O'REILLY®

Разработка обслуживаемых программ на языке Java

Джуст Виссер

ДМК ПРЕСС ИЗДАТЕЛЬСТВО

УДК 004.457
ББК 32.972.13
В53

Виссер Дж.
В53 Разработка обслуживаемых программ на языке Java / пер. с англ. Р. Н. Рагимова. – М.: ДМК Пресс, 2017. – 182 с.: ил.

ISBN 978-5-97060-447-2

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 Java, но существует аналогичная книга с примерами на языке C#.

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

УДК 004.457
ББК 32.972.13

ISBN 978-1-4919-5352-5 (анг.)
ISBN 978-5-97060-447-2 (рус.)

Содержание

Глава 1

Введение

Кто написал этот фрагмент кода??
Я не мог этого сделать!!

Любой программист

Здорово быть разработчиком программного обеспечения. Вам ставят задачи и определяют требования, а вы должны придумать решение и перевести его на понятный компьютеру язык. Это сложная и хорошо вознаграждаемая работа. Но разработка программного обеспечения нередко превращается в весьма кропотливую работу. Если вам регулярно приходится вносить изменения в исходный код, написанный другими (или даже вами), вы уже знаете, что это может буть как очень простым, так и очень сложным занятием. Иногда строки кода, которые необходимо изменить, находятся очень быстро. Изменения полностью изолированы, а тесты подтверждают, что все работает, как нужно. Но бывают ситуации, когда единственным выходом является использование обходных путей, что создает больше проблем, чем решает их.

Простоту или сложность внесения изменений в программную систему обычно называют *обслуживаемостью*. Обслуживаемость программной системы определяется свойствами ее исходного кода. Эта книга посвящена рассмотрению этих свойств и представляет 10 рекомендаций, которые помогут писать легко изменяемый исходный код.

В данной главе мы проясним, что понимается под обслуживаемостью. Затем, обсудим важность обслуживаемости. Это создаст основу для перехода к главной теме книги: как разрабатывать изначально обслуживаемое программное обеспечение. В конце этого введения мы перечислим типичные заблуждения, касающиеся обслуживаемости, и принципы, лежащие в основе 10 предлагаемых в книге рекомендаций.

1.1. Что такое обслуживаемость?

Представьте две различные программные системы, имеющие тождественную функциональность. Для одних и тех же входных данных они вернут одинаковый результат. Одна из этих систем работает быстро, дружественно настроена к пользователю, и вносить изменения в ее исходный код не составляет особого труда. Другая медленная, сложная в использовании, и в ее исходном коде практически невозможно разобраться, уж не говоря о том, чтобы что-то в нем менять. Несмотря на идентичную функциональность обеих систем, их качество явно отличается.

Обслуживаемость (простота внесения изменений в систему) является одной из качественных характеристик программного продукта. Производительность (скорость вычисления результата) – это совершенно иная характеристика.

Международный стандарт ISO/IEC 25010: 2011 (который в этой книге будет упоминаться просто как ISO 25010[1]) различает восемь характеристик программного обеспечения: обслуживаемость, функциональная пригодность, эффективность работы, совместимость, удобство использования, надежность, безопасность и переносимость. Эта книга посвящена исключительно обслуживаемости.

Несмотря на то что стандарт ISO 25010 не определяет способов оценки качества программного обеспечения, это не значит, что его невозможно количественно оценить. В приложении А описан порядок количественной оценки качества программного продукта, принятый в компании Software Improvement Group (SIG), соответствующий стандарту ISO 25010.

Четыре вида обслуживаемости программного обеспечения

Обслуживание программного обеспечения не связано с его поломками или износом. Программное обеспечение не является физической сущностью и, следовательно, не склонно к износу, как это характерно для физического оборудования. Тем не менее большинство программных систем постоянно модернизируется после ввода в эксплуа-

[1] Полное наименование стандарта: International Standard ISO/IEC 25010. Systems and Software Engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. First Edition, 2011-03-01.

тацию. Это и называется обслуживанием программного обеспечения. Можно выделить следующие четыре вида обслуживания программного обеспечения:

- устранение выявленных ошибок (так называемое *корректирующее обслуживание*);
- адаптация системы к изменениям в окружающей среде, где она функционирует, например при обновлении операционной системы или смене технологий (называется *адаптивным обслуживанием*);
- обеспечение изменения требований пользователей системы (и/или других заинтересованных сторон) (это *улучшающее обслуживание*);
- повышение качества или предотвращение будущих ошибок (*профилактическое обслуживание*).

1.2. Почему так важна обслуживаемость?

Как упоминалось выше, обслуживаемость – это лишь одна из восьми характеристик качества программного продукта, определенных в стандарте ISO 25010. Так почему же обслуживаемость так важна, что заслуживает отдельной, книги? На этот вопрос имеются два ответа:

- обслуживаемость или отсутствие таковой оказывает существенное влияние на деловую сторону вопроса;
- обслуживаемость обеспечивает улучшение других характеристик качества.

Оба ответа поясняются в следующих двух разделах.

Обслуживаемость значительно влияет на деловую сторону вопроса

В области разработки программного обеспечения период обслуживания программной системы обычно длится 10 и более лет. Большую часть этого времени постоянно возникают вопросы, которые нужно решать (корректирующее и адаптивное обслуживание), и появляются новые требования, которые следует удовлетворять (улучшающее обслуживание). Эффективность и результативность решения вопросов и реализация усовершенствований волнуют все заинтересованные стороны.

Сложность обслуживания невысока, когда решение вопросов и реализация усовершенствований даются просто и быстро. Кроме того,

если эффективность обслуживания позволяет сократить обслуживающий персонал (число разработчиков), то при этом снижаются затраты на обслуживание. Если число разработчиков не уменьшается, при высокой эффективности обслуживания, у них появляется больше времени на выполнение других задач, таких как создание новых функциональных возможностей. Возможность быстрого усовершенствования означает уменьшение срока выхода на рынок новых продуктов и услуг, поддерживаемых системой. Если решение вопросов и реализация усовершенствований даются с трудом и требуют времени, сроки не выдерживаются, и система может быть признана непригодной к эксплуатации.

Компания SIG собрала эмпирические доказательства, что в системах с обслуживаемостью на уровне выше среднего решение возникающих вопросов и реализация усовершенствований занимают в два раза меньше времени, чем в системах с обслуживаемостью на уровне ниже среднего. Ускорение в два раза считается очень значительным в практике корпоративных систем. Время, необходимое для решения вопросов и внесения улучшений, составляет от нескольких дней до нескольких недель. Важность такого улучшения заключается не в том, что исправляется 5 или 10 ошибок в час, а, например, в том, первой ли компания вышла на рынок с новым продуктом или ее опередили конкуренты на несколько месяцев.

И это именно та разница между обслуживаемостью выше и ниже среднего уровня. Работая в компании SIG, мы сталкивались со вновь созданными системами, обслуживаемость которых была настолько низка, что делала невозможным изменение системы еще до момента ввода ее в эксплуатацию. Изменения вносили ошибок больше, чем исправляли. Разработка – достаточно длительный процесс, в течение которого бизнес-окружение (а следовательно, и требования пользователей) меняется. Необходимо вносить коррективы, которые приводят к появлению новых ошибок. Чаще всего от таких систем отказываются еще до достижения ими версии 1.0.

Обслуживаемость обеспечивает улучшение других качественных характеристик

Еще одна причина, почему обслуживаемость является особым аспектом качества программного обеспечения, заключается в том, что она служит стимулятором для других характеристик качества. Если система обладает высоким уровнем обслуживаемости, это облегчает

проведение работ для улучшения других ее качеств, например исправление ошибок, связанных с безопасностью. В конечном счете оптимизация программной системы требует внесения изменений в ее исходный код, направленных на улучшение производительности, функциональной пригодности, безопасности или любой другой из семи прочих характеристик, определяемых стандартом ISO 25010.

Порой требуются лишь небольшие, локальные изменения. Но иногда изменения влекут коренную перестройку. Любые изменения сводятся к поиску определенного фрагмента кода, анализу, изучению внутренней логики, определению места в автоматизируемом бизнес-процессе, отслеживанию зависимостей относительно других частей кода, тестированию и перемещению дальше по конвейеру разработки. В любом случае, в системе с высоким уровнем обслуживаемости проще вносить изменения, что ускоряет оптимизацию качества системы. Например, легко обслуживаемый код стабильнее, чем необслуживаемый, поскольку внесение в него изменений влечет меньшее количество случайных побочных эффектов, чем в запутанный код, который трудно анализировать и тестировать.

1.3. Три принципа, на которых основаны рекомендации

Если обслуживаемость так важна, тогда как можно ее улучшить? В этой книге представлено 10 рекомендаций, следование которым обеспечит высокую обслуживаемость кода. Они будут представлены и рассмотрены в следующих главах. А в этой мы познакомимся с принципами, лежащими в их основе:

1) рекомендации должны быть простыми;
2) не следует вспоминать об обслуживаемости после окончания разработки, нужно уделять ей должное внимание с самого начала. Здесь важен вклад каждого разработчика;
3) не все отступления от рекомендаций дают одинаковый отрицательный эффект. Чем точнее программная система соответствует рекомендациям, тем выше уровень ее обслуживаемости.

Ниже приведены пояснения к этим принципам.

Принцип 1: рекомендации должны быть простыми

Многие полагают, что для обеспечения обслуживаемости требуется что-то вроде «серебряной пули», то есть какой-то один способ или

принцип, автоматически решающий вопрос обслуживаемости раз и навсегда. Мы придерживаемся противоположной точки зрения: высокий уровень обслуживаемости обеспечивается неуклонным следованием очень простым рекомендациям. Следование рекомендациям гарантирует достаточный, но не максимальный уровень обслуживаемости (что бы под этим не понималось). Приведение исходного кода в соответствие с рекомендациями, повышает его обслуживаемость. Но в какой-то момент, рост уровня обслуживаемости становится все меньше и меньше, а затраты растут все выше и выше.

Принцип 2: применение рекомендаций с самого начала и значимость вклада каждого разработчика

Обслуживаемостью надо заниматься с первых дней работы над проектом. Понятно, что трудно оценить влияние отдельного «нарушения» рекомендаций из этой книги на общую обслуживаемость системы. Именно поэтому все разработчики должны быть дисциплинированными и следовать рекомендациям по улучшению обслуживаемости системы в целом – индивидуальный вклад каждого имеет большое значение.

Следование рекомендациям, приведенным в этой книге, не только поможет писать легко обслуживаемый код, но и послужит достойным примером коллегам-разработчикам. Это позволит избежать «эффекта разбитых окон», возникающего из-за того, что кто-то расслабился и пошел по пути наименьшего сопротивления. Подающий правильный пример не обязательно должен быть самым опытным, скорее самым дисциплинированным.

Не забывайте, что вы пишете код не только для себя, но и для менее опытных разработчиков, которые придут после вас. Эта мысль поможет вам применять простые решения при программировании.

Принцип 3: не все отступления от рекомендаций дают одинаковый отрицательный эффект

Изложенные в этой книге рекомендации представляют измеримые пороговые значения в форме абсолютных правил. Например, в главе 2 рекомендуется никогда не писать методы, содержащие более 15 строк кода. Мы прекрасно понимаем, что на практике всегда присутствуют исключения из основного правила. Что, если фрагмент исходного кода нарушает одну или несколько из этих рекомендаций? Многие виды инструментов оценки качества программного обеспече-

ния предполагают, что никакое нарушение недопустимо. Подразумевается, что все нарушения должны быть устранены. На практике исключение всех нарушений не является необходимостью и часто бесполезно. Подход «все или ничего» по отношению к нарушениям способен лишь заставить разработчиков вообще игнорировать их все.

Рассмотрим другой подход. Чтобы сохранить простоту и практичность оценки, мы определяем качество всей базы кода не по количеству нарушений, а по их *профилю качества*. Профиль качества делит оценки на отдельные категории, начиная от совершенного кода до кода с серьезными нарушениями. Используя профили качества, можно отделить умеренные нарушения (например, метод с 20 строками кода) от серьезных (например, метод с 200 строк кода). После обсуждения в следующем разделе типичных заблуждений, касающихся обслуживаемости, мы поясним порядок использования профилей качества для оценки обслуживаемости систем.

1.4. Заблуждения относительно обслуживаемости

В этом разделе обсуждается несколько заблуждений, касающихся обслуживаемости, часто встречающихся на практике.

Заблуждение: обслуживаемость зависит от языка программирования

«*При разработке системы используется современный язык программирования. Поэтому уровень ее обслуживаемости не может быть хуже, чем у любой другой систем*».

Имеющиеся у компании SIG данные не подтверждают, что применяемая технология (язык программирования) является доминирующим фактором, определяющим обслуживаемость системы. Нам известны примеры систем, написанных на языке Java, как с очень высоким, так и с очень низким уровнем обслуживаемости. В среднем обслуживаемость всех проверенных нами Java-систем совпадает со средним уровнем обслуживаемости, то же самое справедливо и для систем, написанных на языке C#. Это показывает, что на языках Java (или C#) можно создавать отлично обслуживаемые системы, но сам по себе выбор этих языков не гарантирует высокого уровня обслуживаемости. Очевидно, существуют другие факторы, определяющие уровень обслуживаемости.

Для единообразия во всей книге используются фрагменты кода, написанные на языке Java. Однако наши рекомендации применимы не только к языку Java. На основе рекомендаций и показателей, содержащихся в книге, компания SIG провела тестирование систем, написанных на более чем ста языках программирования.

Заблуждение: обслуживаемость зависит от прикладной области

«Моя команда разрабатывает встроенное программное обеспечение для автомобильной промышленности. Здесь обслуживаемость оценивается по-своему».

Мы считаем, что наши рекомендации применимы при разработке всех видов программного обеспечения: встроенного, игрового, научного, таких программных компонентов, как компиляторы и СУБД, а также программного обеспечения для администрирования. Конечно, между этими прикладными областями существуют различия. Например, для создания научного программного обеспечения часто используются языки программирования специального назначения, такие как R, применяемый для статистического анализа. Тем не менее и при использовании языка R имеет смысл делать блоки кода короткими и простыми. Встроенное программное обеспечение должно работать в условиях, где существенное значение имеет предсказуемая производительность, а ресурсы ограничены. Поэтому всякий раз, когда приходится выбирать между производительностью и обслуживаемостью, предпочтение отдается первому в ущерб последнему. Но и в этой прикладной области применимы характеристики стандарта ISO 25010.

Заблуждение: обслуживаемость гарантирует отсутствие ошибок

«Вы утверждаете, что система имеет уровень обслуживаемости выше среднего. Но оказалось, что в ней полно ошибок!»

В соответствии с определениями стандарта ISO 25010 система может иметь высокий уровень обслуживаемости, но при этом у нее могут быть низкими другие характеристики качества. То есть система, обладающая уровнем обслуживаемости выше среднего, может вместе с тем страдать от проблем, связанных с функциональной пригодностью, производительностью, надежностью, и многих других. Высокий уровень обслуживаемости означает лишь, что внесение из-

менений для уменьшения количества ошибок может быть проделано с высокой степенью эффективности и результативности.

Заблуждение: обслуживаемость оценивается одной из двух альтернатив

«*Моя команда неоднократно исправляла ошибки в системе. Следовательно, она является обслуживаемой*».

Важное отличие. Под термином «обслуживаемость» понимается простота обслуживания. Согласно определению в стандарте ISO 25010, обслуживаемость исходного кода не характеризуется одним из двух альтернативных значений. Напротив, она оценивается степенью эффективности и результативности внесения изменений. Поэтому правильной постановкой вопроса является не «какие были сделаны изменения (например, исправлены ошибки)», а «сколько усилий было затрачено на исправление ошибок (эффективность) и были ли действительно устранены ошибки (результативность)?».

Основываясь на определении обслуживаемости в стандарте ISO 25010, можно утверждать, что программные системы никогда не бывают ни максимально обслуживаемыми, ни совершенно не обслуживаемыми. В компании SIG мы сталкивались с системами, которые можно считать практически не обслуживаемыми. Эти системы имеют такую низкую степень эффективности и результативности внесения изменений, что их владельцы не могли позволить себе продолжать их эксплуатацию.

1.5. Рейтинг обслуживаемости

Теперь мы знаем, что обслуживаемость является качественной характеристикой, определяемой по шкале. Она обозначает степень возможности обслуживать систему. Но как определить простоту или сложность обслуживания? Очевидно, что сложную систему проще будет обслуживать эксперту, чем менее опытному разработчику. В компании SIG при ответе на этот вопрос учитываются параметры эталонных систем индустрии разработки программного обеспечения. Если параметры программной системы ниже, чем средние для эталонных систем, ее трудно обслуживать. Калибровка параметров эталонных систем производится один раз в год. По мере того как в индустрии растет эффективность написания кода (например, при применении новых технологий), среднее значение параметров эталонных систем

увеличивается. То, что было нормой в программной инженерии несколько лет назад, не применимо в настоящее время. Таким образом, показатели эталонных систем отражают состояние улучшения технологий в области программной инженерии.

В компании SIG для обозначения рейтинга систем используются звезды, от 1 (сложное обслуживание) до 5 (простое обслуживание). Распределение систем по этим рейтингам от 1 до 5 звезд составляет 5% – 30% – 30% – 30%– 5%. То есть системам, попавшим в число 5% лучших, присваивается 5 звезд. Безусловно, в этих системах имеются нарушения рекомендаций, но их гораздо меньше, чем в системах с более низким рейтингом.

Рейтинги в виде звезд обеспечивают предсказание реального уровня обслуживаемости. В компании SIG было получено эмпирическое доказательство, что в системах с 4 звездами решение проблем и внесение усовершенствований осуществляются в два раза быстрее, чем в системах с 2 звездами.

Эталонные системы оцениваются на основании характеризующих их профилей качества. На рис. 1.1 приведены три примера профилей качества, основанных на оценке размеров блоков кода (читатель может найти цветные рисунки этого и других профилей качества в репозитории с файлами к этой книге: https://github.com/oreillymedia/building_maintainable_software).

Рис. 1.1 ❖ Пример трех профилей качества

На первой диаграмме показан профиль качества по размерам блоков кода популярного сервера непрерывной интеграции с открытым исходным кодом Jenkins версии 1.625. Профиль качества показывает, что 64% всего кода сервера Jenkins находится в методах длиной не более 15 строк (соответствует рекомендации). Профиль также сообщает, что 18% кода включено в методы длиной от 16 до 30 строк