: Simon & Schuster, 1996.

Wiegers, Karl. *Software Requirements.* Redmond, Wash.: Microsoft Press, 1999. Есть русский перевод: *Вигерс К.* Разработка требований к программному обеспечению. – М: Русская редакция, 2004.

Wysocki, Robert, Robert Beck, Jr., and David Crane. *Effective Project Management.* New York: Wiley, 1995.

Xerox Corporate Education and Training. "Benchmarking for Quality Improvement". *Leadership Through Quality.* Stamford, Conn: Xerox, 1989.

Xerox Corporate Education and Training. "Concepts of Quality". *Leadership Through Quality.* Stamford, Conn: Xerox, 1988.

Xerox Corporate Education and Training. "Problem-Solving Process". *Leadership Through Quality.* Stamford, Conn: Xerox, 1986.

Xerox Corporate Education and Training. "Quality Improvement Process". *Leadership Through Quality.* Stamford, Conn: Xerox, 1988.

Yourdon, Ed. *Death March.* Upper Saddle River, NJ: Prentice Hall, 1999. Есть русский перевод: *Йордон Э.* Путь камикадзе. Второе издание. – М.: Лори, 2004.

Ключевые процессы тестирования

"Рекс Блэк написал новую книгу, одну из тех, которую тестировщики читают от корки до корки, а затем часто ссылаются на нее."

— Рик Крэйг, менеджер по тестированию программного обеспечения, автор книг, лектор и консультант

В современных условиях стремительно меняющейся среды разработки процессы тестирования программного обеспечения играют все большую роль. Если методологии ускоренной разработки нацелены на потребность компании в быстром выпуске продукта, то процессы тестирования направлены на столь же существенную потребность в выпуске его в надлежащем виде.

В этой книге Рекс Блэк, опираясь на свой богатый опыт, выделяет *двенадцать процессов тестирования,* являющихся ключевыми для достижения успеха. За описанием каждого из этих процессов следует отменно выстроенный пример использования процесса в различных организационных, операционных и технологических условиях. Вместо громоздких правил представлены списки контрольных вопросов — легкие, гибкие инструменты для внедрения тестирования, ориентированного на процесс, для сбора измерений и внесения последовательных изменений.

Вы научитесь:

- Наиболее эффективно и последовательно проводить тестирование
- Собирать сплоченную, слаженно работающую команду
- Создавать репутацию надежности за счет эффективного предоставления результатов тестирования
- Нацеливать тестирование на достижение максимального удовлетворения потребностей заказчика и стоящих перед компанией целей

Поскольку тестирование требует коллективных усилий и участия сотрудников различных подразделений компании, автор серьезно изучает вопросы межличностного общения и культурные аспекты. Книга даст вам новое представление о внутренних аспектах вашей деятельности и поможет лучше организовать работу.

Рекс Блэк — президент и ведущий консультант компании Rex Black Consulting Services, Inc., предоставляющей услуги тестирования и контроля качества таким клиентам, как Bank One, Cisco, Dell и Министерство обороны США. Автор книги *"Managing the Test Process"* ("Управление процессом тестирования"), второе издание которой вышло в издательстве John Wiley & Sons в 2002 г.

www.awprofessional.com

Дизайн обложки — Мелани Бак
Фото на обложке © Royalty Free/C

интернет-магазин
bookshop
http://www.bookshop.ua
тел. (044) 550-9050, 201-7012

Addison-Wesley
Pearson Education

ательство "ЛОРИ"
www. lory-press.ru