

Марк Симан

Код, который умещается в голове

**Эвристики
для разработчиков**



Санкт-Петербург • Москва • Минск

2024

КРАТКОЕ СОДЕРЖАНИЕ

Предисловие Роберта Мартина	20
Введение	24
Об авторе	30

ЧАСТЬ I РАЗВИТИЕ

Глава 1. Искусство или наука?	32
Глава 2. Чек-листы: история, виды, преимущества	45
Глава 3. Преодоление трудностей	65
Глава 4. Вертикальный срез	80
Глава 5. Инкапсуляция	120
Глава 6. Триангуляция	144
Глава 7. Декомпозиция	163
Глава 8. Проектирование API	191
Глава 9. Командная работа	216

ЧАСТЬ II УСТОЙЧИВОСТЬ

Глава 10. Расширение кодовой базы	242
Глава 11. Редактирование модульных тестов	262
Глава 12. Устранение неполадок	275
Глава 13. Разделение ответственности	299
Глава 14. Организация рабочего процесса	319
Глава 15. Очевидные аспекты	331
Глава 16. Краткий обзор	356
Приложение. Перечень методов	377
Библиография	389

ОГЛАВЛЕНИЕ

Предисловие Роберта Мартина	20
Введение	24
Для кого эта книга	25
Исходные требования	25
Примечание для архитекторов ПО	26
Структура книги	26
О стиле кода	27
Типизировать явно или неявно	27
Примеры кода	28
Примечание к библиографии	28
О моих книгах	29
Благодарности	29
От издательства	29
Об авторе	30

ЧАСТЬ I РАЗВИТИЕ

Глава 1. Искусство или наука?	32
1.1. Строительство здания	33
1.1.1. Проблема проекта	33
1.1.2. Этапы разработки	34
1.1.3. Зависимости	35
1.2. Возделывание сада	36
1.2.1. Что заставляет сад расти?	36
1.3. С точки зрения инженерии	37
1.3.1. Программирование как ремесло	37
1.3.2. Эвристика	39
1.3.3. Ранние представления о разработке ПО	40
1.3.4. Становление и развитие программной инженерии	41
1.4. Заключение	43
Глава 2. Чек-листы: история, виды, преимущества	45
2.1. Как ничего не забыть	45
2.2. Чек-лист для новой кодовой базы	47
2.2.1. Использовать Git	48
2.2.2. Автоматизировать сборку	50
2.2.3. Включить все сообщения об ошибках	54
2.3. Добавление проверок в существующие кодовые базы	61
2.3.1. Постепенное улучшение	61
2.3.2. «Взломайте» свою организацию	62
2.4. Заключение	63
Глава 3. Преодоление трудностей	65
3.1. Цель	66
3.1.1. Надежность	67
3.1.2. Ценность	68

3.2. Почему программировать так сложно?	70
3.2.1. Аналогия с мозгом	70
3.2.2. Код больше читается, чем пишется	72
3.2.3. Удобочитаемость	73
3.2.4. Интеллектуальный труд	74
3.3. Навстречу программной инженерии	76
3.3.1. Связь с computer science	77
3.3.2. Гуманный код	78
3.4. Заключение	79
Глава 4. Вертикальный срез	80
4.1. Начните с рабочего ПО	81
4.1.1. От поступления данных до их сохранения	81
4.1.2. Минимальный вертикальный срез	82
4.2. «Ходячий скелет»	84
4.2.1. Характеризационные тесты	85
4.2.2. Паттерн AAA (Arrange-Act-Assert)	87
4.2.3. Модерация статического анализа	88
4.3. Модель тестирования «от общего к частному» (outside-in) ...	92
4.3.1. Получение данных JSON	93
4.3.2. Размещение бронирования	96
4.3.3. Модульное тестирование	101
4.3.4. DTO и модель предметной области (доменная модель)	103
4.3.5. Fake Object, или фиктивный объект	106
4.3.6. Интерфейс Repository	107
4.3.7. Работа с интерфейсом Repository	108
4.3.8. Настройка зависимостей	109
4.4. Завершение среза	111
4.4.1. Схема	111
4.4.2. Репозиторий SQL	113

4.4.3. Конфигурация базы данных.....	115
4.4.4. Дымовой тест, или smoke-тестирование.....	116
4.4.5. Граничный тест с фиктивной базой данных.....	117
4.5. Заключение	119
Глава 5. Инкапсуляция	120
5.1. Сохранение данных	120
5.1.1. Предпосылки приоритета трансформации (TPP)	121
5.1.2. Параметризованные тесты	123
5.1.3. Копирование данных dto в модель предметной области	124
5.2. Валидация	126
5.2.1. Невалидные данные	127
5.2.2. Цикл «красный, зеленый, рефакторинг»	129
5.2.3. Натуральные числа	132
5.2.4. Закон Постела (принцип надежности).....	136
5.3. Защита инвариантов	139
5.3.1. Постоянная валидность.....	140
5.4. Заключение	143
Глава 6. Триангуляция	144
6.1. Кратковременная и долговременная память	145
6.1.1. Легаси-код и память	146
6.2. Объем памяти	147
6.2.1. Переполнение	148
6.2.2. Метод «Адвокат дьявола».....	152
6.2.3. Существующее резервирование	155
6.2.4. Метод «Адвокат дьявола» и цикл «красный, зеленый, рефакторинг».....	157
6.2.5. Когда тестов будет достаточно?	160
6.3. Заключение	161

Глава 7. Декомпозиция	163
7.1. Деградация кода	163
7.1.1. Пороговые значения	164
7.1.2. Цикломатическая сложность	166
7.1.3. Правило 80/24	168
7.2. Код, который уместается в вашей голове	170
7.2.1. Гексагональные цветки	170
7.2.2. Связность	173
7.2.3. «Завистливые функции»	177
7.2.4. Потери при передаче	179
7.2.5. Анализ вместо валидации	180
7.2.6. Фрактальная архитектура	183
7.2.7. Подсчет переменных	188
7.3. Заключение	189
Глава 8. Проектирование API	191
8.1. Принципы проектирования API	192
8.1.1. Аффорданс (возможность)	192
8.1.2. Рока-Йоке, или «защита от ошибок»	194
8.1.3. Пишите для читателей	196
8.1.4. Предпочитайте комментариям хорошо написанный код	197
8.1.5. Исключение имен	198
8.1.6. Command Query Separation (CQS), или разделение команд и запросов	201
8.1.7. Иерархия коммуникации	204
8.2. Проектирование API: примеры	205
8.2.1. Класс MaitreD (метрдотель)	206
8.2.2. Взаимодействие с инкапсулированным объектом	209
8.2.3. Детали реализации	212
8.3. Заключение	214

Глава 9. Командная работа	216
9.1. Git	217
9.1.1. Сообщение коммита	218
9.1.2. Непрерывная интеграция	221
9.1.3. Малые коммиты	224
9.2. Коллективное владение кодом	227
9.2.1. Парное программирование	230
9.2.2. Моб-программирование	231
9.2.3. Задержка код-ревью	232
9.2.4. Отклонение набора изменений	235
9.2.5. Код-ревью	236
9.2.6. Пул-реквесты	238
9.3. Заключение	240

ЧАСТЬ II УСТОЙЧИВОСТЬ

Глава 10. Расширение кодовой базы	242
10.1. Функциональные флаги	243
10.1.1. Календарь	244
10.2. Паттерн Strangler («Душитель»)	249
10.2.1. Паттерн Strangler. Уровень метода	251
10.2.2. Паттерн Strangler. Уровень класса	255
10.3. Версионирование	259
10.3.1. Заблаговременное предупреждение	260
10.4. Заключение	261
Глава 11. Редактирование модульных тестов	262
11.1. Рефакторинг модульных тестов	262
11.1.1. Смена подушки безопасности	263

11.1.2. Добавление нового тестового кода.	264
11.1.3. Разделяйте рефакторинг тестового и продакшен-кода.	267
11.2. Непройдённые тесты.	273
11.3. Заключение.	273
Глава 12. Устранение неполадок.	275
12.1. Понимание.	275
12.1.1. Научный подход.	276
12.1.2. Упрощение.	277
12.1.3. Метод утенка.	278
12.2. Дефекты.	280
12.2.1. Воспроизведение дефектов в виде тестов.	281
12.2.2. Медленные тесты.	284
12.2.3. Недетерминированные дефекты.	287
12.3. Метод бисекции.	292
12.3.1. Метод бисекции с Git.	292
12.4. Заключение.	297
Глава 13. Разделение ответственности.	299
13.1. Композиция.	300
13.1.1. Вложенная композиция.	301
13.1.2. Последовательная композиция.	304
13.1.3. Ссылочная прозрачность.	306
13.2. Сквозная функциональность.	310
13.2.1. Логирование.	310
13.2.2. Паттерн проектирования Decorator («Декоратор»).	311
13.2.3. Что регистрировать.	315
13.3. Заключение.	317

Глава 14. Организация рабочего процесса	319
14.1. Индивидуальный процесс работы	320
14.1.1. Тайм-боксинг	320
14.1.2. Делайте перерывы	322
14.1.3. Используйте время разумно	323
14.1.4. Метод слепой печати	325
14.2. Рабочий процесс в команде	326
14.2.1. Регулярное обновление зависимостей	326
14.2.2. Планирование других действий	328
14.2.3. Закон Конвея	329
14.3. Заключение	330
Глава 15. Очевидные аспекты	331
15.1. Производительность	332
15.1.1. Устаревшие знания	332
15.1.2. Удобочитаемость	334
15.2. Безопасность	337
15.2.1. Модель угроз STRIDE	337
15.2.2. Спуфинг	338
15.2.3. Незаконное изменение	339
15.2.4. Отказ от авторства	340
15.2.5. Раскрытие информации	341
15.2.6. Отказ в обслуживании	343
15.2.7. Повышение привилегий	344
15.3. Прочие техники	345
15.3.1. Тестирование на основе свойств	345
15.3.2. Поведенческий анализ кода	351
15.4. Заключение	354

Глава 16. Краткий обзор	356
16.1. Навигация	356
16.1.1. Общее представление	358
16.1.2. Организация файлов	361
16.1.3. Поиск деталей	364
16.2. Архитектура	366
16.2.1. Монолитная архитектура	366
16.2.2. Циклы	367
16.3. Использование	371
16.3.1. Обучение на тестах	372
16.3.2. Прислушивайтесь к своим тестам	374
16.4. Заключение	375
Приложение. Перечень методов	377
П.1. Правило 50/72	377
П.2. Правило 80/24	378
П.3. Шаблон Arrange-Act-Assert (AAA)	378
П.4. Бисекция	378
П.5. Чек-лист для новой кодовой базы	379
П.6. Разделение команд и запросов (CQS)	379
П.7. Подсчет переменных	379
П.8. Цикломатическая сложность	380
П.9. Паттерн проектирования Decorator для сквозной функциональности	380
П.10. Метод «Адвокат дьявола»	380
П.11. Функциональный флаг	381
П.12. Функциональное ядро, императивная оболочка	381
П.13. Иерархия отношений	382
П.14. Обоснование исключений из правил	382

П.15. Анализировать, а не проверять	382
П.16. Закон Постела	383
П.17. Цикл «красный, зеленый, рефакторинг»	383
П.18. Регулярное обновление зависимостей	384
П.19. Воспроизведение дефектов в виде тестов	384
П.20. Код-ревью	384
П.21. Семантическое версионирование	385
П.22. Раздельный рефакторинг тестового и продакшен-кода	385
П.23. Срез	385
П.24. Паттерн Strangler	386
П.25. Модель угроз STRIDE	386
П.26. Предпосылки приоритета трансформации (TPP)	387
П.27. X-ориентированная разработка	387
П.28. Исключение имен	388
Библиография	389