УДК 004.55

**Е. В. СОКОЛОВА**

***Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Украина***

# ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ МЕТОДОВ РЕФАКТОРИНГА

*Рассмотрен процесс рефакторинга программ с целью повышения качества программного обеспечения. Отражены основные показатели качества ПО и определены метрики оценки качества исходного кода. Сформулированы цели и задачи проведения исследований. Предложена методика обработки результатов исследований. Выбран инструмент исследований – система контроля версий с открытым исходным кодом TortoiseSVN. Приведены результаты экспериментальных исследований эффективности применения методов рефакторинга проекта с открытым кодом AppMetrics. Доказано, что наиболее распространенными методами рефакторинга являются: добавление параметра, удаление параметра, встраивание метода, выделение метода.*

***Ключевые слова:*** *программное обеспечение, рефакторинг, метрики исходного кода.*

## Введение

В настоящее время актуальной является проблема разработки качественного программного обеспечения (ПО). При разработке такого ПО важно учитывать различные показатели качества, одним из которых является корректность – соответствие реализации программы заданной спецификации. В современных проектах по разработке ПО изменение спецификации даже внутри цикла разработки – обычное явление, вызванное уточнением понимания предметной области и изменением внешних условий. Другой особенностью современных программных проектов является регулярная перестройка исходного кода программы для облегчения поддержки изменений в спецификации и улучшения читаемости. Такая перестройка, в которой сохраняется поведение программы, называется рефакторингом [1].

Рефакторинг – изменение внутренней структуры системы без изменения существующей функциональности с целью упрощения понимания и внесения дальнейших изменений. Это широко известный способ повышения качества программного обеспечения.

Рефакторинг применяется постоянно при разработке кода. Основные причины его проведения: добавление новой функции, которая недостаточно укладывается в принятое архитектурное решение; исправление ошибки, причины возникновения которой сразу не ясны; преодоление трудностей в командной разработке, которые обусловлены сложной логикой программы.

Во многом при рефакторинге полагаются на интуицию, основанную на опыте. Тем не менее, выделяют проблемы в коде, требующие рефакторинга: дублирование кода, длинный метод, большой класс, длинный список параметров, «завистливые» функции (метод, который чрезмерно обращается к данным другого объекта), избыточные временные переменные, классы данных, несгруппированные данные.

Рефакторинг изначально не предназначен для исправления ошибок и добавления новой функциональности, он вообще не меняет поведение ПО и это помогает избежать ошибок и облегчить добавление функциональности. Он выполняется для улучшения понятности кода или изменения его структуры, для удаления «мёртвого кода» – всё это для того, чтобы в будущем код было легче поддерживать и развивать. В частности, добавление в программу нового поведения может оказаться сложным для существующей структуры – в этом случае разработчик может выполнить необходимый рефакторинг, а уже затем добавить новую функциональность.

Например, это может быть перемещение поля из одного класса в другой, вынесение фрагмента кода из метода и превращение его в самостоятельный метод или даже перемещение кода по иерархии классов. Каждый отдельный шаг может показаться элементарным, но совокупный эффект таких малых изменений в состоянии радикально улучшить проект или даже предотвратить распад плохо спроектированной программы.

Исходя из этого, можно утверждать, что исследования, направленные на сравнительный анализ эффективности методов рефакторинга проектов с открытым кодом актуальны.

Основная цель работы – экспериментальный

анализ эффективности применения методов рефакторинга проектов с открытым кодом с целью повышения качества ПО. Для достижения этой цели необходимо решить следующие задачи:

– провести обзор и проанализировать проблемы и методы рефакторинга проектов с открытым кодом;

– определить факторы и отклики эксперимента;

– проанализировать результаты экспериментальных исследований эффективности применения методов рефакторинга проектов с открытым кодом.

## Метрики оценки качества ПО

Качество программного обеспечения определяется в стандарте ISO 9126 как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц [2].

Классическое определение качества заключается в том, что разработанный программный продукт подтверждает свою спецификацию, при этом спецификация должна быть ориентирована на характеристики, которые желает получить клиент.

Современные стандарты уточняют понятие качества, вводя совокупность черт и характеристик, которые влияют на его способность удовлетворять заданным потребностям пользователей.

Выделяют характеристики качества.

1. *Функциональность* – это способность программного продукта выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей. Набор таких функций определяется во внешнем описании программного продукта.

2. *Удобство* – это характеристики программного продукта, которые позволяют минимизировать усилия пользователя по подготовке исходных данных, применению программного продукта и оценке полученных результатов, а также вызывать положительные эмоции определенного или подразумеваемого пользователя.

3. *Эффективность* – это отношение уровня услуг, предоставляемых программным продуктом пользователю при заданных условиях, к объему используемых ресурсов.

4. *Сопровождаемость* – это характеристики программного продукта, позволяющие минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации в соответствии с изменяющимися потребностями пользователей.

5. *Переносимость* – это способность программного продукта быть перенесенным из одной среды в другую, в частности, с одной аппаратной архитектуры на другую.

Различают понятия: внутреннего качества, связанного с характеристиками ПО самого по себе, без учета его поведения; внешнего качества, характеризующего ПО с точки зрения его поведения; качества ПО при использовании в различных контекстах того качества, которое ощущается пользователями при конкретных сценариях работы ПО. Взаимоотношения между этими аспектами качества показаны на схеме (рис. 1), принятой ISO 9126.

Для показателей качества введены метрики, позволяющие оценить их. Метрики сложности программ принято разделять на три основные группы: метрики размера программ; метрики сложности потока управления программ; метрики сложности потока данных программ.

Откликами эксперимента являются метрики сложности ПО, такие как: цикломатическая сложность (cyclomatic complexity), показывающая структурную сложность кода, т. е. количество различных ветвей в коде; глубина наследования (depth of inheritance); связность классов (class coupling); количество строк кода (lines of code); комплексный показатель качества кода (maintainability index) [3].

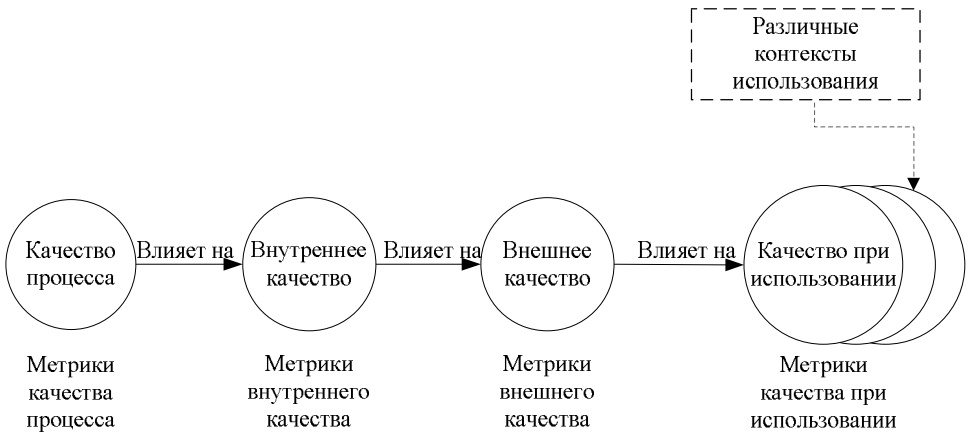


Рис. 1. Основные аспекты качества ПО

Для анализа исходного кода использовалась система контроля версий TortoiseSVN, которая позволяет управлять файлами и каталогами, и сделанными в них изменениями во времени. Это позволяет восстановить более ранние версии данных и изучить историю всех изменений. Расчет метрик проводился с помощью Microsoft Visual Studio 2012 (рис. 2).

Для проведения эксперимента выбран проект с открытым кодом AppMetrics. Проект имеет большую историю изменений (на протяжении двух лет). Над ним проводилось множество рефакторингов, поэтому он подходит для проведения эксперимента.

## Экспериментальные исследования

Для выбора версии ПО, сравнения измененных классов проекта и анализа примененных методов рефакторинга используем TortoiseSVN.

В результате анализа 11 версий ПО AppMetrics установлено, что наиболее распространенными методами рефакторинга являются: добавление параметра, удаление параметра, встраивание метода, выделение метода. В итоге получены значения метрик сложности, представленные в таблице 1.

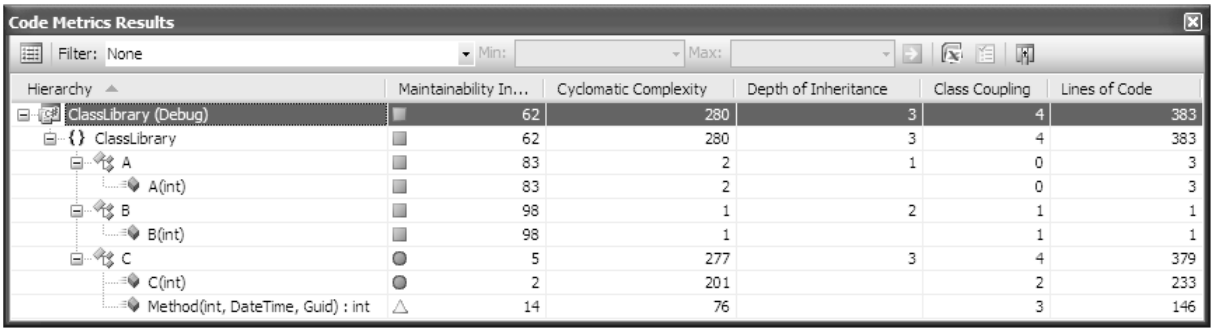


Рис. 2. Пример расчета метрик в Microsoft Visual Studio 2012

Таблица 1

Значения метрик сложности проекта «AppMetrics»

| № Версии ПО | **Метрика сложности** | | | | |
|---|---|---|---|---|---|
| | Цикломатическая сложность CC | Глубина наследования | Связность классов | Количество строк кода LoC | Комплексный показатель качества кода |
| 1 | 194 | 2 | 66 | 4100 | 75 |
| 2 | 196 | 3 | 66 | 4280 | 76 |
| 3 | 197 | 3 | 67 | 4340 | 76 |
| 4 | 196 | 3 | 67 | 4420 | 77 |
| 5 | 196 | 3 | 69 | 4570 | 77 |
| 6 | 198 | 2 | 68 | 4600 | 78 |
| 7 | 196 | 3 | 69 | 4650 | 78 |
| 8 | 197 | 4 | 71 | 4730 | 78 |
| 9 | 199 | 4 | 71 | 4750 | 78 |
| 10 | 200 | 4 | 72 | 4770 | 78 |
| 11 | 201 | 4 | 72 | 4800 | 79 |

Метрика сложности «комплексный показатель качества кода»[4] MI принимает значения от 0 до 100 и показывает относительную сложность поддержки кода:

$$MI = MAX(0,172 - 5,2\ln(HV) - \\ - 0,23CC - 16,2\ln(LoC) * 100 / 171),$$

где HV – вычислительная сложность. CC – цикломатическая сложность, LoC – количество строк кода.

Чем больше операторов, тем больше значение этой метрики, и тем легче поддерживать код.

В результате анализа значений комплексного показателя качества кода построена гистограмма (рис. 3). Анализ метрики показал, что с ростом версии этот показатель увеличивается, следовательно, код становится более поддерживаемый. В то же время количество различных ветвей в коде возрастает и для полного покрытия кода должно увеличиваться количество тестов.


Рис. 3. Гистограмма метрики MI

## Выводы

В связи с требованиями постоянных изменений и с учетом сложности разработки качественного ПО необходимость постоянного проведения рефакторинга очевидна. В то же время при рефакторинге полагаются на опыт и интуицию, не проводя оценку эффективности применения того или иного метода.

Для оценки эффективности применения методов рефакторинга предложено использовать метрики сложности проекта, что позволило оценить код ПО с различных точек зрения. Перспективным направлением данной работы является разработка алгоритма динамического выбора применяемого метода рефакторинга.

## Литература

*1. Фаулер, М. Рефакторинг: улучшение существующего кода [Текст] : пер. с англ. / М. Фаулер. – СПб. : Символ плюс, 2003. – 432 с.*

*2. Богданов, Д. В. Стандартизация жизненного цикла и качества программных средств [Текст] : учеб. пособие / Д. В. Богданов, В. В. Фильчаков. – СПб. : ГУАП, 2000. – 210 с.*

*3. Kannangara, S. H. An Empirical Evaluation of Impact of Refactoring On Internal and External Measures of Code Quality [Text] / S. H. Kannangara, W. M. J. I. Wijayanake // International Journal of Software Engineering & Applications. – 2015. – V. 1, № 6. – P. 51-67.*

*4. Oman, P. Metrics for assessing a software system's maintainability [Text] / P. Oman, J. Hagemeister // Software Maintenance : Proceerdings, Conference on 9-12 Nov. 1992. – P. 337-344.*

*Поступила в редакцию 6.03.2015, рассмотрена на редколлегии 20.03.2015*

**ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ МЕТОДІВ РЕФАКТОРІНГУ**

***Є. В. Соколова***

Розглянуто процес рефакторінгу програм з метою підвищення якості програмного забезпечення. Відображено основні показники якості ПЗ та визначено метрики оцінки якості вихідного коду. Сформульовано мету і задачі проведення досліджень. Визначено методику обробки результатів досліджень. Вибрано інструмент досліджень - система контролю версій з відкритим кодом TortoiseSVN. Наведено результати експериментальних досліджень ефективності застосування методів рефакторінга проекту з відкритим кодом AppMetrics. Доведено, що найбільш поширеними методами рефакторінга є: додавання параметра, видалення параметра, вбудовування методу, виділення методу.

**Ключові слова:** програмне забезпечення, рефакторінг, метрики вихідного коду.

**EXPERIMENTAL RESEARCH USING AN EFFICIENCY REFACTORINGS**

***E. V. Sokolova***

The process of the refactoring programs to improve the software quality have been considered. The main software quality performances are reflected and exit code qualities assessment are defined. The research problems and objectives are formulated. The processing technique and the research results are defined. The research tool, i.e. version control system with open source TortoiseSVN have been chosen. The refactoring methods efficiency of open source project AppMetricshave experience results have been shown. It was proved that the most common methods of the refactoring are adding a parameter, removal of parameters, embedding method, the allocation method.

**Key words:** software, refactoring, source code metrics.

**Соколова Евгения Витальевна** – канд. тех. наук, доцент каф. инженерии программного обеспечения, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: EVS_khai@gmail.com.