

Владимир Хориков

Принципы ЮНИТ- тестирования



Санкт-Петербург • Москва • Минск

2025

Оглавление

Предисловие к русскому изданию	14
Предисловие к оригинальному изданию	15
Благодарности	16
О книге	17
Для кого написана эта книга.....	17
Структура книги.....	18
О коде.....	18
Форум для обсуждения книги	19
Об авторе	19
Иллюстрация на обложке.....	19
От издательства.....	20
Часть I. Общая картина	21
Глава 1. Цель юнит-тестирования.....	22
1.1. Текущее состояние дел в юнит-тестировании.....	23
1.2. Цель юнит-тестирования.....	24
1.2.1. В чем разница между плохими и хорошими тестами?	26

1.3. Использование метрик покрытия для оценки качества тестов	28
1.3.1. Метрика покрытия code coverage	28
1.3.2. Branch coverage	30
1.3.3. Проблемы с метриками покрытия.....	31
1.3.4. Процент покрытия как цель	34
1.4. Какими должны быть успешные тесты?	35
1.4.1. Интеграция в цикл разработки.....	35
1.4.2. Проверка только самых важных частей кода	35
1.4.3. Максимальная защита от багов при минимальных затратах на сопровождение	36
1.5. Что вы узнаете из книги	37
Итоги	38
Глава 2. Что такое юнит-тест?	40
2.1. Определение юнит-теста	40
2.1.1. Вопрос изоляции: лондонская школа	41
2.1.2. Вопрос изоляции: классический подход	47
2.2. Классическая и лондонская школы юнит-тестирования.....	50
2.2.1. Работа с зависимостями в классической и лондонской школах.....	51
2.3. Сравнение классической и лондонской школ юнит-тестирования	54
2.3.1. Юнит-тестирование одного класса за раз.....	55
2.3.2. Юнит-тестирование большого графа взаимосвязанных классов.....	56
2.3.3. Выявление точного местонахождения ошибки	57
2.3.4. Другие различия между классической и лондонской школами	57
2.4. Интеграционные тесты в двух школах	58
2.4.1. Сквозные (end-to-end) тесты как подмножество интеграционных тестов	60
Итоги	62
Глава 3. Анатомия юнит-теста.....	64
3.1. Структура юнит-теста	65
3.1.1. Паттерн AAA.....	65
3.1.2. Избегайте множественных секций arrange, act и assert	66
3.1.3. Избегайте команд if в тестах.....	67
3.1.4. Насколько большой должна быть каждая секция?	68

3.1.5. Сколько проверок должна содержать секция проверки?	70
3.1.6. Нужна ли завершающая (teardown) фаза?	71
3.1.7. Выделение тестируемой системы	71
3.1.8. Удаление комментариев «arrange/act/assert» из тестов	72
3.2. Фреймворк тестирования xUnit	72
3.3. Переиспользование тестовых данных между тестами	74
3.3.1. Сильная связность (high coupling) между тестами как антипаттерн	75
3.3.2. Использование конструкторов в тестах ухудшает читаемость	76
3.3.3. Более эффективный способ переиспользования тестовых данных	76
3.4. Именованые юнит-тесты	78
3.4.1. Рекомендации по именованию юнит-тестов	80
3.4.2. Пример: переименование теста в соответствии с рекомендациями	80
3.5. Параметризованные тесты	82
3.5.1. Генерирование данных для параметризованных тестов	85
3.6. Использование библиотек для дальнейшего улучшения читаемости тестов	86
Итоги	88
Часть II. Обеспечение эффективной работы ваших тестов	89
Глава 4. Четыре аспекта хороших юнит-тестов	90
4.1. Четыре аспекта хороших юнит-тестов	91
4.1.1. Первый аспект: защита от багов	91
4.1.2. Второй аспект: устойчивость к рефакторингу	92
4.1.3. Что приводит к ложным срабатываниям?	94
4.1.4. Тестирование конечного результата вместо деталей имплементации	98
4.2. Связь между первыми двумя атрибутами	99
4.2.1. Максимизация точности тестов	100
4.2.2. Важность ложных и ложноотрицательных срабатываний: динамика	101
4.3. Третий и четвертый аспекты: быстрая обратная связь и простота поддержки	103
4.4. В поисках идеального теста	103
4.4.1. Возможно ли создать идеальный тест?	104

4.4.2. Крайний случай № 1: сквозные (end-to-end) тесты	105
4.4.3. Крайний случай № 2: тривиальные тесты	106
4.4.4. Крайний случай № 3: хрупкие тесты	106
4.4.5. В поисках идеального теста: результаты	108
4.5. Известные концепции автоматизации тестирования	111
4.5.1. Пирамида тестирования	111
4.5.2. Выбор между тестированием по принципу «черного ящика» и «белого ящика»	114
Итоги	115
Глава 5. Моки и хрупкость тестов	117
5.1. Отличия моков от стабов	118
5.1.1. Разновидности тестовых заглушек	118
5.1.2. Мок-инструмент и мок — тестовая заглушка	120
5.1.3. Не проверяйте взаимодействия со стабами	121
5.1.4. Использование моков вместе со стабами	122
5.1.5. Связь моков и стабов с командами и запросами	123
5.2. Наблюдаемое поведение и детали имплементации	124
5.2.1. Наблюдаемое поведение — не то же самое, что публичный API	125
5.2.2. Утечка деталей имплементации: пример с операцией	126
5.2.3. Хорошо спроектированный API и инкапсуляция	129
5.2.4. Утечка деталей имплементации: пример с состоянием	130
5.3. Связь между моками и хрупкостью тестов	132
5.3.1. Определение гексагональной архитектуры	132
5.3.2. Внутрисистемные и межсистемные взаимодействия	136
5.3.3. Внутрисистемные и межсистемные взаимодействия: пример	138
5.4. Еще раз о различиях между классической и лондонской школами юнит-тестирования	141
5.4.1. Не все внепроцессные зависимости должны заменяться моками	142
5.4.2. Использование моков для проверки поведения	144
Итоги	144
Глава 6. Стили юнит-тестирования	147
6.1. Три стиля юнит-тестирования	148
6.1.1. Проверка выходных данных	148

6.1.2. Проверка состояния	149
6.1.3. Проверка взаимодействий	150
6.2. Сравнение трех стилей юнит-тестирования.....	151
6.2.1. Сравнение стилей по метрикам защиты от багов и скорости обратной связи	152
6.2.2. Сравнение стилей по метрике устойчивости к рефакторингу	152
6.2.3. Сравнение стилей по метрике простоты поддержки	153
6.2.4. Сравнение стилей: результаты.....	156
6.3. Функциональная архитектура	157
6.3.1. Что такое функциональное программирование?	157
6.3.2. Что такое функциональная архитектура?	161
6.3.3. Сравнение функциональных и гексагональных архитектур	163
6.4. Переход на функциональную архитектуру и тестирование выходных данных	164
6.4.1. Система аудита.....	165
6.4.2. Использование моков для отделения тестов от файловой системы... 167	
6.4.3. Рефакторинг для перехода на функциональную архитектуру.....	170
6.4.4. Потенциальные будущие изменения.....	176
6.5. Недостатки функциональной архитектуры.....	177
6.5.1. Применимость функциональной архитектуры	177
6.5.2. Недостатки по быстродействию.....	179
6.5.3. Увеличение размера кодовой базы	179
Итоги.....	180
Глава 7. Рефакторинг для получения эффективных юнит-тестов.....	182
7.1. Определение кода для рефакторинга.....	183
7.1.1. Четыре типа кода.....	183
7.1.2. Использование паттерна «Простой объект» для разделения переусложненного кода.....	187
7.2. Рефакторинг для получения эффективных юнит-тестов	190
7.2.1. Знакомство с системой управления клиентами	190
7.2.2. Версия 1: преобразование неявных зависимостей в явные	192
7.2.3. Версия 2: уровень сервисов приложения.....	193
7.2.4. Версия 3: вынесение сложности из сервисов приложения.....	195
7.2.5. Версия 4: новый класс Company	197

7.3. Анализ оптимального покрытия юнит-тестов	200
7.3.1. Тестирование слоя предметной области и вспомогательного кода	200
7.3.2. Тестирование кода из трех других четвертей	201
7.3.3. Нужно ли тестировать предусловия?	202
7.4. Условная логика в контроллерах	203
7.4.1. Паттерн «CanExecute/Execute»	205
7.4.2. Использование доменных событий для отслеживания изменений доменной модели	208
7.5. Заключение	212
Итоги	214
Часть III. Интеграционное тестирование	217
Глава 8. Для чего нужно интеграционное тестирование?	218
8.1. Что такое интеграционный тест?	219
8.1.1. Роль интеграционных тестов	219
8.1.2. Снова о пирамиде тестирования	220
8.1.3. Интеграционное тестирование и принцип Fail Fast	222
8.2. Какие из внепроцессных зависимостей должны проверяться напрямую ...	224
8.2.1. Два типа внепроцессных зависимостей	224
8.2.2. Работа с управляемыми и неуправляемыми зависимостями	225
8.2.3. Что делать, если вы не можете использовать реальную базу данных в интеграционных тестах?	226
8.3. Интеграционное тестирование: пример	227
8.3.1. Какие сценарии тестировать?	228
8.3.2. Классификация базы данных и шины сообщений	229
8.3.3. Как насчет сквозного тестирования?	229
8.3.4. Интеграционное тестирование: первая версия	231
8.4. Использование интерфейсов для абстрагирования зависимостей	232
8.4.1. Интерфейсы и слабая связность	232
8.4.2. Зачем использовать интерфейсы для внепроцессных зависимостей?	233
8.4.3. Использование интерфейсов для внутривнепроцессных зависимостей	235
8.5. Основные приемы интеграционного тестирования	235

8.5.1. Явное определение границ модели предметной области.....	235
8.5.2. Сокращение количества слоев.....	236
8.5.3. Исключение циклических зависимостей.....	237
8.5.4. Использование нескольких секций действий в тестах	240
8.6. Тестирование функциональности логирования.....	241
8.6.1. Нужно ли тестировать функциональность логирования?.....	241
8.6.2. Как тестировать функциональность логирования?	243
8.6.3. Какой объем логирования можно считать достаточным?	248
8.6.4. Как передавать экземпляры логов?	249
8.7. Заключение	250
Итоги.....	250
Глава 9. Рекомендации при работе с моками.....	254
9.1. Достижение максимальной эффективности моков	254
9.1.1. Проверка взаимодействий на границах системы	257
9.1.2. Замена моков шпионами	261
9.1.3. Как насчет IDomainLogger?	263
9.2. Практики мокирования	263
9.2.1. Моки только для интеграционных тестов	264
9.2.2. Несколько моков на тест.....	264
9.2.3. Проверка количества вызовов.....	265
9.2.4. Используйте моки только для принадлежащих вам типов	265
Итоги.....	267
Глава 10. Тестирование базы данных	268
10.1. Предусловия для тестирования базы данных.....	269
10.1.1. Хранение базы данных в системе контроля версий.....	269
10.1.2. Справочные данные являются частью схемы базы данных.....	270
10.1.3. Отдельный экземпляр для каждого разработчика.....	271
10.1.4. Развертывание базы данных на основе состояния и на основе миграций.....	271
10.2. Управление транзакциями.....	274
10.2.1. Управление транзакциями в рабочем коде	274
10.2.2. Управление транзакциями в интеграционных тестах.....	282

10.3. Жизненный цикл тестовых данных	284
10.3.1. Параллельное или последовательное выполнение тестов?.....	284
10.3.2. Очистка данных между запусками тестов.....	285
10.3.3. Не используйте базы данных в памяти	287
10.4. Переиспользование кода в секциях тестов	287
10.4.1. Переиспользование кода в секциях подготовки	288
10.4.2. Переиспользование кода в секциях действий.....	290
10.4.3. Переиспользование кода в секциях проверки	291
10.4.4. Не создает ли тест слишком много транзакций?	292
10.5. Типичные вопросы при тестировании баз данных	293
10.5.1. Нужно ли тестировать операции чтения?.....	294
10.5.2. Нужно ли тестировать репозитории?	294
10.6. Заключение	296
Итоги.....	297
Часть IV. Антипаттерны юнит-тестирования.....	299
Глава 11. Антипаттерны юнит-тестирования	300
11.1. Юнит-тестирование приватных методов	300
11.1.1. Приватные методы и хрупкость тестов.....	301
11.1.2. Приватные методы и недостаточное покрытие.....	301
11.1.3. Когда тестирование приватных методов допустимо	302
11.2. Раскрытие приватного состояния.....	304
11.3. Утечка доменных знаний в тесты	306
11.4. Загрязнение кода	307
11.5. Мокирование конкретных классов.....	310
11.6. Работа со временем	313
11.6.1. Время как неявный контекст	313
11.6.2. Время как явная зависимость	314
11.7. Заключение	315
Итоги.....	315