Оглавление

Отзывы о втором издании **14**

Предисловие ко второму изданию **17**

Предисловие к первому изданию **19**

Введение **21**

Благодарности **23**

О книге **24**

Что нового в третьем издании 24
Для кого эта книга 25
Структура книги 25
О коде в книге 26
Программные требования 27
Форум LiveBook 27
Другие проекты Роя Ошерова 27
Другие проекты Владимира Хорикова 28

Об авторах **29**

Иллюстрация на обложке **30**

От издательства **31**

О научном редакторе русского издания 31

Часть I. Первые шаги

Глава 1. Основы юнит-тестирования **34**

1.1. Первый шаг 36
1.2. Определение юнит-тестирования шаг за шагом 37
1.3. Точки входа и точки выхода 38
1.4. Типы точек выхода 44
1.5. Разные точки выхода, разные методы 45
1.6. Написание теста с нуля 45

1.7. Характеристики хорошего юнит-теста ... 48
1.7.1. Как выглядит хороший юнит-тест? ... 49
1.7.2. Чек-лист юнит-теста ... 51
1.8. Интеграционные тесты ... 51
1.9. Окончательное определение ... 56
1.10. Разработка через тестирование ... 57
1.10.1. TDD не заменит хорошие юнит-тесты ... 59
1.10.2. Три основных навыка, необходимых для успешного применения TDD ... 61
Итоги ... 62

Глава 2. Первый юнит-тест ... 64

2.1. Знакомство с Jest ... 65
2.1.1. Подготовка среды ... 65
2.1.2. Подготовка рабочего каталога ... 65
2.1.3. Установка Jest ... 66
2.1.4. Создание файла теста ... 66
2.1.5. Выполнение Jest ... 68
2.2. Библиотека для тестирования, библиотека утверждений, запуск тестов и представление отчета ... 70
2.3. Что предлагают фреймворки юнит-тестирования ... 71
2.3.1. Фреймворки xUnit ... 73
2.3.2. Структуры xUnit, TAP и Jest ... 74
2.4. Проект Password Verifier ... 75
2.5. Первый тест Jest для verifyPassword ... 76
2.5.1. Паттерн «подготовь — действуй — проверь» ... 76
2.5.2. Тестируем тест ... 77
2.5.3. Имена USE ... 77
2.5.4. Сравнения строк и сопровождаемость ... 78
2.5.5. Использование describe() ... 79
2.5.6. Структура, подразумевающая контекст ... 80
2.5.7. Функция it() ... 81
2.5.8. Две разновидности Jest ... 81
2.5.9. Рефакторинг рабочего кода ... 82
2.6. Решение с beforeEach() ... 85
2.6.1. beforeEach() и утомительная прокрутка ... 86

2.7. Решение с фабричным методом ... 89
2.7.1. Полная замена beforeEach() фабричными методами ... 90
2.8. Возвращение к test() ... 92
2.9. Рефакторинг: параметризованные тесты ... 93
2.10. Проверка ожидаемых ошибок ... 95
2.11. Назначение категорий тестов ... 97
Итоги ... 98

Часть 2. Приемы и методы

Глава 3. Устранение зависимостей при помощи стабов ... 102
3.1. Типы зависимостей ... 103
3.2. Причины для использования стабов ... 106
3.3. Общепринятые подходы к созданию стабов ... 108
3.3.1. Создание стаба для времени с внедрением параметра ... 108
3.3.2. Зависимости, внедрения и контроль ... 110
3.4. Функциональные методы внедрения ... 111
3.4.1. Внедрение функции ... 112
3.4.2. Внедрение зависимости через частичное применение ... 113
3.5. Модульные методы внедрения ... 113
3.6. Переход к объектам с функциями-конструкторами ... 117
3.7. Объектно-ориентированные методы внедрения ... 117
3.7.1. Внедрение через конструктор ... 118
3.7.2. Внедрение объекта вместо функции ... 119
3.7.3. Выделение общего интерфейса ... 123
Итоги ... 126

Глава 4. Тестирование взаимодействий с использованием моков ... 128
4.1. Тестирование взаимодействий, моки и стабы ... 129
4.2. Зависимость от логгера ... 130
4.3. Стандартный стиль: рефакторинг с введением параметра ... 132
4.4. Почему важно отличать моки от стабов ... 134
4.5. Моки в модульном стиле ... 135
4.5.1. Пример рабочего кода ... 135
4.5.2. Рефакторинг рабочего кода в стиле модульного внедрения ... 137
4.5.3. Пример теста с внедрением в модульном стиле ... 138

4.6. Моки в функциональном стиле ... 139
4.6.1. Использование стиля каррирования ... 139
4.6.2. Работа с функциями высшего порядка без каррирования ... 140
4.7. Моки в объектно-ориентированном стиле ... 141
4.7.1. Рефакторинг рабочего кода для внедрения ... 141
4.7.2. Рефакторинг рабочего кода с внедрением интерфейсов ... 143
4.8. Сложные интерфейсы ... 145
4.8.1. Пример сложного интерфейса ... 146
4.8.2. Написание тестов со сложными интерфейсами ... 146
4.8.3. Недостатки прямого использования сложных интерфейсов ... 148
4.8.4. Принцип разделения интерфейсов ... 148
4.9. Частичные моки ... 148
4.9.1. Пример частичного мока ... 149
4.9.2. Объектно-ориентированный пример частичного мока ... 149
Итоги ... 151

Глава 5. Изолирующие фреймворки ... 152

5.1. Определение изолирующих фреймворков ... 153
5.1.1. Выбор разновидности: свободные и типизированные ... 154
5.2. Динамическое создание фейков для модулей ... 154
5.2.1. Некоторые особенности API Jest, на которые следует обратить внимание ... 157
5.2.2. Абстрагирование прямых зависимостей ... 158
5.3. Функциональные динамические моки и стабы ... 159
5.4. Объектно-ориентированные динамические моки и стабы ... 160
5.4.1. Использование фреймворка со свободной типизацией ... 160
5.4.2. Переход на фреймворк с поддержкой типов ... 162
5.5. Динамическое создание стабов ... 164
5.5.1. Объектно-ориентированный пример с моком и стабом ... 165
5.5.2. Стабы и моки с substitute.js ... 166
5.6. Преимущества и опасности изолирующих фреймворков ... 168
5.6.1. Моки в большинстве случаев не нужны ... 169
5.6.2. Нечитаемый код теста ... 169
5.6.3. Проверка не того, что нужно ... 170
5.6.4. Наличие более одного мока на тест ... 170

5.6.5. Излишняя детализация тестов ... 170
Итоги ... 171

Глава 6. Юнит-тестирование асинхронного кода ... 172
6.1. Асинхронная загрузка данных ... 173
6.1.1. Первая попытка написания интеграционного теста ... 174
6.1.2. Ожидание действия ... 175
6.1.3. Интеграционное тестирование async/await ... 175
6.1.4. Проблемы с интеграционными тестами ... 176
6.2. Создание кода, удобного для юнит-тестов ... 177
6.2.1. Выделение точки входа ... 177
6.2.2. Паттерн «выделение адаптера» ... 183
6.3. Таймеры ... 191
6.3.1. Создание стабов для таймеров ... 191
6.3.2. Фейки setTimeout средствами Jest ... 192
6.4. Распространенные события ... 194
6.4.1. Генераторы событий ... 194
6.4.2. События click ... 195
6.5. Использование DOM Testing Library ... 198
Итоги ... 199

Часть 3. Код тестов

Глава 7. Достоверные тесты ... 202
7.1. Как определить, что тесту можно доверять ... 203
7.2. Почему тесты не проходят ... 204
7.2.1. В рабочем коде была обнаружена реальная ошибка ... 204
7.2.2. Тест не проходит из-за ошибки в самом тесте ... 204
7.2.3. Тест устарел из-за изменения в функциональности ... 206
7.2.4. Тест конфликтует с другим тестом ... 206
7.2.5. Ненадежность теста ... 207
7.3. Избегание логики в юнит-тестах ... 207
7.3.1. Логика в утверждениях: создание динамических ожидаемых значений ... 208
7.3.2. Другие формы логики ... 209
7.3.3. Еще больше логики ... 211

7.4. Ложное чувство доверия к проходящим тестам ... 211
7.4.1. Тесты, которые ничего не проверяют ... 212
7.4.2. Тесты непонятны ... 213
7.4.3. Смешение юнит-тестов с ненадежными интеграционными тестами ... 213
7.4.4. Тестирование нескольких точек выхода ... 214
7.4.5. Тесты, которые продолжают изменяться ... 216
7.5. Ненадежные тесты ... 217
7.5.1. Что делать при обнаружении ненадежного теста? ... 219
7.5.2. Предотвращение ненадежности в высокоуровневых тестах ... 220
Итоги ... 221

Глава 8. Простота в сопровождении ... 222
8.1. Изменения из-за непроходящих тестов ... 223
8.1.1. Тест неактуален или конфликтует с другим тестом ... 223
8.1.2. Изменения в API рабочего кода ... 224
8.1.3. Изменения в других тестах ... 227
8.2. Рефакторинг для улучшения сопровождаемости ... 232
8.2.1. Избегайте тестирования приватных или защищенных методов ... 232
8.2.2. Соблюдение принципа DRY в тестах ... 234
8.2.3. Нежелательность подготовительных методов ... 234
8.2.4. Использование параметризованных тестов для устранения дублирования ... 235
8.3. Избегайте излишней детализации ... 236
8.3.1. Излишняя детализация внутреннего поведения с моками ... 237
8.3.2. Излишняя детализация выводов ... 239
Итоги ... 243

Часть 4. Проектирование и процесс

Глава 9. Читабельность ... 246
9.1. Выбор имен юнит-тестов ... 247
9.2. Магические значения и имена переменных ... 248
9.3. Отделение проверок от действий ... 250
9.4. Подготовка и завершение ... 250
Итоги ... 252

Глава 10. Разработка стратегии тестирования 253
10.1. Основные виды и уровни тестирования 254
10.1.1. Критерии для выбора вида теста 255
10.1.2. Юнит-тесты и компонентные тесты 255
10.1.3. Интеграционные тесты 257
10.1.4. Тесты API 257
10.1.5. Изолированные тесты E2E/UI 258
10.1.6. Системные тесты E2E/UI 259
10.2. Антипаттерны в разработке стратегии тестирования 260
10.2.1. Антипаттерн «только сквозные тесты» 260
10.2.2. Антипаттерн «только низкоуровневые тесты» 263
10.2.3. Рассоединение низкоуровневых и высокоуровневых тестов 265
10.3. Рецепты тестирования как стратегия 266
10.3.1. Как написать рецепт тестирования 266
10.3.2. Когда писать и использовать рецепты тестирования? 268
10.3.3. Правила составления рецептов тестирования 269
10.4. Управление конвейером поставки ПО 270
10.4.1. Конвейеры поставки и информационные конвейеры 270
10.4.2. Параллелизация уровней тестирования 272
Итоги 273

Глава 11. Интеграция юнит-тестирования в организацию 275
11.1. Как стать инициатором изменений 276
11.1.1. Подготовьтесь к непростым вопросам 276
11.1.2. Убедите окружающих: сторонников и скептиков 276
11.1.3. Определите возможные отправные точки 278
11.2. Пути к успеху 279
11.2.1. Партизанская реализация (снизу вверх) 280
11.2.2. Привлечение руководства (сверху вниз) 280
11.2.3. Эксперименты как первый шаг 281
11.2.4. Привлечение внешнего сторонника 282
11.2.5. Демонстрация прогресса 282
11.2.6. Ориентация на конкретные цели, метрики и KPI 284
11.2.7. Понимание возможных препятствий 286

11.3. Возможные неудачи ... 287
11.3.1. Недостаточный направляющий импульс ... 287
11.3.2. Недостаток политической поддержки ... 287
11.3.3. Ситуативные реализации и первые впечатления ... 288
11.3.4. Недостаток поддержки в команде ... 288
11.4. Факторы влияния ... 289
11.5. Непростые вопросы и ответы ... 291
11.5.1. Насколько юнит-тестирование удлинит текущий процесс? ... 291
11.5.2. Не будет ли юнит-тестирование угрожать моей работе в отделе QA? ... 293
11.5.3. Есть ли доказательства, что юнит-тестирование действительно помогает? ... 294
11.5.4. Почему отдел QA все еще находит ошибки? ... 294
11.5.5. У нас большой объем кода без тестов: с чего начинать? ... 294
11.5.6. А если мы разрабатываем комбинацию программного и аппаратного обеспечения? ... 295
11.5.7. Как узнать, что в наших тестах нет ошибок? ... 295
11.5.8. Зачем мне нужны тесты, если отладчик показывает, что мой код работает? ... 295
11.5.9. Как насчет TDD? ... 296
Итоги ... 296

Глава 12. Работа с унаследованным кодом ... 297
12.1. С чего начинать добавление тестов? ... 298
12.2. Принятие решения по стратегии выбора ... 300
12.2.1. Плюсы и минусы стратегии «начать с простого» ... 300
12.2.2. Плюсы и минусы стратегии «начать со сложного» ... 301
12.3. Написание интеграционных тестов перед рефакторингом ... 302
12.3.1. Прочитайте книгу Майкла Физерса об унаследованном коде ... 303
12.3.2. Используйте CodeScene для анализа своего рабочего кода ... 304
Итоги ... 304

Приложения. Манки-патчинг функций и модулей ... 305
П.1. Обязательное предупреждение ... 305
П.2. Манки-патчинг функций и возможные проблемы ... 306

П.2.1. Манки-патчинг функции по схеме Jest 308
П.2.2. Шпионы Jest 308
П.2.3. spyOn с mockImplementation() 309
П.3. Игнорирование целого модуля в Jest 310
П.4. Моделирование поведения модуля в каждом тесте 311
П.4.1. Создание стаба для модуля базовым вызовом require.cache 312
П.4.2. Создание стабов для нестандартных данных из модулей в Jest затруднено 314
П.4.3. Избегайте ручных моков Jest 316
П.4.4. Создание стаба для модуля с Sinon.js 316
П.4.5. Создание стаба для модуля с testdouble 317