

УДК 004.415.53
ББК 32.372
А67

Аниче М.

А67 Эффективное тестирование программного обеспечения / пер. с англ.
А. Н. Киселева. – М.: ДМК Пресс, 2023. – 370 с.: ил.

ISBN 978-5-97060-997-2

В этой книге представлены основы систематического эффективного тестирования программного обеспечения. Показаны способы автоматизировать часть этого процесса, приводятся шаблоны проектирования, которые помогут писать легко контролируемый и простой для наблюдения код. Обсуждаются модульные, интеграционные и системные тесты; рассматривается передовой опыт работы с тестовым кодом.

Издание адресовано разработчикам с разным уровнем знаний: начинающие детально изучат процесс тестирования на конкретных примерах, опытные познакомятся с новыми практическими приемами и отточат имеющиеся навыки.

УДК 004.415.53
ББК 32.372

© LICENSEE 2023. Authorized translation of the English edition © 2021 Manning Publications. This translation is published and sold by permission of Manning Publications, the owner of all rights to publish and sell the same.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-6334-3993-1 (англ.)
ISBN 978-5-97060-997-2 (рус.)

© Manning Publications, 2022
© Перевод, оформление, издание, ДМК Пресс, 2022

Оглавление

1	■ Эффективное и систематическое тестирование программного обеспечения	28
2	■ Тестирование на основе спецификаций	64
3	■ Структурное тестирование и охват кода	101
4	■ Проектирование по контрактам	138
5	■ Тестирование на основе свойств	161
6	■ Дублиеры и имитации для тестирования	187
7	■ Проектирование с учетом простоты тестирования	222
8	■ Разработка через тестирование	251
9	■ Большие тесты	270
10	■ Качество тестового кода	319
11	■ Заключение	340

Содержание

Оглавление	5
Предисловие	12
Вступление	15
Благодарности	17
О книге	20
Об авторе	26
Об иллюстрации на обложке	27

1 Эффективное и систематическое тестирование программного обеспечения 28

1.1	Разница между разработчиками, тестирующими и не тестирующими свой код	29
1.2	Эффективное тестирование программного обеспечения для разработчиков.....	40
1.2.1	Эффективное тестирование в процессе разработки	41
1.2.2	Эффективное тестирование как итеративный процесс	43
1.2.3	Сосредоточение внимания на разработке, а затем на тестировании.....	43
1.2.4	Миф о «правильности по замыслу».....	44
1.2.5	Стоимость тестирования	44
1.2.6	Что подразумевается под словами «эффективный» и «систематический»	45
1.2.7	Роль автоматизации тестирования	45
1.3	Принципы тестирования программного обеспечения (или Почему тестирование такое сложное)	46
1.3.1	Всеобъемлющее тестирование невозможно	46
1.3.2	Своевременное прекращение тестирования.....	47
1.3.3	Изменчивость важна (парадокс пестицидов)	47
1.3.4	В одних частях ошибок больше, чем в других.....	47
1.3.5	Никакой объем тестирования не будет идеальным или достаточным.....	48
1.3.6	Контекст имеет решающее значение	48
1.3.7	Верификация не валидация.....	49
1.4	Пирамида тестирования и на чем следует сосредоточиться.....	49
1.4.1	Модульное тестирование	50
1.4.2	Интеграционное тестирование.....	52
1.4.3	Системное тестирование	53

1.4.4	Когда использовать каждый уровень тестирования	54
1.4.5	Почему я предпочитаю модульные тесты?	55
1.4.6	Что я тестирую на разных уровнях?	56
1.4.7	Что делать, если вы не согласны с пирамидой тестирования	57
1.4.8	Поможет ли эта книга найти все ошибки?	60
Упражнения		60
Итоги		62

2 Тестирование на основе спецификаций

2.1	Требования говорят сами за себя	65
2.1.1	Шаг 1: изучение требований, входных и выходных данных	68
2.1.2	Шаг 2: исследование, что делает программа для различных входных данных	68
2.1.3	Шаг 3: изучение возможных входных и выходных данных и определение разделов	69
2.1.4	Шаг 4: анализ границ	72
2.1.5	Шаг 5: определение списка тестов	74
2.1.6	Шаг 6: автоматизация тестирования	76
2.1.7	Шаг 7: расширение набора тестов с применением творческой смекалки и опыта	79
2.2	Коротко о тестировании на основе спецификаций	80
2.3	Поиск ошибок с помощью тестирования на основе спецификаций	82
2.4	Тестирование на основе спецификаций в реальных условиях	91
2.4.1	Процесс тестирования должен быть итеративным, а не последовательным	91
2.4.2	Как далеко следует заходить при тестировании на основе спецификаций?	91
2.4.3	Раздел или граница? Не имеет значения!	91
2.4.4	Точек включения и исключения достаточно, но не стесняйтесь добавлять точки входа и выхода	92
2.4.5	Используйте варианты одних и тех же входных данных для более полного понимания	92
2.4.6	Когда количество комбинаций резко возрастает, оставайтесь прагматичными	92
2.4.7	Если сомневаетесь, используйте самые простые входные данные	93
2.4.8	Выбирайте разумные значения входных данных, которые вас не беспокоят	93
2.4.9	Проверяйте на null и исключительные случаи, только когда это имеет смысл	93
2.4.10	Используйте параметризованные тесты, когда тесты имеют одну и ту же структуру	94
2.4.11	Требования могут иметь любую степень детализации	94
2.4.12	Применима ли предложенная методика для тестирования классов и состояний?	95
2.4.13	Роль опыта и творчества	96
Упражнения		97
Итоги		99

3 Структурное тестирование и охват кода

3.1	Охват кода, правильный способ	102
3.2	Кратко о структурном тестировании	105

3.3	Критерии охвата кода	107
3.3.1	Охват строк	107
3.3.2	Охват ветвей	107
3.3.3	Охват условий + ветвей	108
3.3.4	Охват путей	109
3.4	Сложные условия и критерий охвата MC/DC	110
3.4.1	Абстрактный пример	110
3.4.2	Создание набора тестов по критерию MC/DC	111
3.5	Тестирование циклов и других подобных конструкций	113
3.6	Классификация и выбор критериев	114
3.7	Тестирование на основе спецификаций и структурное тестирование: практический пример	115
3.8	Граничное и структурное тестирование	120
3.9	Одного структурного тестирования часто недостаточно	121
3.10	Структурное тестирование в реальном мире	123
3.10.1	Почему некоторые испытывают неприязнь к оценке охвата кода?	123
3.10.2	Что означает достижение 100 % охвата?	125
3.10.3	Какой критерий охвата использовать	127
3.10.4	MC/DC для слишком сложных выражений, которые нельзя упростить	127
3.10.5	Другие критерии охвата	129
3.10.6	Когда полный охват нежелателен?	129
3.11	Мутационное тестирование	130
	Упражнения	133
	Итоги	137

4 Проектирование по контрактам

4.1	Пред- и постусловия	139
4.1.1	Ключевое слово <code>assert</code>	140
4.1.2	Строгие и слабые пред- и постусловия	141
4.2	Инварианты	143
4.3	Изменение контрактов и принцип подстановки Лисков	147
4.3.1	Наследование и контракты	149
4.4	Как проектирование по контрактам связано с тестированием?	151
4.5	Проектирование по контрактам в реальном мире	152
4.5.1	Слабые или строгие предусловия?	153
4.5.2	Проверка допустимости входных данных, контракты или и то и другое?	153
4.5.3	Утверждения и исключения: когда использовать то или другое ...	155
4.5.4	Исключение или возврат специального значения?	157
4.5.5	Когда не следует использовать проектирование по контрактам	157
4.5.6	Следует ли писать тесты для предусловий, постусловий и инвариантов?	158
4.5.7	Инструментальная поддержка	158
	Упражнения	159
	Итоги	160

5 Тестирование на основе свойств

5.1	Пример 1: программа проверки оценок за экзамены	162
-----	---	-----

5.2	Пример 2: тестирование метода unique	166
5.3	Пример 3: тестирование метода indexOf.....	168
5.4	Пример 4: тестирование класса Basket	174
5.5	Пример 5: создание сложных объектов предметной области.....	182
5.6	Тестирование на основе свойств в реальном мире	183
5.6.1	Тестирование на основе примеров и на основе свойств	183
5.6.2	Общие проблемы в тестах на основе свойств	184
5.6.3	Творческая смекалка имеет решающее значение	185
	Упражнения.....	186
	Итоги	186

6	Дублеры и имитации для тестирования	187
6.1	Пустышки, фиктивные объекты, заглушки, шпионы и имитации	190
6.1.1	Объекты-пустышки	190
6.1.2	Фиктивные объекты	190
6.1.3	Заглушки	190
6.1.4	Имитации	191
6.1.5	Шпионы	191
6.2	Введение в фреймворки имитаций	191
6.2.1	Заглушки	192
6.2.2	Имитации и ожидания.....	197
6.2.3	Захват аргументов	200
6.2.4	Моделирование исключений	204
6.3	Имитации в реальном мире	206
6.3.1	Недостатки имитаций	207
6.3.2	Что следует и не следует имитировать.....	208
6.3.3	Обертки для даты и времени.....	213
6.3.4	Имитация типов, которыми вы не владеете	215
6.3.5	Что другие говорят об имитациях?	217
	Упражнения.....	219
	Итоги	220

7	Проектирование с учетом простоты тестирования.....	222
7.1	Отделение инфраструктурного кода от предметного	223
7.2	Внедрение зависимостей и управляемость.....	232
7.3	Улучшение наблюдаемости классов и методов.....	235
7.3.1	Пример 1: добавление методов для упрощения проверок	236
7.3.2	Пример 2: наблюдение за поведением методов void	237
7.4	Передача зависимостей через конструктор класса и значений через параметры методов	240
7.5	Проектирование с учетом простоты тестирования на практике.....	244
7.5.1	Связность тестируемого класса.....	244
7.5.2	Тесная связь с тестируемым классом.....	245
7.5.3	Сложные условия и тестируемость	246
7.5.4	Приватные методы и тестируемость	246
7.5.5	Статические методы, синглтоны и тестируемость	247
7.5.6	Гексагональная архитектура и имитации как метод проектирования	247
7.5.7	Дополнительная информация о проектировании с учетом	

тестируемости	248
Упражнения.....	248
Итоги	250

8 Разработка через тестирование	251
8.1 Наш первый сеанс TDD	252
8.2 Размышления о первом опыте применения TDD	260
8.3 TDD в реальном мире	262
8.3.1 Использовать или не использовать TDD?	262
8.3.2 TDD следует использовать постоянно?	263
8.3.3 Подходит ли TDD для всех типов приложений?	263
8.3.4 Что говорят исследования о TDD?	264
8.3.5 Другие школы TDD	265
8.3.6 TDD и правильное тестирование	267
Упражнения.....	267
Итоги	269

9 Большие тесты	270
9.1 Когда использовать большие тесты	271
9.1.1 Тестирование больших компонентов	271
9.1.2 Тестирование больших компонентов, взаимодействующих с внешним кодом	280
9.2 База данных и тестирование SQL	285
9.2.1 Что тестировать в SQL-запросе.....	285
9.2.2 Автоматизированные тесты для SQL-запросов.....	287
9.2.3 Настройка инфраструктуры для тестирования SQL	293
9.2.4 Рекомендации	295
9.3 Системные тесты	296
9.3.1 Введение в Selenium	297
9.3.2 Проектирование объектов страниц	300
9.3.3 Шаблоны и лучшие практики	310
9.4 Заключительные замечания по большим тестам	314
9.4.1 Как сочетаются методы тестирования в больших тестах?.....	314
9.4.2 Анализ затрат/выгод	315
9.4.3 Будьте осторожны с методами, охваченными, но не проверенными тестами	315
9.4.4 Инфраструктурный код имеет большое значение	316
9.4.5 DSL и инструменты для разработки тестов клиентами	316
9.4.6 Тестирование веб-систем других типов	316
Упражнения.....	317
Итоги	318

10 Качество тестового кода	319
10.1 Отличительные черты поддерживаемого тестового кода	320
10.1.1 Тесты должны быть быстрыми	320
10.1.2 Тесты должны быть связными, независимыми и изолированными	320
10.1.3 Тесты должны иметь причину для существования	321
10.1.4 Тесты должны быть воспроизводимыми и надежными	321
10.1.5 Тесты должны иметь строгие утверждения	323

10.1.6	Тесты должны терпеть неудачу при изменении поведения.....	323
10.1.7	Тесты должны иметь единственную и четкую причину неудачи...	324
10.1.8	Тесты должны быть простыми в разработке	324
10.1.9	Тесты должны легко читаться	325
10.1.10	Тесты должны позволять легко изменять и развивать их.....	329
10.2	Дурно пахнущие тесты.....	329
10.2.1	Чрезмерное дублирование.....	330
10.2.2	Нечеткие утверждения.....	331
10.2.3	Неправильное обращение со сложными или внешними ресурсами	331
10.2.4	Слишком обобщенные наборы тестовых данных.....	332
10.2.5	Чувствительные утверждения	333
	Упражнения.....	336
	Итоги	339

11	Заключение	340
11.1	Хотя модель выглядит линейной, итерации имеют фундаментальное значение	340
11.2	Разработка программного обеспечения без ошибок: миф или реальность?	341
11.3	Вовлекайте в процесс тестирования конечных пользователей.....	342
11.4	Модульное тестирование – сложная задача	343
11.5	Уделяйте внимание мониторингу.....	344
11.6	Что дальше?	344
	Приложение А. Решения упражнений	346
	Ссылки.....	354
	Предметный указатель.....	361