

УДК 004.457
ББК 32.972.13
В53

Виссер Дж.
В53 Разработка обслуживаемых программ на языке C# / пер. с англ.
Р. Н. Рагимова. – М.: ДМК Пресс, 2017. – 192 с.: ил.

ISBN 978-5-97060-446-5

Данное практическое руководство познакомит вас с 10 простыми рекомендациями, помогающими писать программное обеспечение, которое легко поддерживать и адаптировать. Эти тезисы сформулированы на основании анализа сотен реальных систем.

Написанная консультантами компании Software Improvement Group книга содержит ясные и краткие советы по применению рекомендаций на практике. Примеры для этого издания написаны на языке C#, но существует аналогичная книга с примерами на языке Java.

Издание предназначено программистам на C#, желающим научиться писать качественный и хорошо поддерживаемый код.

УДК 004.457
ББК 32.972.13

Authorized Russian translation of the English edition of 'Building Maintainable Software, C# Edition'.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls 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.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-491-95452-2 (анг.) © 2016 Software Improvement Group, B.V.
ISBN 978-5-97060-446-5 (рус.) © Оформление, издание, перевод, ДМК
Пресс, 2017

Содержание

Об авторах	11
Предисловие	13
Глава 1. Введение	26
1.1. Что такое обслуживаемость?	27
Четыре вида обслуживаемости программного обеспечения	27
1.2. Почему так важна обслуживаемость?	28
Обслуживаемость значительно влияет на деловую сторону	
вопроса.....	28
Обслуживаемость обеспечивает улучшение других	
качественных характеристик	29
1.3. Три принципа, на которых основаны рекомендации.....	30
Принцип 1: рекомендации должны быть простыми.....	30
Принцип 2: применение рекомендаций с самого начала	
и значимость вклада каждого разработчика.....	31
Принцип 3: не все отступления от рекомендаций дают	
одинаковый отрицательный эффект.....	31
1.4. Заблуждения относительно обслуживаемости	32
Заблуждение: обслуживаемость зависит от языка	
программирования	32
Заблуждение: обслуживаемость зависит от прикладной области.....	33
Заблуждение: обслуживаемость гарантирует отсутствие ошибок	33
Заблуждение: обслуживаемость оценивается одной из двух	
альтернатив	34
1.5. Рейтинг обслуживаемости.....	34
1.6. Обзор рекомендаций по улучшению обслуживаемости	36
Глава 2. Пишите короткие блоки кода	38
2.1. Мотивация.....	41
Короткие блоки кода проще тестировать	41
Короткие блоки кода проще анализировать.....	42
Короткие блоки кода проще повторно использовать.....	42
2.2. Как применять рекомендацию	42
При написании нового блока кода.....	43
При добавлении в блок новых функциональных возможностей.....	44

Два метода рефакторинга для приведения кода в соответствие с рекомендацией	45
2.3. Типичные возражения против коротких блоков кода	51
Возражение: увеличение количества блоков кода плохо сказывается на производительности	51
Возражение: разделение кода ухудшает читаемость	51
Рекомендация препятствует надлежащему форматированию кода	52
Этот блок кода невозможно разделить	53
Разделение блоков кода не дает заметных преимуществ	54
2.4. Дополнительные сведения	55
Глава 3. Пишите простые блоки кода	57
3.1. Мотивация	64
Простые блоки проще изменять	64
Простые блоки проще тестировать	64
3.2. Как применять рекомендацию	64
Цепочки условий	65
Вложенность	67
3.3. Типичные возражения против создания простых блоков кода	70
Возражение: высокая сложность неизбежна	70
Возражение: разделение методов не уменьшает сложности	70
3.4. Дополнительные сведения	71
Глава 4. Не повторяйте один и тот же код	73
Виды дублирования	76
4.1. Мотивация	78
Код с дубликатами сложнее анализировать	78
В дублированный код сложно вносить изменения	78
4.2. Как применять рекомендацию	78
Извлечение суперкласса	81
4.3. Типичные возражения против исключения дублирования	84
Копирование фрагментов из другой базы кода допустимо	84
При незначительных изменениях дублирование неизбежно	85
Этот код никогда не изменится	85
Дублирование всех файлов допустимо в целях создания их резервных копий	86
Модульные тесты защитят меня	86
Дублирование строковых литералов неизбежно и совершенно безвредно	87
4.4. Дополнительные сведения	87

Глава 5. Стремитесь к уменьшению размеров интерфейсов	90
5.1. Мотивация	92
Интерфейсы небольшого размера упрощают понимание и повторное использование кода	93
В методы с компактным интерфейсом проще вносить изменения	93
5.2. Как применять рекомендацию	93
5.3. Типичные возражения против сокращения размеров интерфейсов	98
Возражение: объекты параметров требуют определения конструкторов с большим количеством параметров	99
Преобразование интерфейсов большого размера не улучшает ситуацию	99
Фреймворки или библиотеки предоставляют интерфейсы с длинными списками параметров	99
5.4. Дополнительные сведения	100
Глава 6. Разделяйте задачи на модули	102
6.1. Мотивация	107
Небольшие слабо связанные модули позволяют разработчикам иметь дело с надежно изолированными частями системы	108
Небольшие слабо связанные модули упрощают навигацию по коду	108
Небольшие слабо связанные модули делают все области кода более понятными новым разработчикам	108
6.2. Как применять рекомендацию	109
Разделение классов по решаемым задачам	109
Соккрытие подробностей реализации за интерфейсами	110
Замена пользовательского кода библиотеками или фреймворками от сторонних производителей	114
6.3. Типичные возражения против разделения задач	114
Возражение: слабые связи вступают в конфликт с возможностью повторного использования	114
Возражение: интерфейсы языка C# не предназначены для ослабления связей	115
Возражение: высокая нагрузка на служебные классы неизбежна	115
Возражение: не все слабо связанные решения улучшают обслуживаемость	115

Глава 7. Избегайте тесных связей между элементами архитектуры.....	118
7.1. Мотивация.....	119
Слабая зависимость между компонентами обеспечивает изолированность его обслуживания	122
Слабая зависимость компонентов способствует разделению ответственности за обслуживание	122
Слабая зависимость компонентов упрощает тестирование.....	123
7.2. Как применять рекомендацию	123
Абстрактная фабрика как шаблон проектирования	124
7.3. Типичные возражения против устранения тесных связей компонентов.....	126
Возражение: зависимости между компонентами невозможно смягчить из-за тесного переплетения компонентов	126
Возражение: нет времени на исправление	127
Возражение: транзитный код просто необходим.....	127
7.4. Дополнительные сведения	128

Глава 8. Стремитесь к сбалансированности архитектуры компонентов.....	130
8.1. Мотивация.....	132
Хороший баланс компонентов упрощает поиск и анализ кода	132
Хорошо сбалансированные компоненты улучшают изолированность обслуживания	132
Хорошо сбалансированные компоненты позволяют распределять ответственность при обслуживании.....	133
8.2. Как применять рекомендацию	133
Выбор правильного концептуального уровня при распределении функциональных возможностей по компонентам	133
Последовательное применение разделения системы на основе анализа предметной области.....	134
8.3. Типичные возражения против стремления к сбалансированности компонентов	135
Возражение: системы с дисбалансом компонентов отлично работают	135
Возражение: запутанность связей между компонентами не позволяет их сбалансировать.....	135
8.4. Дополнительные сведения	136

Глава 9. Следите за размером базы кода	138
9.1. Мотивация.....	139
Проект с большой базой кода, скорее всего, обречен на неудачу.....	139

Большие базы кода труднее обслуживать.....	139
Большие системы отличаются высокой плотностью дефектов.....	141
9.2. Как применять рекомендацию	142
Функциональные меры.....	142
Технические меры.....	142
9.3. Типичные возражения против уменьшения размеров базы кода	144
Возражение: сокращение размера базы кода снижает	
продуктивность разработки	145
Возражение: выбранный язык программирования препятствует	
уменьшению объема кода	145
Возражение: сложность системы заставляет дублировать код.....	146
Возражение: разделение базы кода невозможно	
из-за архитектуры платформы.....	146
Возражение: разделение кода приводит к дублированию.....	147
Возражение: разделение базы кода невозможно из-за тесной	
связанности	147
Глава 10. Автоматизируйте тестирование	149
10.1. Мотивация.....	151
Автоматизация делает тестирование повторяемым	151
Автоматизированное тестирование увеличивает	
эффективность разработки.....	151
Автоматизированное тестирование делает код предсказуемым	151
Тесты документируют тестируемый код	152
Разработка тестов улучшает качество кода	152
10.2. Как применять рекомендацию	153
Начало работы с NUnit	154
Общие принципы разработки хороших модульных тестов	157
Оценка охвата для определения достаточности количества	
тестов.....	162
10.3. Типичные возражения против автоматизации тестов.....	164
Возражение: нам нужно и ручное тестирование.....	164
Возражение: мне не разрешается писать модульные тесты	164
Возражение: зачем тратить время на модульные тесты	
при низком текущем охвате ими?	165
10.4. Дополнительные сведения	165
Глава 11. Пишите чистый код	167
11.1. Не оставляйте следов	167
11.2. Как применять рекомендацию	168
Правило 1: не оставляйте после себя грязи на уровне блоков	
кода.....	168

10 ❖ Содержание

Правило 2: не оставляйте после себя неудачных комментариев.....	169
Правило 3: не оставляйте после себя закомментированного кода.....	171
Правило 4: не оставляйте после себя неиспользуемого кода	172
Правило 5: не оставляйте после себя длинных идентификаторов	173
Правило 6: не оставляйте после себя таинственных констант	173
Правило 7: не оставляйте после себя плохую обработку исключений.....	175
11.3. Типичные возражения против написания чистого кода	175
Возражение: комментарии являются документацией	175
Возражение: обработка исключений увеличивает объем кода	176
Возражение: почему выбраны именно эти правила?	176
Глава 12. Дальнейшие действия	178
12.1. Применение рекомендаций на практике	178
12.2. Низкоуровневые (блоки кода) рекомендации имеют более высокий приоритет, если они противоречат высокоуровневым (компоненты) рекомендациям	179
12.3. Помните, что учитывается каждое действие	179
12.4. Передовой опыт разработки будет рассмотрен в следующей книге	180
Приложение А. Как в компании SIG оценивается обслуживаемость	181
Предметный указатель	184