*На правах рукописи*

**КНЯЗЕВ Евгений Геннадьевич**

# Автоматизированная классификация изменений исходного кода на основе кластеризации метрик в процессе разработки программного обеспечения

Специальность 05.13.11. Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург
2009

Работа выполнена в Санкт-Петербургском государственном университете информационных технологий, механики и оптики (СПбГУ ИТМО)

Научный руководитель: доктор технических наук, профессор
Шалыто Анатолий Абрамович

Официальные оппоненты: доктор технических наук, профессор
Воробьев Владимир Иванович

кандидат физ.-мат. наук, доцент
Новиков Федор Александрович

Ведущая организация: Санкт-Петербургский
государственный университет
аэрокосмического приборостроения

Защита диссертации состоится 24 декабря 2009 года в 15 часов 30 минут на заседании диссертационного совета Д212.227.06 в Санкт-Петербургском государственном университете информационных технологий, механики и оптики, 197101, Санкт-Петербург, Кронверкский проспект, д. 49.

С диссертацией можно ознакомиться в библиотеке СПбГУ ИТМО.

Автореферат разослан 23 ноября 2009 г.

Ученый секретарь
диссертационного совета
доктор технических наук, доцент Л. С. Лисицына

## Общая характеристика работы

**Актуальность проблемы**

Современные организации-разработчики программного обеспечения работают с очень большим объемом исходного кода, что усложняет его понимание и анализ, а, как следствие, затрудняет контроль его качества. В процессе контроля качества программного обеспечения важную роль имеет экспертиза исходного кода (*code review*).

В ходе экспертизы просматривается код с целью обнаружения таких недостатков как, например, алгоритмические и архитектурные ошибки, нарушение принятого стиля кодирования, неясное назначение фрагментов кода. Кроме того, эксперт обычно осуществляет поиск неиспользуемого кода, отслеживает внесение избыточных или несвоевременных изменений, а также внесение изменений, потенциально способных нарушить работоспособность системы, усложнить ее дальнейшее развитие. Экспертиза исходного кода позволяет на ранних стадиях разработки обнаруживать ошибки, которые иначе были бы найдены только на этапе тестирования. Применение экспертизы исходного кода на практике обычно требует существенных временных затрат.

Для упрощения экспертизы кода часто ограничиваются только экспертизой его изменений, так как разработка кода обычно происходит итеративным путем и сводится к внесению изменений (включая новую функциональность). Использование информации о модификациях исходного кода упрощает его понимание за счет концентрации внимания эксперта. Благодаря повсеместному использованию систем контроля версий, при разработке большинства программ доступна история изменений. Однако экспертиза изменений обычно затруднительна из-за их большого числа и ограничения на время работы эксперта. Поэтому приходится проводить выборочную экспертизу изменений. Критерием выбора изменений может быть принадлежность к некоторому классу.

Целесообразно выделять классы изменений, такие как, например, реализация новой функциональности, удаление неиспользуемого кода, рефакторинг, исправление логики, форматирование кода. Классификация изменений выполняется не только для понимания исходного кода, как отмечено выше, но и для оценки качества изменений по коду, а не с помощью тестирования.

Классификация требуется для контроля процесса разработки, так как, например, если продукт стабилизирован, то никакие изменения, кроме исправления ошибок, проводить не следует. Классификация позволяет также автоматизировать передачу информации между участниками процесса разработки. Она используется для того, чтобы сформировать список изменений, которые требуется проверять, а при необходимости описывать.

В работе предлагается автоматизированный метод классификации изменений исходного кода, состоящий из двух шагов – кластеризации и сопоставления кластеров классам. Распределение изменений по кластерам осуществляется автоматически. Сопоставление их классам выполняет эксперт. Автоматизация

распределения изменений по кластерам существенно сокращает время экспертизы изменений кода.

Задача автоматизации классификации изменений исходного кода решалась многими исследователями: *A. Hassan, R. Holt, S. Demeyer, S. Ducasse, S. Raghavan, R. Rohana, J. Maletic, M. Collard, R. Robbes, S. Kim, J. Whitehead* и другими. Ими были разработаны методы классификации изменений на базе *эвристического, синтаксического, метрического* и *Data Mininig* подходов к анализу изменений.

Недостатком *эвристических* методов является их некорректная работа при несоответствии входных данных построенным предположениям. Недостатками *синтаксических* методов являются зависимость от языка программирования и высокая алгоритмическая сложность. Недостатком методов, основанных на *Data Mining,* является сложность создания обучающего множества. Кроме того, большинство перечисленных методов обладает общим недостатком – они предназначены для построения единственной классификации и не допускают настройки на другие классификации.

Указанные недостатки могут быть устранены при совместном использовании метрического подхода и метода кластеризации, который не требует формирования обучающего множества. Поэтому, как следует из изложенного выше, разработка метода автоматизированной классификации изменений с использованием кластеризации и метрического подхода является актуальной задачей.

**Цель диссертационной работы –** разработка метода автоматизированной классификации изменений кода в процессе создания программного обеспечения на основе кластеризации метрик.

**Основные задачи исследования**:

1. Обоснование возможности частичной автоматизации классификации изменений исходного кода методом кластеризации метрик.
2. Разработка метода автоматизированной классификации изменений исходного кода на основе кластеризации метрик изменений.
3. Внедрение результатов работы в практику разработки программного обеспечения.

**Научная новизна.** На защиту выносятся результаты, обладающие научной новизной.

1. Обоснование возможности частичной автоматизации классификации изменений исходного кода методом кластеризации метрик за счет формулировки гипотезы об автоматизированной классификации и ее экспериментального подтверждения.
2. Обоснование выбора метода *k-средних* с *мерой близости* объектов для кластеризации, *основанной на косинусе угла между векторами метрик* изменений.
3. Метод автоматизированной классификации изменений исходного кода на основе кластеризации метрик изменений, позволяющий сократить число изменений для классификации, выполняемой вручную.

Перечисленные результаты получены в ходе выполнения работ в СПбГУ ИТМО и ЗАО «Транзас Технологии» (Санкт-Петербург).

**Методы исследования.** В работе использованы методы кластерного анализа, математической статистики и программной инженерии.

**Достоверность** научных положений, выводов и практических рекомендаций, полученных в диссертации, подтверждается корректным применением методов кластерного анализа и совпадением результатов автоматизированной и экспертной оценки в пределах заданной точности.

**Практическое значение** работы состоит в том, что все полученные результаты используются в настоящее время и будут использоваться в дальнейшем для повышения качества программного обеспечения в ходе разработки сложных программных комплексов. Предложенный подход применялся для классификации изменений исходного кода в продуктах, разрабатываемых ЗАО «Транзас Технологии» (система мониторинга мобильных объектов *Navi-Manager*, набор компонент глобальной системы *LRIT* (*Long-Range Identification and Tracking*) для отслеживания положения судов в мировом океане, система контроля действий студентов на тренажерах *e-Tutor 5000*), а также в двух программных системах с открытым кодом.

**Внедрение результатов.** Результаты, полученные в диссертации, внедрены в указанных системах *Navi-Manager, LRIT, e-Tutor 5000*, а также в учебном процессе на кафедре «Компьютерные технологии» СПбГУ ИТМО по курсу лекций «Современные технологии разработки программного обеспечения».

**Апробация результатов.** Основные положения диссертационной работы докладывались на научно-методической конференции «Телематика-2007» (СПб., 2007), X международной конференции по мягким вычислениям и измерениям (СПб., 2007), на конференции «Software Egineering Conference (Russia) 2007» (М., 2007), XXXVI научной и учебно-методической конференции профессорско-преподавательского и научного состава СПбГУ ИТМО (СПб., 2007), на семинаре Российского Северо-Западного регионального отделения IEEE по компьютерным технологиям и инженерному менеджменту (IEEE Region 8 Russia North-West Computer Society/Engineering Management Society Joint Chapter) (СПб., 2007), IV и V Межвузовской конференции молодых ученых (СПбГУ ИТМО, 2007, 2008), XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке» (СПб., 2008).

**Публикации.** По теме диссертации опубликовано 11 печатных работ, в том числе две статьи в журналах из списка ВАК. Результаты, приводимые в диссертации, опубликованные без соавторов, получены лично автором. В работах под номерами 1 и 7 в списке публикаций автором предложены способы использования автоматизированной классификации изменений. В работе 6 автором предложен способ расчета метрики покрытия изменения кода модульными тестами. В работах 2, 3 и 8 автором предложен метод автоматизированной классификации изменений на основе предложенного автором способа расчета метрик изменений и их кластеризации. Остальные результаты в статьях под номерами 1, 2, 3, 7 и 8 принадлежат соавтору.

**Структура диссертации.** Диссертация изложена на 126 страницах и состоит из введения, трех глав и заключения. Список литературы содержит 95 наименований. Работа иллюстрирована 21 рисунком и содержит 34 таблицы.

## Содержание работы

**Во введении** обосновывается актуальность темы диссертации, описывается предмет исследования, ставятся цель и задачи исследования, формулируются положения, выносимые на защиту.

**1. В первой главе** приведен обзор состояния проблемы классификации изменений исходного кода. Известны следующие методы автоматизации классификации изменений исходного кода: 1. Эвристический метод поиска характерных слов в комментариях к изменениям (*A. Hassan, R. Holt*). 2. Эвристический метод поиска и классификации рефакторингов на основе значений определенных метрик (*S. Demeyer, S. Ducasse* и другие). 3. Метод сравнения синтаксических деревьев версий кода (*S. Raghavan, R. Rohana* и другие). 4. Метод анализа синтаксической разницы версий кода с помощью встраиваемых в код тегов (*J. Maletic, M. Collard*). 5. Метод, основанный на реализации системы контроля версий, которая хранит абстрактные синтаксические деревья кода, полученные на основе данных из среды разработки (*R. Robbes*). 6. Метод классификации изменений по признаку возможного наличия в них ошибки (*S. Kim, J. Whitehead* и другие).

Недостатком первых двух методов является работа с ошибками для некоторых входных данных, так как при разработке эвристик сложно учесть все возможные значения таких данных. Недостатками следующих двух методов является их алгоритмическая сложность, зависимость от языка программирования, а также отсутствие адаптивности. Применение синтаксических методов классификации изменений оправдано только для анализа простых изменений. Недостатком пятого метода является необходимость избыточного хранения и обработки полного набора модификаций в системе контроля версий, проводимых пользователем в среде разработки, так как далеко не все они сохраняются в итоговом коде. Недостатком шестого метода, основанного на подходе *Data Mining*, является необходимость построения обучающего множества, содержащего изменения, порождающие ошибки и их исправляющие. Построение такого множества – нетривиальная задача. Кроме того, большинство указанных методов обладают недостатком – специализированностью.

В настоящей работе предлагается метод классификации изменений исходного кода на основе кластеризации метрик, свободный от указанных выше недостатков, который, в отличие от приведенных методов, можно применять для достаточно широкого круга задач. По сравнению с большинством существующих методов классификации изменений исходного кода, предложенный метод обладает **достоинством** – является *настраиваемым* (набор метрик исходного кода выбирается в зависимости от того, по каким аспектам изменений строится классификация). Сравнение эффективности разработанного в диссертации метода

с другими методами не выполнялось в связи с различием используемых в них классификаций, что делает прямое сравнение результатов классификаций некорректным, а настройка на другие классификации для большинства методов невозможна.

В автореферате приводятся результаты сравнения классификаций изменений в программной системе с открытым исходным кодом, выполненные с использованием предложенного автоматизированного метода и вручную. К участию к эксперименту привлекались независимые эксперты, имеющие опыт разработки сложных программных систем не менее пяти лет. Всего в диссертации проанализировано пять программных систем.

**2. Во второй главе** исследована возможность автоматизации классификации изменений исходного кода методом кластеризации метрик. Обоснован выбор метода *k*-средних с мерой близости объектов для кластеризации, основанной на косинусе угла между векторами метрик изменений. Предложенный метод позволяет улучшать качество классификации за счет возможности увеличения числа используемых метрик. В настоящее время в методе используется 11 метрик, перечень которых приведен в разд. 2.2. Автор предполагает, что возможно совершенствование качества классификации за счет увеличения числа используемых метрик, а также настройка на другие перечни классов изменений, кроме тех, которые рассмотрены в диссертации.

**2.1. Исследование возможности автоматизации классификации изменений исходного кода методом кластеризации метрик**. Рассмотрим некую *программную систему P* в процессе ее развития во времени. Состояние этой системы в каждый момент времени $t$ задается ее текущим кодом $S_t$. Для удобства обозначим множества неизменных состояний $S_t$, в течение последовательных интервалов времени $t \in (t_{r-1},\ldots,t_r)$, через $S_r$, где $r$ – целое число, $1 \le r \le N$, $N$ – общее число различных состояний кода. *Изменением исходного кода* назовем отображение $\delta_r$, переводящее код из предшествующего состояния $S_{r-1}$ в модифицированное состояние $S_r$:

$$S_{r-1} \xrightarrow{\delta_r} S_r.$$

С помощью каждого изменения кода разработчиком достигается определенная цель по развитию программной системы. При реализации программных систем на практике становится ясно, что часто цели, достигаемые с помощью различных изменений, имеют много общего между собой. Для упрощения задач экспертизы кода целесообразно делить изменения на классы в соответствии с выбранными целями.

Каждое изменение $\delta$ в соответствии с целью его внесения может быть отнесено к некоторому классу $c$, где $c \in C$. При этом $C$ представляет собой множество классов изменений, специфичное для разрабатываемой программной системы. Наиболее распространенными являются следующие классы изменений: реализация новой функциональности, исправление логики, рефакторинг, удаление избыточного кода, форматирование кода.

В общем виде способ классификации множества изменений с трудом поддается формализации. Поэтому задача отнесения изменения к тому или иному

классу *c* трудоемка и требует участия эксперта высокой квалификации. Эксперт определяет состав множества классов, специфичный для каждой программной системы, а также решает, к какому из классов относится конкретное изменение.

Следует заметить, что результат экспертной классификации зависит не только от анализируемой программной системы и набора изменений, но и от конкретного эксперта. Поэтому можно ожидать, что при классификации некоторых изменений разными экспертами возможно различное распределение изменений по заданным классам.

Определим здесь проверочное множество $\Delta_e$ как некоторое множество изменений, классифицированное экспертом. Для устранения возможных противоречий в данной работе построение проверочных множеств изменений производится двумя экспертами, и изменения, классифицированные ими различным образом, из проверочного множества исключаются.

Метод кластеризации метрик изменений позволяет нивелировать субъективность классификации, и, как показано в диссертации, может быть обеспечена частичная автоматизация классификации изменений исходного кода. Автоматизация классификации выполняется на основе следующей гипотезы.

***Гипотеза 1***. *Автоматизированная классификация изменений кода некоторой программной системы возможна методом кластеризации метрик.*

*Пусть задано множество классов изменений и требуется классифицировать множество изменений программной системы. Пусть также для каждого изменения построен вектор метрик. Тогда данные векторы метрик можно кластеризовать таким образом, что каждому полученному кластеру будет соответствовать один класс изменений.*

Обоснование справедливости данной гипотезы будет приведено в последней главе диссертации на основе экспериментальной ее проверки.

**2.2. Метод автоматизированной классификации изменений исходного кода.** Автоматизацию классификации изменений исходного кода в данной работе предлагается строить на основе кластеризации метрик изменений.

Схема метода приведена на рис. 1. В качестве входных данных метода используются множество изменений для классификации $\Delta = \{\delta_r\}$, множество экспертных классов $C = \{c_i\}$ $(1 \le i \le n)$, набор метрик изменений $\mu$, число кластеров $k$, которое первоначально задается равным числу экспертных классов $n$.

Выходные данные метода – множество классифицированных изменений. Опишем этапы метода, соответствующие блокам схемы, приведенной на рис. 1.

**2.2.1. Экспертная настройка.** В процессе настройки экспертом выбирается *набор метрик изменений*, который позволяет обеспечить заданные уровни $P_{Cmin}$, $E_{Cmax}$ критериев качества классификации. Описание этих критериев приводится в пунктах 2.2.4, 2.2.7.

В эксперименте, описанном в главе 3, используются одиннадцать специально выбранных метрик, приводящих к удовлетворительным значениям характеристик качества для практического использования метода.

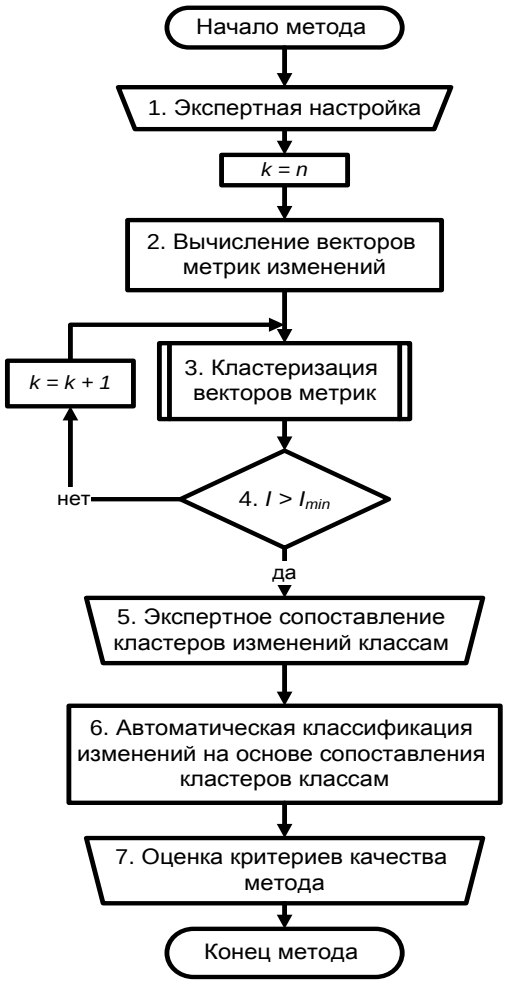


Рис. 1. Схема метода классификации изменений исходного кода на основе кластеризации метрик изменений

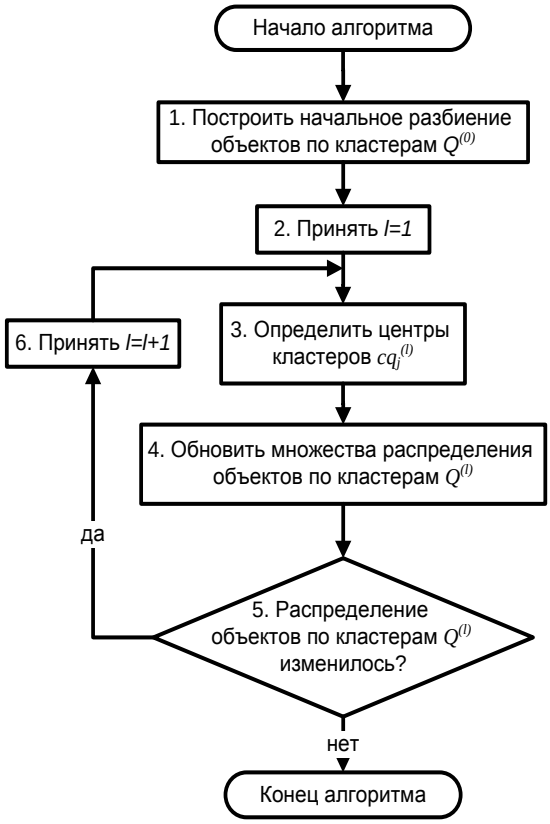


Рис. 2. Схема алгоритма кластеризации *k*-средних

**2.2.2. Вычисление векторов метрик изменений**. Для каждого изменения формируется вектор из значений метрик, выбранных на этапе настройки метода. Формулы для расчета значений метрик изменений приведены ниже. На выходе этапа формируется множество векторов метрик изменений $\{\mu\delta_r\}$.

Назовем *метрикой изменения* $\delta_r$ *исходного кода* числовую величину $\mu\delta_r$, которая характеризует данное изменение. Эту метрику для кода $S$ предлагается вычислять по формуле:

$$\mu\delta_r = M^{\delta}(S_{r-1}, S_r),$$

где $M^{\delta}$ – метрика изменения. Метрика $M^{\delta}$ вычисляется на основе кода, предшествующего изменению $S_{r-1}$, и измененного кода $S_r$. На практике более удобно производить расчет метрик в терминах добавленных, измененных и удаленных строк кода. Обобщим понятие строки кода $l$ как последовательности лексем $s_1 s_2 \ldots s_e$, где $s_e$ – лексема конца строки.

Каждое изменение $\delta_r$ можно представить в виде совокупности добавленных $L_+$, удаленных $L_-$ и измененных $L^*$ строк кода:

$$\delta_r = \{L_+, L_-, L^*\},$$

где $L^*$ – множество пар $<l_-^*, l_+^*>$, где $l_-^*$ – строка до внесения изменения и $l_+^*$ – измененная строка. В настоящей работе такое представление изменений производится методом поиска *редакционного предписания*. Редакционное предписание – это последовательность действий по вставке, удалению и замене символов, необходимая для получения простейшим образом из одной строки другой.

Приведем пример расчета метрик изменений, основанных на метрике цикломатической сложности кода. Цикломатическая сложность кода предназначена для оценивания сложности потока управления программы.

Зададим функцию *CC(l),* позволяющую вычислить упрощенную цикломатическую сложность строки кода $l$ как число конструкций языка, управляющих потоком исполнения программы, которые встречаются в строке $l$:

$$CC(l) = \sum_{s_i \in \boldsymbol{l}} [s_i \in S_{CF}],\ 1 \le i \le n_l.$$

Здесь $\{s_i\}$ – совокупность лексем исходной строки $l$, $n_l$ – число лексем в строке $l$, $S_{CF}$ – множество лексем, представляющих конструкции управления потоком исполнения программы. Выражение $[B]$ в настоящей работе принимает значение *единица,* когда значение предиката $B$ *истинно,* и *ноль,* когда значение $B$ *ложно.*

Приведем в качестве примера формулы для расчета метрик цикломатической сложности добавленного $CC_+$, удаленного $CC_-$, и измененного кода $CC^*$:

$$CC_+ = \sum_{l_+ \in L_+} CC(l_+),$$

$$CC_- = \sum_{l_- \in L_-} CC(l_-),$$

$$CC^* = \sum_{<l_-^*, l_+^*> \in L^*} (CC(l_+^*) - CC(l_-^*)).$$

Метрики $CC_+$, $CC_-$ рассчитываются по цикломатическим числам добавленной и удаленной строк соответственно. Цикломатическая сложность $CC^*$ рассчитывается как разность цикломатических чисел строки до внесения изменения и измененной строки.

Метрики, используемые в диссертации при анализе исходного кода, приведены в табл. 1.

Для каждого изменения $\delta_r$ рассчитывается $p$-мерный вектор метрик, где $p$ – число используемых метрик:

$$\boldsymbol{\mu}\delta_r = <\mu_1\delta_r, \mu_2\delta_r, \ldots, \mu_p\delta_r>.$$

Векторы метрик изменений используются для распределения изменений по кластерам.

**2.2.3. Кластеризация векторов метрик изменений.** На данном этапе формируется множество кластеров изменений $Q$. Кластеры $q_j \in Q$ – это непересекающиеся подмножества изменений, объединенные по заданным критериям сходства в процессе кластеризации.

Кластеризацию будем выполнять с помощью алгоритма *k-средних* с использованием *косинусной меры* близости изменений.

Выбор алгоритма кластеризации *k-средних* объясняется простотой его использования при приемлемом качестве результата. При его использовании необходимо предположение относительно ожидаемого числа кластеров. На

первом этапе метода число кластеров полагается равным числу экспертных классов, а в дальнейшем может уточняться.

Таблица 1. Метрики изменений кода, используемые в диссертации

| Обозначение метрики | Описание метрики | Обозначение метрики | Описание метрики |
|---|---|---|---|
| *1.* $LOC_+$ | Число добавленных строк кода | *7.* $F^*$ | Число измененных файлов исходного кода |
| *2.* $LOC_-$ | Число удаленных строк кода | *8.* $I_+$ | Число добавленных интерфейсов |
| *3.* $LOC^*$ | Число измененных строк кода | *9.* $I_-$ | Число удаленных интерфейсов |
| *4.* $CC_+$ | Цикломатическая сложность добавленного кода | *10.* $CS_+$ | Число добавленных классов и структур |
| *5.* $CC_-$ | Цикломатическая сложность удаленного кода | *11.* $CS_-$ | Число удаленных классов и структур |
| *6.* $CC^*$ | Цикломатическая сложность измененного кода | | |

Метод *k*-средних имеет недостаток – зависимость результата кластеризации от масштабов изменения отдельных переменных. При использовании евклидовой меры близости изменения группируются в кластеры на основании их масштаба, а не экспертного класса.

Автором диссертации было найдено решение проблемы – использование меры близости изменений кода, не учитывающей длину векторов метрик изменений. Благодаря использованию меры $\rho$, основанной на косинусе угла между векторами метрик, появилась возможность выделения классов изменений без учета их масштаба:

$$\rho(v_1, v_2) = \cos(v_1, v_2),\ \cos(v_1, v_2) = \frac{v_1^t v_2}{\|v_1\|\|v_2\|},$$

где $v_1$, $v_2$ – векторы метрик изменений, ${v_1}^t$ – транспонированный вектор $v_1$. Данная мера широко применяется при решении других задач, и оправдала себя, например, для кластеризации текстовых документов. Объекты считаются тем более близкими, чем ближе значение соответствующей меры $\rho$ к единице.

Выбор метода кластеризации *k*-средних с косинусной мерой обосновывается достижением достаточных для практического использования значений критериев качества классификации. Подтверждающие это эксперименты приведены в разд. 2 третьей главы.

В алгоритме кластеризации векторов метрик изменений используются следующие входные данные: число кластеров *k* и векторы метрик изменений $\mu\delta_r$. Выходные данные алгоритма: множество кластеров изменений *Q*. Схема алгоритма приведена на рис. 2.

Ниже приведено описание работы этого алгоритма.

2.2.3.1. Производится начальное разбиение $Q^{(0)} = \{q_1^{(0)}, q_2^{(0)}, \ldots, q_k^{(0)}\}$ множества объектов $\{\delta_i\}$ таким образом, чтобы $k$ наиболее далеко расположенных друг от друга изменений были первыми включены в различные кластеры:

$$q_1^{(0)}=\{\delta_1\},\ q_j^{(0)} = \{\delta_i \mid \rho(\mu\delta_i, cq_t^{(0)}) = min_{i,t}\,\rho(\mu\delta_i, cq_t^{(0)})\},$$
$$2 \le j \le k,\ 1 \le i \le N,\ 1 \le\ t \le j,$$

где $N$ – общее число изменений.

2.2.3.2. Номер итерации $l$ принимается равным единице.

2.2.3.3. Центры кластеров $cq_j^{(l)} = \{cq_j^{(l)}\}$ определяются по формуле:

$$cq_j^{(l)} = \frac{\sum_i \left(\left[\delta_i \in q_j^{(l-1)}\right]\mu\,\delta_i\right)}{\sum_i \left[\delta_i \in q_j^{(l-1)}\right]},\ 1 \le i \le N,\ 1 \le j \le k,$$

2.2.3.4. Множества распределения объектов по кластерам обновляются по формуле $Q^{(l)} = \{q_j^{(l)}\}$, где:

$$Q^{(l)} = \{\delta_i \mid \rho(\mu\delta_i, cq_j^{(l)}) = max_i\,\rho(\mu\delta_i, cq_j^{(l)})\},\ 1 \le i \le N,\ 1 \le j \le k.$$

2.2.3.5. Проверяется условие: $\sum_i ||q_j^l - q_j^{l-1}|| = 0$. Здесь под знаком "–" понимается операция взятия симметрической разности множеств: $A–B = (A \cup B) \setminus (A \cap B)$. Если условие выполнено, то процесс завершается и осуществляется переход к шагу 3 и номер итерации $l$ увеличивается на единицу.

Приведенный алгоритм позволяет автоматически разбить множество изменений на кластеры. В каждый из них объединяются наиболее схожие друг с другом изменения.

Этот алгоритм реализован в открытом программном средстве *CLUTO (Karypis Lab, University of Minnesota, США)*. Средство *CLUTO* используется в настоящей работе для кластеризации метрик изменений. Ниже приводится способ расчета функционала качества кластеризации.

**2.2.4. Оценка критерия качества кластеризации.** Для оценки качества разбиения изменений на кластеры был выбран функционал $I$, как наиболее простая мера «плотности» группировки изменений по кластерам. Этот функционал рассчитывается по следующей формуле:

$$I = \sum_{j=1}^{k} \sqrt{\sum_{\delta_1, \delta_2 \in q_j} \rho(\delta_1, \delta_2)},$$

где $k$ – число кластеров, $q_j$ – кластер с номером $j$, $\rho$ – мера близости изменений, $\delta_1$, $\delta_2$ – изменения из кластера $q_j$. Смысл этого функционала состоит в том, что чем выше его значение, тем плотнее изменения группируются в пределах кластеров, что свидетельствует о повышении качества решения задачи кластеризации.

На данном этапе метода проверяется условие превышения значения функционала $I$ над заданным экспертом минимальным уровнем $I_{min}$. Если условие не выполняется, то проводится подбор нового числа кластеров $k$.

**2.2.5. Экспертное сопоставление кластеров изменений классам.** На данном этапе каждому кластеру сопоставляется некоторый экспертный класс. При

таком сопоставлении устанавливается соответствие между кластерами $q_1, q_2, \ldots, q_k$ и классами изменений $c_1, c_2, \ldots, c_n$. В настоящей работе предлагается проводить сопоставление кластеров классам на основании выборочной экспертной классификации изменений из каждого кластера.

В каждом кластере выделяется несколько изменений, наиболее близких к его центру. Производится их экспертная классификация. На основе преобладания изменений некоторого класса $c$ в кластере $q$ все объекты кластера $q$ классифицируются как $c$: $q \to c$.

Процесс автоматизированной классификации изменений можно считать завершенным, если значения критериев качества, рассчитанных с помощью приведенных ниже формул, удовлетворяют заданным уровням. Иначе необходимо повторно выполнить этап экспертной настройки, описанный в п. 2.2.1.

**2.2.6.** Результатом **автоматической классификации изменений** на основе сопоставления кластеров, которым они принадлежат, классам, является построенная классификация изменений по заданным экспертным классам.

**2.2.7. Оценка критериев качества метода.** Оценка качества метода автоматизированной классификации проводится на проверочном множестве изменений $\Delta_e$.

Для каждого кластера $q_j$ рассчитываются следующие критерии: чистота $P(q_j)$ и энтропия $E(q_j)$. Чистота $P(q_j)$ кластера $q_j$ – отношение числа его изменений, принадлежащих экспертному классу $c_i$, соответствующему данному кластеру ($q_j \to c_i$), к числу изменений множества $\Delta_e$ в кластере $q_j$. Усредненные по всем кластерам значения чистоты $P_Q$ и энтропии $E_Q$ используются как оценки качества распределения изменений экспертных классов по соответствующим кластерам на всем множестве изменений $\Delta$. Критерии чистоты $P_Q$ и энтропии $E_Q$ используются для проверки гипотезы 1.

Оценки соответствия автоматизированной и экспертной классификации предпочтительно строить в терминах экспертных классов с помощью альтернативных критериев чистоты $P_C$ и энтропии $E_C$ *распределения множества изменений по экспертным классам*. Их расчет проводится аналогично расчету значений $P_Q$ и $E_Q$, с тем отличием, что роль кластеров в формулах играют экспертные классы. Если значения критериев $P_C$, $E_C$ не удовлетворяют заданным экспертом уровням $P_{Cmin}$, $E_{Cmax}$, то необходимо повторно выполнить этап экспертной настройки, описанный в п. 2.2.1.

Формулы для расчета значений чистоты $P_Q(\Delta_e)$ и энтропии $E_Q(\Delta_e)$ разбиения множества изменений $\Delta_e$ по кластерам $Q$ следующие:

$$P_Q(\Delta_e) = \sum_{j=1}^{k} \frac{n_j^e}{N_e} P(q_j),\ P(q_j) = \frac{n_j^a}{n_j^e},$$

$$E_Q(\Delta_e) = \sum_{j=1}^{k} \frac{n_j^e}{N_e} E(q_j),\ E(q_j) = -\frac{1}{\log n} \sum_{i=1}^{n} \frac{n_j^i}{n_j^e} \log \frac{n_j^i}{n_j^e},$$

где $N_e$ – общее число изменений множества $\Delta_e$, $n$ – число классов изменений, $k$ – число кластеров, $n^e{}_j$ – число попавших в кластер $q_j$ изменений множества $\Delta_E$, $n^i{}_j$ – число попавших в кластер $q_j$ изменений множества $\Delta_e$, принадлежащих классу $c_i$,

$n^a_j$ – число попавших в кластер $q_j$ изменений множества $\Delta_e$, принадлежащих экспертному классу $c^a_j$ которому сопоставлен кластер $q_j$. В случае, если $n^i_j$ равно нулю, то значение члена $\frac{n^i_j}{n_j}\log\frac{n^i_j}{n_j}$ также полагается равным нулю.

Чистота $P(q_j)$ кластера $q_j$ – отношение числа присутствующих в нем изменений экспертного класса, соответствующего этому кластеру, к общему числу классифицированных изменений кластера. Чистота характеризует степень соответствия кластера экспертному классу. Чем большее значение чистоты – тем лучше выполнена кластеризация. Энтропия $E(q_j)$ кластера $q_j$ характеризует неопределенность соответствия кластера экспертному классу. Чем меньше энтропия, тем лучше полученное решение. Чистота решения, в котором каждый кластер полностью представлен одним классом, равна единице, а энтропия – нулю.

Используем критерии $P_Q$, $E_Q$ для проверки гипотезы 1. Для этого переформулируем ее в следующем виде:

*Для проверочного множества изменений $\Delta_e$ автоматизированный метод классификации изменений производит такое разбиение изменений по кластерам с последующем сопоставлением их классам, что для заданных уровней $P_{Qmin}$, $E_{Qmax}$ справедливо следующее неравенство:*

$$P_Q(\Delta_E) > P_{Qmin},\ E_Q(\Delta_E) < E_{Qmax},$$

где $P_{Qmin}$, $E_{Qmax}$ – допустимые уровни чистоты и энтропии распределения экспертных изменений по кластерам.

Для проверки данной гипотезы и оценки доверительных интервалов значений чистоты и энтропии предлагается использовать известный метод размножения выборок, который применяется, так как в эксперименте недостаточно данных для статистического оценивания. Классификация изменений – процесс, требующий существенных затрат времени экспертов. Поэтому на практике очень сложно построить множества классифицированных изменений достаточного объема.

Разделим заданное множество изменений на *M* равных частей. Затем исключим первую часть и используем *M-1* оставшихся частей как отдельную выборку. Вернем исключенную на предыдущем шаге часть, исключим вторую и используем оставшиеся части с номерами *1,3,…,M* как следующую выборку и так далее. На основе сгенерированных таким образом выборок произведем оценивание статистических параметров.

В третьей главе будет показано, для каких значений $P_{Qmin}$, $E_{Qmax}$ на практике выполняется гипотеза об автоматизированной классификации изменений.

**3.1. В первом разделе третьей главы** содержится подробное описание вариантов применения автоматизированной классификации изменений в процессе разработки программ. Автоматизированная классификация изменений может быть полезна всем членам команды разработки. Изложенное в данном разделе было использовано в ходе внедрения, описанного в следующем разделе.

**3.2. Во втором разделе третьей главы** описано внедрение в компании *ЗАО* «*Транзас Технологии»* автоматизированной классификации изменений исходного

кода на основе кластеризации метрик. Внедрение выполнено (о чем свидетельствуют акты внедрения) при создании: 1. Системы мониторинга мобильных объектов *Navi-Manager; 2.* Компонент системы глобального мониторинга флота *LRIT; 3.* Системы *e-Tutor 5000* контроля действий студентов в процессе обучения на судовом, крановом и других тренажерах.

Предлагаемый метод использовался при анализе программных систем с открытым кодом: системы контроля версий *Subversion* (*subversion.tigris.org, CollabNET, США*) и объектной обертки над реляционными базами данных *NHibernate* (*JBoss, США*).

В качестве примера использования предложенного метода опишем его использование в процессе доработки системы *NHibernate*. Входные данные эксперимента приведены в табл. 2.

Таблица 2. Входные данные эксперимента по использованию метода классификации изменений в системе *NHibernate*

| | |
|---|---|
| **Общая информация** | |
| Экспертные классы $C$ | *Удаление кода, реализация новой функциональности, исправление логики, рефакторинг, форматирование кода с изменением комментариев* |
| Метод кластеризации метрик изменений | $k$-средних |
| Мера близости векторов метрик $\rho$ | Косинус угла между векторами метрик |
| Набор метрик для кластеризации $\mu$ | Приведен в табл. 1 |
| **Входные данные метода** | |
| Общее число классифицируемых изменений | 2069 |
| Число классифицированных экспертом изменений | 73 |
| **Данные настройки метода** | |
| Выбранное число кластеров $k$ | 12 |

Опишем, как выполняется автоматизированная классификация изменений.

3.2.1. В процессе **экспертной настройки** выбраны метрики, приведенные в табл. 1. Проверочное множество $\Delta_{e68}$ из 68 изменений выбрано следующим образом. Из каждого выделяемого автоматизированным методом класса случайным образом выбрано по 16 изменений. Из 80 полученных отобрано 68 изменений, для которых совпали результаты экспертной классификации двух независимых экспертов.

3.2.2. Произведено **вычисление векторов метрик** для рассматриваемого множества из 2069 изменений.

3.2.3. Проведено разбиение векторов метрик изменений на двенадцать кластеров. Выбор числа кластеров $k$ проводился на основе анализа значения функционала качества кластеризации $I$, а также размера «плохих» кластеров с сильно отличающимися друг от друга мерами близости изменений.

3.2.4. Осуществлена оценка критерия качества кластеризации для различного числа кластеров $k$ от 5 до 15. Значение функционала качества кластеризации $I$ увеличивается с ростом значений $k$. При значении $k = 12$ достигается наименьший размер самого «плохого» кластера – 547 изменений.

3.2.5. Сопоставление кластеров изменений экспертным классам проведено с помощью отбора изменений из каждого кластера, их экспертной классификации и установлении соответствия кластера классу на основе преобладания в нем изменений данного класса.

3.2.6. Автоматическая классификация изменений на основе сопоставления их кластеров классам проводится на основе предыдущего шага.

3.2.7. Значения критериев качества метода чистоты $P_Q(\Delta_{e68})$ и энтропии $E_Q(\Delta_{e68})$ разбиения проверочного множества $\Delta_{e68}$ по кластерам приведены в табл. 3.

Таблица 3. Распределение изменений проверочного множества по кластерам. Обозначения экспертных классов: «И» – исправление логики, «Ф» – форматирование кода с изменением комментариев, «У» – удаление кода, «Н» – реализация новой функциональности, «Реф.» – рефакторинг. $n^e_j$ – число изменений множества $\Delta_{e68}$ в кластере $q_j$

| № кластера $j$ | Результат экспертного сопоставления $q_j \rightarrow c_i$ | Экспертные классы $c_i$ | | | | | $n^e_j$ | $P_Q$ | $E_Q$ |
|---|---|---|---|---|---|---|---|---|---|
| | | И | Ф | Н | У | Реф. | | | |
| 0 | И | 9 | 0 | 0 | 1 | 1 | 11 | 0,82 | 0,37 |
| 1 | Ф | 1 | 4 | 0 | 0 | 0 | 5 | 0,8 | 0,31 |
| 2 | Ф | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 3 | Ф | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | Ф | 1 | 7 | 0 | 0 | 0 | 8 | 0,88 | 0,23 |
| 5 | Н | 0 | 0 | 8 | 0 | 0 | 8 | 1 | 0 |
| 6 | Н | 0 | 0 | 4 | 0 | 0 | 4 | 1 | 0 |
| 7 | Н | 1 | 1 | 1 | 0 | 0 | 3 | 0,33 | 0,68 |
| 8 | У | 1 | 0 | 0 | 13 | 1 | 15 | 0,87 | 0,3 |
| 9 | Реф. | 2 | 1 | 0 | 0 | 3 | 6 | 0,5 | 0,63 |
| 10 | Реф. | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 11 | Реф. | 2 | 2 | 0 | 0 | 2 | 6 | 0,33 | 0,68 |
| Итого | | 17 | 16 | 13 | 14 | 8 | 68 | 0,78 | 0,32 |

В табл. 4 приведены результаты оценки критериев $P_C$ и $E_C$. Значения в таблице получены, во-первых, объединением строк для кластеров, представляющих один и тот же экспертный класс, расчета $P_C(\Delta_{e68})$ и $E_C(\Delta_{e68})$ для них, и, во-вторых, усреднением полученных значений по пяти экспериментам, проведенным с помощью метода размножения выборок. Уровень значимости при построении доверительных интервалов для значений $P_C$ и $E_C$ принят равным 0,05.

Таблица 4. Усредненные значения критериев качества автоматизированной классификации изменений программной системы *NHibernate*

| | |
|---|---|
| Число изменений размноженного проверочного множества | 272 |
| Качество классификации | $P_C = 0{,}75 \pm 0{,}05$<br>$E_C = 0{,}37 \pm 0{,}06$ |

Из данной таблицы следует, что для проанализированной программной системы в среднем 75% изменений будут корректно классифицированы разработанным методом. Кроме того, полученное значение энтропии свидетельствует о наличии существенной неопределенности соответствия кластеров классам. На этапе экспертного сопоставления кластера классам было обнаружено, что это в основном связано с недостаточно качественным выделением кластеров, соответствующих классу *рефакторинг*.

Применение автоматизированной классификации изменений позволило распределить 2069 изменений по пяти классам на основе кластеризации по двенадцати кластерам.

Для сравнения предлагаемого подхода с «чисто экспертной оценкой» приведем следующие цифры. Только 73 изменения были классифицированы для сопоставления кластеров классам в ходе экспертной классификации. Автоматизированный классификатор распределил 53 изменения проверочного множества из 68 по тем же классам, что и эксперты. Следовательно, в результате применения метода **удалось существенно сократить затраты времени экспертов** на классификацию изменений.

Для изменений проанализированной программной системы **подтверждается гипотеза 1** для следующих значений $P_{Qmin}$ и $E_{Qmax}$ автоматизированной классификации с уровнем значимости 0,05:

$$P_{Qmin} = 0{,}71,\ E_{Qmax} = 0{,}34,$$

что свидетельствуют о приемлемом качестве автоматизированной классификации для практического применения метода.

На основе проведенных экспериментов можно сделать вывод, что метод работает хорошо, если производятся однотипные изменения, а когда в изменениях сочетаются разнородные модификации кода, то качество автоматизированной классификации ухудшается.

Из изложенного следует, что метод позволяет сократить время классификации изменений по сравнению с «чисто ручной» экспертизой при удовлетворительном для практики качестве.

**3.3. В третьем разделе третьей главы** описано разработанное автором программное средство, реализующее предложенный в диссертации метод. Оно состоит из следующих частей: модуль расчета метрик изменений, указанных в табл. 1 (модуль разработан автором, особенности расчета метрик изложены в п. 2.2.2); модуль поиска редакционных предписаний (рассмотрены в п. 2.2.2) изменений кода (основан на открытой системе контроля версий файлов *Subversion*); модуль кластеризации с оптимизацией функционала качества разбиения (основан на инструментальном средстве *CLUTO*, описание которого приведено в следующем разделе).

**3.4. Особенности реализации алгоритма кластеризации** в программном средстве *CLUTO* описаны в четвертом разделе третьей главы. *CLUTO* – специализированное программное средство, разработанное для кластеризации данных. Его особенностью является возможность задания параметра «число повторений» (*«ntrials»*), который указывает вычислить заданное число кластерных решений различными способами и выбрать лучшее из них на основе максимизации значения функционала качества кластеризации. Это позволяет улучшить качество кластеризации за счет снижения вероятности попадания значения функционала в локальный минимум. Кроме того, инструмент позволяет задавать различные алгоритмы для кластеризации, а также выбирать меру близости объектов кластеризации.

## Заключение

В диссертации получены следующие результаты.

1. Обоснована возможность частичной автоматизации классификации изменений исходного кода методом кластеризации метрик.
2. Обоснован выбор метода *k-средних* с мерой близости объектов для кластеризации, основанной на косинусе угла между векторами метрик изменений.
3. Разработан метод автоматизированной классификации изменений исходного кода на основе кластеризации метрик изменений, позволяющий сократить число изменений для классификации, выполняемой вручную.
4. Разработано программное средство автоматизированной классификации изменений предложенным методом.

Перечисленные результаты получены в ходе выполнения совместных работ *СПбГУ ИТМО* и *ЗАО «Транзас Технологии»* и используются как при разработке программного обеспечения сложных систем, так и в учебном процессе.

В работе приводятся результаты сравнения классификаций изменений в программной системе с открытым исходным кодом, выполненные с использованием предложенного автоматизированного метода и вручную. Для этого привлекались эксперты, имеющие опыт разработки сложных программных систем не менее пяти лет. Всего в работе было проанализировано пять программных систем. Исследования показали, что метод позволяет сократить время классификации изменений по сравнению с «чисто ручной» экспертизой (при анализе системы *NHibernate* необходимо классифицировать вручную лишь 73 изменения из 2069) при следующих значениях характеристик качества классификации: чистота $P_C = 0{,}75 \pm 0{,}05$ и энтропия $E_C = 0{,}37 \pm 0{,}06$.

В настоящее время в методе используется 11 метрик, перечень которых приведен в разд. 2.2. Автор предполагает, что возможно совершенствование качества классификации за счет увеличения числа используемых метрик, а также настройки на другие перечни классов изменений, кроме тех, которые рассмотрены в диссертации.

## Список публикаций


1. *Князев Е. Г., Шопырин Д. Г.* Использование автоматизированной классификации изменений программного кода в управлении процессом разработки программного обеспечения // Информационно-управляющие системы. 2008. № 5, с. 15–21. (Журнал из списка ВАК)
2. *Князев Е. Г., Шопырин Д. Г.* Автоматизированная классификация изменений программного кода методами многомерного статистического анализа // Информационные технологии. 2008. № 5, с. 48–53. (Журнал из списка ВАК)
3. *Князев Е. Г., Шопырин Д. Г.* Анализ изменений программного кода методом кластеризации метрик // Научно-технический вестник СПбГУ ИТМО. Исследования в области информационных технологий. 2007. Вып. 39, с. 197–208.
4. *Knyazev E.* Automated Source Code Changes Classification for Effective Code Review and Analysis / Proceedings of SYRCoSE 2008 (SYRCoSE) Spring Young Researchers Colloquium on Software Engineering. Volume 2, pp. 55–59.
5. *Князев Е. Г.* Методы обнаружения закономерностей эволюции программного кода / Труды XIV Всероссийской научно-методической конференции «Телематика-2007». СПбГУ ИТМО. 2007. Т.2, с. 435–436.
6. *Князев Е. Г., Лобанов П. Г.* Оценка опасности изменений программного кода путем анализа покрытия кода модульными тестами / Сборник докладов X международной конференции по мягким вычислениям и измерениям. СПбГЭТУ «ЛЭТИ». 2007. Т.2, с. 273–275.
7. *Князев Е. Г. Шопырин Д. Г.* Использование автоматической классификации изменений программного кода в управлении процессом разработки программного обеспечения / Тезисы докладов Software Engineering Conference (Russia) (SECR–2007). М.: Tekama. 2007. Цифровой носитель.
8. *Князев Е. Г., Шопырин Д. Г.* Анализ изменений программного кода методом кластеризации метрик / Сборник тезисов IV межвузовской конференции молодых ученых. СПбГУ ИТМО. 2007, с. 68.
9. *Князев Е. Г.* Применение автоматической классификации изменений программного кода для отслеживания реализации несанкционированной функциональности / V Санкт-Петербургская межрегиональная конференция «Информационная безопасность регионов России». СПб.: СПОИСУ. 2007, с. 84.
10. *Князев Е. Г.* Применение алгоритма нечеткой кластеризации метрик для классификации изменений программного кода / Сборник тезисов V Всероссийской межвузовской конференции молодых ученых. СПбГУ ИТМО. 2008, с. 282, 283.
11. *Князев Е. Г.* Применение алгоритма потоковой кластеризации для задачи классификации модификаций исходного кода / Тезисы докладов XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке». СПб ГПУ. 2008, с. 289, 290.

Тиражирование и брошюровка выполнены в копировальном
центре «Средний-28»
199004, Санкт-Петербург, Средний просп. В.О., 28/29. Тел. (812)323-40-44
Тираж 100 экз.
Усл. печ. л. 1,0.