На правах рукописи

Кирносенко Семён Игоревич

Метод прогнозирования дефектов в компонентах программного обеспечения на основе метрик исходного кода

05.13.01 – Системный анализ, управление и обработка информации (промышленность)

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Волгоград – 2012

Работа выполнена на кафедре «ЭВМ и системы» Волгоградского государственног технического университета.

Научный руководитель	доктор технических наук, профессор Лукьянов Виктор Сергеевич.
Официальные оппоненты:	доктор технических наук, профессор Шевчук Валерий Петрович;
	кандидат технических наук Стрельников Олег Иванович.
Ведущая организация	Пензенский государственный университет.

Защита состоится 1 марта 2012 г. в 15:00 на заседании диссертационного совета Д 212.028.04 при Волгоградском государственном техническом университете по адресу: 400005, г. Волгоград, пр-т им. В.И. Ленина, 28, аудитория 209.

С диссертацией можно ознакомиться в библиотеке Волгоградского государственного технического университета.

Автореферат разослан «26» января 2012 г.

Ученый секретарь
диссертационного совета Водопьянов В. И.

Общая характеристика работы

Актуальность работы. Современные технологические подходы к разработке программного обеспечения носят сложный, часто итерационный характер. Такие виды деятельности, как проектирование, кодирование, тестирование и эксплуатация в этих подходах часто совмещены, либо сменяют друг друга циклически при переходе от итерации к итерации. Процесс тестирования может осуществляться непрерывно, поскольку в исходный код постоянно вносятся новые функции и исправления. Чем больше объём исходного кода, тем больше трудозатрат необходимо для поддержания качества такой программной системы на должном уровне. Для решения этой проблемы применяются различные практики и методологии программирования, автоматизированное тестирование, методы обнаружения и прогнозирования дефектов. Применение методологий разработки нацелено на снижение числа вносимых в исходный код дефектов. Целью автоматизированного тестирования является также снижение числа вносимых дефектов и недопущение повторного появления ранее обнаруженных и исправленных дефектов. В выявлении уже внесённых в исходный код ефектов, кроме ручного тестирования, могут помочь только средства обнаружения прогнозирования дефектов. Средства обнаружения дефектов работают на основе татического анализа исходного кода. При этом осуществляется поиск дефектов, оответствующих типичным ошибочным паттернам, таким как использование неициализированной переменной. Поэтому такими средствами можно выявить лишь ебольшую долю дефектов.

Для прогнозирования дефектов в компонентах программного обеспечения существуют программные средства: AgenaRisk, AID, SCULLY, Software Evolution Project и другие. К сожалению, бóльшая часть этих программных средств не доступна для вободного использования и предоставляется только вместе с услугами консалтинга. ругие программные средства либо дорого стоят, либо требуют реализации дополнительных утилит для сбора данных. Кроме того, применяемым в них методам прогнозирования дефектов свойственны серьёзные недостатки. Методы на основе Байесовских сетей доверия требуют для работы получения экспертных оценок, что означает еобходимость привлечения персонала, обладающего специальной квалификацией. етоды на основе регрессионных моделей и алгоритмов машинного обучения часто спользуют метрики исходного кода, зависимые от языка программирования, что ребует реализации средств синтаксического разбора исходного кода. Также прогнозирование с использованием этих методов представляет собой «чёрный ящик». хотя прогнозы таких методов могут быть достаточно точными, на основе этих прогнозов нельзя сделать выводы о том, что является причиной конкретных дефектов какие меры можно предпринять для улучшения ситуации в будущем.

Это обуславливает актуальность работы по повышению точности прогнозирования дефектов в компонентах программного обеспечения.

Целью диссертационной работы являлось повышение точности прогно рования дефектов в компонентах программного обеспечения.

Для достижения поставленной цели решены следующие **задачи**:

- анализ существующих методов прогнозирования дефектов в компонентах п граммного обеспечения, их ограничений и требований;
- формулирование гипотез о закономерностях эволюции исходного кода пр граммного обеспечения на стадии разработки;
- разработка метода прогнозирования дефектов в компонентах программно обеспечения на основе сформулированных гипотез;
- разработка набора программных утилит, обеспечивающих прогнозирование д фектов в компонентах программного обеспечения на основе предложенно метода;
- проведение вычислительных экспериментов с целью проверки гипотез о зак номерностях эволюции исходного кода в программном обеспечении на стади разработки;
- проведение вычислительных экспериментов с целью проверки прогностич ской способности предложенного метода прогнозирования дефектов и его сра нения с существующими методами.

В работе использовались **методы** теории вероятностей и математической стат стики, системного анализа, теории принятия решений, сигнальной теории, линейно алгебры, регрессионного анализа, основные принципы проектирования объектн ориентированных приложений.

Научная новизна. На защиту выносятся результаты, обладающие научно новизной.

1. Предложен новый метод прогнозирования дефектов в компонентах программ ного обеспечения, который является независимым от языка программировани и не требует экспертных оценок.
2. Предложен новый метод оценки вероятности нахождения дефектных стро в исходном коде, написанном отдельным разработчиком в отдельном компо ненте, на основе измеренных относительных частот нахождения дефектны строк в исходном коде отдельных разработчиков и компонентов, который м жет быть использован как для прогнозирования дефектов, так и для получе ния непосредственно метрик.
3. Предложены новые численные метрики исходного кода: время стабилизации, востребованность, специализация разработчика и другие.

Практическая значимость. Разработанный в рамках диссертации набор утилит MSR Tools позволяет прогнозировать в каких компонентах программного обеспечения имеются необнаруженные дефекты, каково общее количество необнаруженных дефектов; каково распределение дефектов между отдельными программными подсистемами; кто из разработчиков внёс исходный код, ставший причиной отдельного дефекта и т. п. Эта информация может быть использована для принятия проектных решений различного уровня сложности: распределения ресурсов тестирования; принятия решения о завершении стадии тестирования; распределения задач между разработчиками; оценки квалификации персонала; выявления уровня компетенции разработчиков, в зависимости от вида выполняемых ими работ; выявления проблемных участков исходного кода, нуждающихся в рефакторинге и т. п.

Набор утилит MSR Tools является относительно простым, не требует от пользователя наличия специальных знаний и рассчитан на широкий круг пользователей: разработчиков, менеджеров, QA-специалистов.

Апробация работы. Основные результаты диссертации докладывались на XIII-XIV региональных конференциях молодых исследователей Волгоградской области (2009-2010), на конференциях «Технологии Microsoft в теории и практике программирования» (2008-2009), г. Нижний Новгород, на всероссийской научной конференции «Наука. Технологии. Инновации.» (2008), г. Новосибирск, на 6-й межрегиональной научно-технической конференции студентов и аспирантов (2009), г. Смоленск, на конференции «Приоритетные направления современной российской науки глазами молодых учёных» (2009), г. Рязань, на конференции «Актуальные вопросы современной техники и технологии.» (2010), г. Липецк, на всероссийской научно-практической конференции «Теоретические вопросы разработки, внедрения и эксплуатации программных средств» (2011), г. Орск.

Внедрение результатов работы. В рамках диссертации разработан набор утилит MSR Tools, который был использован в научно-исследовательской работе «Разработка программного комплекса расчетов задач динамики систем многих тел сверхбольшой размерности на основе параллельных алгоритмов для вычислительного кластера» (тема №34-53/170-09) на этапе «Повышение качества системы FRUND посредством локализации дефектов в исходном коде». Эта работа выполняется на кафедре высшей математики Волгоградского государственного технического университета. Применение набора утилит MSR Tools к исходному коду системы FRUND позволило выявить компоненты, нуждающиеся в дополнительном тестировании; оценить сложность исходного кода различных подсистем; сделать выводы о квалификации, трудозатратах, производительности труда и других характеристиках отдельных разработчиков. На основе полученных результатов были даны рекомендации относительно распределения ресурсов тестирования в ближайшей перспективе. Также выполнено внедрение набора утилит MSR Tools в учебный процесс на кафедре «ЭВМ и системы» Волгоградского государственного технического университета по курсу «Технологии программирования».

Достоверность результатов исследования обусловлена корректным испол зованием применяемого математического аппарата и подтверждается эксперименталь установленной высокой прогностической способностью предложенного м тода прогнозирования дефектов, а также согласием с известными результатам полученными другими авторами.

Публикации. Материалы диссертации опубликованы в 12 печатных работах из них 3 в изданиях, включённых в Перечень ВАК. Также имеется свидетельств № 2011614559 о государственной регистрации программы для ЭВМ «Набор утили для расчёта метрик надёжности исходного кода».

Структура и объем диссертации. Диссертация состоит из введения, четы рёх глав и заключения. Содержание работы изложено на 169 страницах машинопи ного текста. Работа иллюстрирована 16 рисунками и содержит 35 таблиц. Списо литературы включает 131 наименование, в том числе 19 отечественных.

Содержание работы

Во введении обоснована актуальность диссертационной работы, сформули рованы цели и задачи исследования, научная новизна, практическая значимост полученных результатов.

В первой главе приводится обзор литературы, касающейся задачи прогн зирования дефектов в компонентах программного обеспечения. Отмечается, что з последние двадцать лет широкое распространение и применение получили методы прогнозирования дефектов на основе анализа эволюции исходного кода. Информация об эволюции исходного кода может быть получена из различных программных репозиториев: систем контроля версий, систем учёта дефектов и других. Активная работа по данному направлению ведётся в Колледже королевы Марии Лондонского университета, Американском национальном институте статистики, университете Западной Вирджинии, а также в лабораториях ряда корпораций: Microsoft Research, AT&T Labs Research и других. Большой вклад в разработку методов прогнозирования дефектов и работу по развитию смежных направлений исследований внесли следующие специалисты: Norman E. Fenton, Todd L. Graves, Ahmed E. Hassan, Taghi M. Khoshgoftaar, Tim Menzies, Audris Mockus, Nachiappan Nagappan, Martin Neil, Thomas J. Ostrand, Thomas Zimmermann. В нашей стране данной темой занимаются в Санкт-Петербургском государственном политехническом университете, Санкт-Петербургском государственном университете информационных технологий, механики и оптики, Институте проблем управления им. В.А. Трапезникова РАН. Однако бóльшая часть отечественных работ посвящена методам обнаружения дефектов посредством статического анализа исходного кода.

Приводится ряд определений дефекта в программном обеспечении из различных источников. Перечисляются причины дефектов, которые являются наиболее существенными в соответствии с рассмотренной литературой. Рассматриваются раз-

личные методы прогнозирования дефектов. Приводятся достоинства и недостатки методов различных типов, типичные сценарии их применения.

Описываются два подхода к прогнозированию дефектов в компонентах программного обеспечения. Первый подход ставит задачей метода классификацию отдельных компонентов программного обеспечения, как содержащих необнаруженные дефекты или не содержащих таковых. Чаще всего классификация осуществляется на основе оценок вероятностей нахождения необнаруженных дефектов, полученных для каждого отдельного компонента. В качестве значения «порога отсечения» используется число 0,5. Соответственно, если оценка вероятности наличия необнаруженных дефектов для компонента оказывается не меньше 0,5, то компонент классифицируется как содержащий необнаруженные дефекты, в противном случае – не содержащий таковых. Второй подход ставит задачей метода выборку минимального множества компонентов, содержащих максимальное число необнаруженных дефектов. При этом для отдельных компонентов также может вычисляться вероятность нахождения в них необнаруженных дефектов. Выполняется выборка 20% компонентов, для которых оценки этой вероятности оказались наиболее высоки. Попавшие в эту выборку компоненты классифицируются как содержащие наибольшее число необнаруженных дефектов. Это обосновывается соблюдением принципа Парето для соотношения числа компонентов и числа дефектов в них, что нашло экспериментальное подтверждение согласно ряду публикаций. Отмечается, что второй подход нельзя считать оптимальным, так как его точность будет снижаться в тех случаях, когда распределение необнаруженных дефектов в компонентах программного обеспечения близко к равномерному. Такая ситуация наиболее вероятна на ранних этапах разработки, а также в случаях быстрого роста объёма исходного кода или выполнения крупных рефакторингов, что может происходить в проекте даже после многих лет разработки. Поэтому делается вывод о предпочтительности использования первого подхода к прогнозированию дефектов в компонентах программного обеспечения.

Описываются три потенциальных источника данных для получения информации об эволюции исходного кода программного обеспечения на стадии разработки: системы контроля версий, системы учёта дефектов и листы почтовой рассылки. Системы контроля версий являются наиболее предпочтительным источником информации, поскольку используются чаще остальных. Даётся объяснение ряда понятий, связанных с организацией данных в системах контроля версий.

Выделены наиболее актуальные, современные проблемы при решении задач прогнозирования дефектов: недостаток инструментальных средств, недостаток экспериментальных данных и сложность их получения, сложность методов прогнозирования и ограниченность их применения. Делаются выводы о том, что несмотря на множество существующих методов прогнозирования дефектов, ряд проблем остаётся нерешённым. Эти проблемы делают применение существующих методов затруднительным и не всегда возможным.

Во второй главе на основе поставленной задачи и обозначенных актуальных проблем в сфере прогнозирования дефектов в программном обеспечении формулируются требования к разрабатываемому методу прогнозирования дефектов.

1. Метод должен быть применим к любому программному обеспечению вне зависимости от языка программирования, на котором оно реализовано.

2. Метод не должен требовать экспертных оценок или какой-либо другой информации, которая не может быть получена из системы контроля версий.

3. Метод должен давать прогноз, в каких компонентах программного обеспечения содержатся необнаруженные дефекты, с точностью, достаточной для использования прогнозов при решении задач управления процессом разработки.

4. Метод должен быть применим вне зависимости от наличия и количества версий программного обеспечения.

На основании имеющегося эмпирического опыта исследований процессов разработки программного обеспечения формулируются различные гипотезы о закономерностях процесса эволюции исходного кода в программном обеспечении на стадии разработки. Формулируется гипотеза о том, что написание отдельной строки исходного кода можно считать независимым испытанием с некоторой неизвестной вероятностью успеха P_{LS} и вероятностью неудачи $P_{LF} = 1 - P_{LS}$; причём успехом считается написание строки исходного кода, не содержащей дефектов, а неудачей – написание строки исходного кода, содержащей дефекты.

Формулируются различные гипотезы о распределении дефектных строк по объёму исходного кода, включая гипотезы о том, что это распределение является равномерным для всего исходного кода программного обеспечения; зависит только от квалификации разработчиков, которая может быть принята постоянной для каждого отдельного разработчика; зависит только от сложности исходного кода, которая может быть принята постоянной для каждого отдельного компонента; зависит одновременно от квалификации разработчика и сложности исходного кода. Последняя гипотеза уточняется различными вариантами вычисления вероятности нахождения дефектных строк в исходном коде, написанном отдельным разработчиком в отдельном компоненте. Предлагается метод вычисления этой вероятности, основанный на использовании измеренных относительных частот нахождения дефектных строк в исходном коде отдельных разработчиков и отдельных компонентов.

Пусть программное обеспечение состоит из N_C компонентов и его исходный код написан при участии N_A разработчиков. Тогда для исходного кода каждого компонента можно записать уравнение вида:

$$W_{LF_C} = P_{LF_C} + \sum_{i=1}^{N_A} P_{LF_{A_i}} \cdot \frac{LOC_{A_{A_i}}}{LOC_{A_C}}, \qquad (1)$$

где P_{LF_C} — неизвестная вероятность написания дефектной строки исходного кода в данном компоненте; $P_{LF_{A_i}}$ — неизвестная вероятность написания дефектной строки исходного кода i-ым разработчиком; W_{LF_C} — измеренная относительная частота появления дефектных строк в исходном коде данного компонента; LOC_{A_C} — измеренное общее число строк исходного кода, добавленных в данный компонент; $LOC_{A_{A_i}}$ — измеренное общее число строк исходного кода, добавленных в данный компонент i-ым разработчиком.

Для исходного кода каждого разработчика аналогично можно записать уравнение вида:

$$W_{LF_A} = P_{LF_A} + \sum_{j=1}^{N_C} P_{LF_{C_j}} \cdot \frac{LOC_{A_{C_j}}}{LOC_{A_A}}, \tag{2}$$

где P_{LF_A} — неизвестная вероятность написания дефектной строки исходного кода данным разработчиком; $P_{LF_{C_j}}$ — неизвестная вероятность написания дефектной строки исходного кода в j-ом компоненте; W_{LF_A} — измеренная относительная частота появления дефектных строк в исходном коде данного разработчика; LOC_{A_A} — измеренное общее число строк исходного кода, добавленных данным разработчиком; $LOC_{A_{C_j}}$ — измеренное общее число строк исходного кода, добавленных данным разработчиком в j-й компонент.

Для всего исходного кода в программном обеспечении можно записать уравнение вида:

$$W_{LF_T} = \sum_{i=1}^{N_A} P_{LF_{A_i}} \cdot \frac{LOC_{A_{A_i}}}{LOC_{A_T}} + \sum_{j=1}^{N_C} P_{LF_{C_j}} \cdot \frac{LOC_{A_{C_j}}}{LOC_{A_T}}, \tag{3}$$

где W_{LF_T} — измеренная относительная частота появления дефектных строк во всём исходном коде программного обеспечения; LOC_{A_T} — измеренное общее число строк исходного кода, добавленных в программное обеспечение. Получаем систему из $N_C + N_A + 1$ линейных уравнений с $N_C + N_A$ неизвестными. Посредством метода наименьших квадратов составляется новая система из $N_C + N_A$ нормальных уравнений, которая в свою очередь решается по обыкновенным правилам алгебры. В результате решения получаем оценки вероятностей написания дефектной строки в исходном коде отдельных компонентов и отдельных разработчиков, которые можно использовать для оценки вероятности написания дефектной строки исходного кода определённым разработчиком в определённом компоненте.

Формулируются гипотезы о различных видах зависимости между вероятностью обнаружения отдельного дефекта и временем внесения дефектных строк исходного кода, которые являются его причиной; о различных видах связи между вероятностью наличия необнаруженных дефектов в отдельном компоненте программного обеспечения и вносимыми в компонент модификациями; о точности прогнозирования, выполненного на основе оценок вероятности наличия необнаруженных дефектов в компонентах и на основе оценок числа необнаруженных дефектных строк в

них, а также ряд других гипотез.

Предлагается метод прогнозирования дефектов, основанный на сформулир ванных гипотезах о закономерностях эволюции исходного кода программного обес печения на стадии разработки. Метод заключается в оценке вероятности наличи необнаруженных дефектов в отдельных компонентах программного обеспечения При этом исходный код каждого компонента разбивается на подмножества. Каждо такое подмножества включает исходный код, написанный отдельным разработчико и добавленный в отдельном изменении системы контроля версий. Для исходного ко да отдельных подмножеств выполняется измерение ряда метрик: количество стро исходного кода, добавленных в подмножестве при его создании; количество строк ис ходного кода, удалённых из подмножества; относительная частота дефектных стро исходного кода для исходного кода компонента, в котором находится подмножество и разработчика, которым подмножество написано; а также другие метрики. На осн ве измеренных метрик и принятых гипотез об эволюции исходного кода в программ ном обеспечении на стадии разработки, выполняется расчёт вероятности наличия необнаруженных дефектов сначала в отдельных подмножествах исходного кода, затем и во всём компоненте. На основе вычисленной оценки вероятности нахожд ния необнаруженных дефектов, каждый отдельный компонент классифицируется либо как содержащий необнаруженные дефекты, либо как не содержащий таковых. Результатом прогнозирования является выборка компонентов программного обесп чения, которые в соответствии с прогнозом содержат необнаруженные дефекты.

Все вычисления, на основе которых работает предложенный метод, не требуют измерения зависимых от языка программирования метрик, наличия экспертных оц нок или версий программного обеспечения. Делаются выводы о том, что этот метод удовлетворяет всем сформулированным требованиям к нему, и лишь выполнение требования о точности требует экспериментального подтверждения.

В третьей главе описываются методы и алгоритмы, используемые в программ ной реализации. Обосновывается необходимость применения промежуточного представления информации, хранимой в системе контроля версий. Формулируются требования к такому представлению, исходя из потребностей предложенного метода прогнозирования дефектов. Рассматриваются ранее предложенные представления аналогичного назначения. Объясняются причины, по которым существующие представления не могут быть использованы для выполнения поставленных задач, что и стало причиной использования нового представления.

Описывается процесс извлечения необходимой информации из системы контроля версий. Это возможно посредством выполнения различных команд систем контроля версий и последующего синтаксического разбора их вывода. Бóльшая часть необходимой информации извлекается с помощью трёх основных команд: вывод списка изменений, сравнение файлов и аннотация исходного кода в отдельном файле. Для системы контроля версий Subversion эти команды имеют наименования «log», «diff» и «blame» соответственно. Поясняется принцип идентификации дефек-

тов и дефектного исходного кода. Для подтверждения корректности полученного промежуточного представления используются специально разработанные автором алгоритмы контроля. Эти алгоритмы позволяют выявлять любые несоответствия промежуточного представления и информации в системе контроля версий. Благодаря этому, можно утверждать, что вычисленные на основе промежуточного представления метрики полностью соответствуют действительности.

Отмечаются важные моменты, связанные с особенностями извлечения необходимой информации из систем контроля версий Subversion и Git. Описывается методика расчёта основных метрик исходного кода, которые необходимы для применения предложенного метода прогнозирования дефектов. В состав этих метрик входят как общеизвестные метрики: число строк исходного кода, число строк дефектного исходного кода, число исправленных дефектов, плотность дефектов, относительная частота дефектных строк исходного кода, время жизни дефекта; так и предлагаемые новые метрики: время стабилизации исходного кода, востребованность исходного кода, специализация разработчика.

Отмечается, что разработанный автором набор утилит MSR Tools включает следующие утилиты:

1. Mapper — утилита для выполнения перевода информации из системы контроля версий в промежуточное представление для последующего расчёта метрик на его основе. Также позволяет выполнять контроль корректности такого перевода. Имеет расширяемую архитектуру, позволяющую дополнять промежуточное представление различной информацией.

2. StatGenerator — утилита для получения различных метрик исходного кода в виде html-отчётов. Имеет расширяемую архитектуру, позволяющую управлять процессом формирования отдельных страниц отчёта и добавлять новые страницы.

3. Visualizer — утилита для построения графиков, гистограмм и других визуальных представлений информации об эволюции исходного кода. Имеет расширяемую архитектуру, позволяющую добавлять дополнительные алгоритмы визуализации.

4. Predictor — утилита для прогнозирования наличия дефектов в компонентах программного обеспечения. Позволяет получать прогнозы о наличии необнаруженных дефектов в отдельных компонентах и выполнять оценку точности таких прогнозов с использованием различных численных характеристик. Имеет расширяемую архитектуру, позволяющую добавлять дополнительные методы прогнозирования дефектов.

Делаются выводы о том, что разработанный набор утилит даёт возможность расчёта метрик и прогнозирования дефектов с использованием как предложенного,

так и других методов прогнозирования дефектов; а также является платформо (framework), позволяющей решать широкий спектр исследовательских задач по изу чению эволюции исходного кода программного обеспечения на стадии разработки.

В четвёртой главе описываются методика и результаты проведённых вычис лительных экспериментов. Приводятся сведения о программных системах, исходны код которых был использован для прогнозирования дефектов и проверки точно сти прогнозов. Основные метрики рассмотренных программных систем приведены Таблице 1. Выборка программных систем является репрезентативной по различны аспектам: назначение программной системы, используемые в процессе разработк методологии и языки программирования, объём исходного кода, длительность пери ода разработки, число активных разработчиков, число выпущенных версий и т. д. Для каждой из десяти программных систем рассматривается от 4 до 13 версий, что в сумме даёт 70 наборов экспериментальных данных.

Приводится описание наиболее распространённых характеристик, которые мо гут быть использованы для оценки точности предложенного метода прогнозирова ния дефектов. Все эти характеристики могут быть вычислены на основе четырёх

Программная система	Тысяч строк кода	Версий	Файлов	Дефектов	Язык программирования
couchdb	30	6	175	337	Erlang
django	185	4	2052	6810	Python
frund	287	4	1695	142	Fortran, C++, C#
gnome-terminal	48	13	1571	598	C
gnuplot	126	10	313	929	C
httpd	238	4	843	2927	C
nhibernate	354	4	4718	1090	C#
pgadmin3	190	8	796	1209	C++
subtle	29	6	131	652	C
wordpress	204	11	1198	6837	PHP

Таблица 1. Рассмотренные программные системы

Прогноз	Реальные данные		
		Дефектный	Недефектный
	Дефектный	TP (true positive)	FP (false positive) ошибки 1-го рода
	Недефектный	FN (false negative) ошибки 2-го рода	TN (true negative)

Таблица 2. Метрики по результатам проверки прогноза

метрик. Каждая из этих метрик вычисляется как число компонентов, для которых результат проверки прогноза соответствует одному из четырёх возможных вариантов, приведённых в Таблице 2. Для оценки точности прогнозирования используются четыре общепринятые характеристики.

1. Общая точность (*Accuracy*) — есть вероятность того, что компонент будет классифицирован верно:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4}$$

2. Точность (*Precision*) отражает какую долю выборки составляют действительно дефектные компоненты:

$$Precision = \frac{TP}{TP + FP}. \tag{5}$$

3. Полнота (*Recall*) отражает какова доля дефектных компонентов, попавших в выборку:

$$Recall = \frac{TP}{TP + FN}. \tag{6}$$

4. *AUC* — есть вероятность того, что для случайно выбранного компонента из множества компонентов, действительно содержащих дефекты, оценка вероятности наличия дефектов в нём будет выше, чем для случайно выбранного компонента из множества компонентов, действительно не содержащих дефекты. Имеет сложный алгоритм вычисления. Пригодна для сравнения результатов прогнозирования, полученных на разных наборах экспериментальных данных.

Характеристики *AUC* и общая точность (*Accuracy*) выбираются в качестве основных для оценки точности прогнозирования, а точность (*Precision*) и полнота (*Recall*) – в качестве дополнительных. Приводится аргументация выбора данных характеристик.

Описывается методика проведённых экспериментов. В ходе каждого эксперимента рассматривается отдельный набор экспериментальных данных. Каждый такой набор соответствует некоторой версии программной системы и включает исходный код, находившийся в соответствующей версии. Все измеряемые метрики вычисляются на этом множестве исходного кода. Вычисляются вероятности нахождения дефектов в отдельных компонентах и выполняется классификация, в ходе которой каждый компонент классифицируется либо как содержащий необнаруженные дефекты, либо как не содержащий таковых. Далее на основе информации об эволюции исходного кода после выпуска рассматриваемой версии устанавливается в каких компонентах действительно были обнаружены дефекты. Эта информация используется для оценки точности прогнозирования.

Выполняется проверка сформулированных гипотез о закономерностях эвол ции исходного кода программного обеспечения на стадии разработки. Для кажд такой гипотезы выполняется прогнозирование дефектов с использованием двух в риантов предложенного метода прогнозирования дефектов. Первый вариант мето строится, исходя из предположения о верности гипотезы, а второй – о неверност Отдельные гипотезы формулируются по каждой из четырёх используемых характ ристик точности: одна нулевая о том, что характеристика статистически значимо изменяется и две альтернативные о том, что характеристика либо повышается, ли понижается. Для проверки статистических гипотез используется двух-выборочны t-критерий Стьюдента для зависимых выборок с уровнем значимости 0,05. Анал гичным образом формулируются и проверяются гипотезы для сравнения различны методов прогнозирования между собой. На основе полученных результатов, форм руется окончательный вариант метода прогнозирования дефектов, основанный н гипотезах, верность которых получила экспериментальное подтверждение.

Выполняется сравнение предложенного метода прогнозирования с двумя ал тернативными методами. Эти методы являются авторской реализацией часто упом наемых в литературе методов прогнозирования дефектов в компонентах програм ного обеспечения. Первый альтернативный метод основан на выборке компоненто содержащих максимальный объём исходного кода. При этом 20% компонентов ма симального размера классифицируются как содержащие необнаруженные дефекть Второй альтернативный метод основан на использовании логистической регресси и следующих метрик эволюции исходного кода: число строк исходного кода в ко поненте на момент прогноза; число строк исходного кода, добавленных в компонен после предыдущей версии; число строк исходного кода, удалённых из компонент после предыдущей версии; число модификаций компонента после предыдущей ве сии. Результаты сравнения приведены на Рисунках 1 и 2. По результатам можн отметить:

- в большинстве случаев предложенный метод оказывается более точным, че альтернативные;
- в тех случаях, когда лучшим оказывается один из альтернативных методов его преимущество над предложенным методом минимально;
- на отдельных наборах данных оба альтернативных метода демонстрируют зна чительное падение точности, чего не наблюдается для предложенного метода.

Выполняется сравнение предложенного метода с наиболее точными из сущ ствующих методов прогнозирования дефектов. Результаты для этих методов были взяты из литературы. Сравнение выполняется только по характеристике AUC, т. к. только она пригодна для сравнения результатов прогнозирования, полученных на различных наборах данных. Результаты сравнения приведены на Рисунке 3. По р зультатам можно отметить:

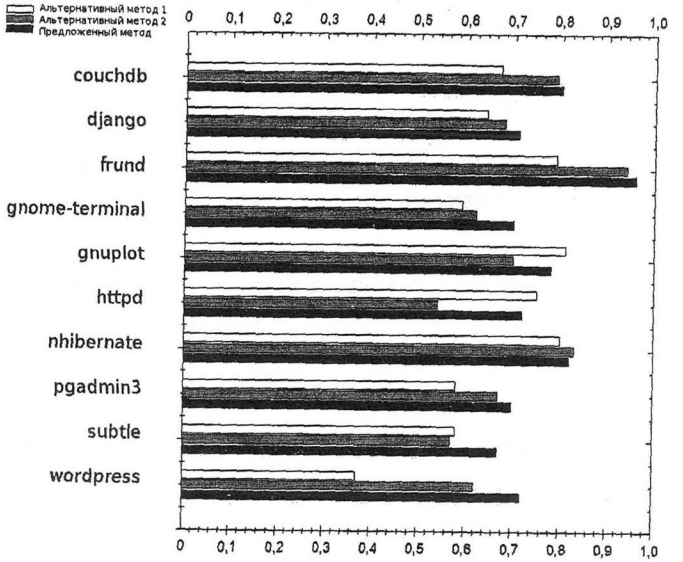

Рис. 1. Сравнение с альтернативными методами по общей точности (*Accuracy*)

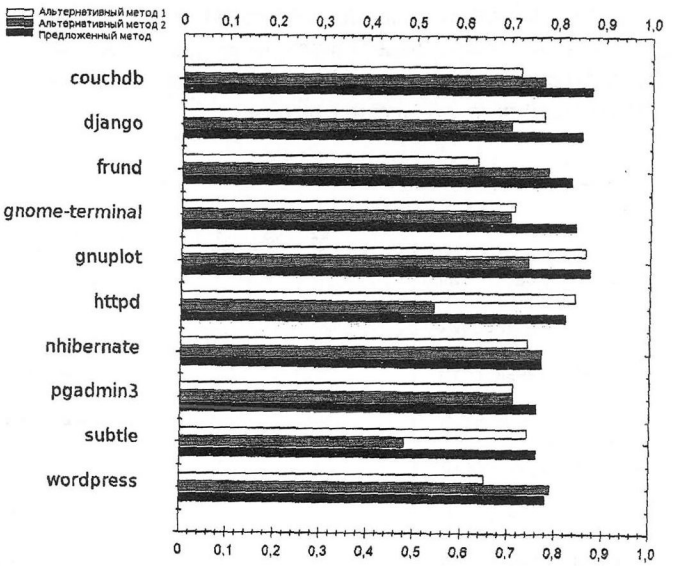

Рис. 2. Сравнение с альтернативными методами по характеристике *AUC*

- по потенциальной максимальной точности предложенный метод уступает всем рассмотренным методам;
- по потенциальной средней точности предложенный метод можно считать примерно эквивалентным всем рассмотренным методам;
- по потенциальной минимальной точности предложенный метод превосходит почти все рассмотренные методы;
- диапазон разброса точности для предложенного метода является самым узким по сравнению с рассмотренными методами, что характеризует метод как один из самых стабильных по характеристике *AUC* методов прогнозирования дефектов в компонентах программного обеспечения.

Перечисляются условия, являющиеся обязательными для применения метода:

- использование системы контроля версий при разработке программного обеспечения;
- следование принципам корректного использования системы контроля версий;
- внесение исправлений отдельных дефектов в систему контроля версий в виде отдельных изменений.

Отмечается, что этим условиям удовлетворяет практически любой проект с открытым исходным кодом, что обеспечивает широкую область применимости предложенного метода прогнозирования дефектов. Даются рекомендации относительно потенциальных путей дальнейшего улучшения метода.

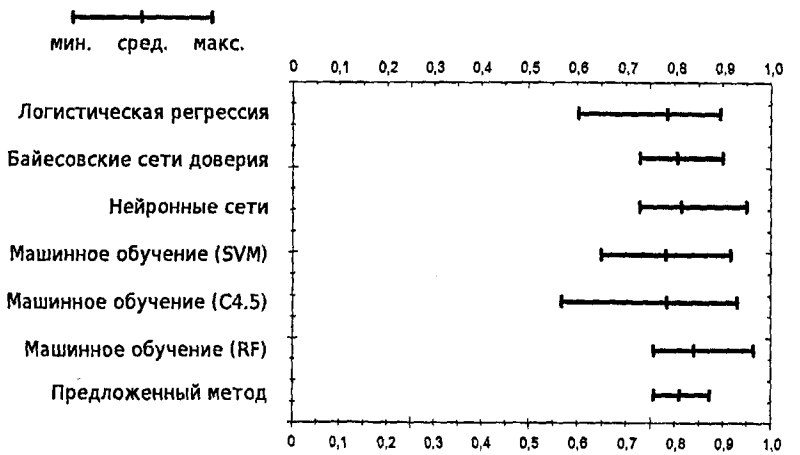

Рис. 3. Сравнение с наиболее точными существующими методами по характеристике *AUC*

Делаются выводы о том, что на основании экспериментальных результатов, можно утверждать, что точность предложенного метода прогнозирования дефектов находится на уровне наиболее точных существующих методов. При этом предложенный метод является независимым от языка программирования и не требует экспертных оценок.

В Заключении перечисляются основные результаты, достигнутые при работе над диссертацией, и даётся оценка её дальнейших перспектив.

Основные результаты работы

При решении поставленных задач получены следующие результаты:

1. Проведён анализ существующих методов прогнозирования дефектов в компонентах программного обеспечения. Обобщены их недостатки.

2. Выдвинут ряд гипотез о закономерностях эволюции исходного кода в программном обеспечении на стадии разработки.

3. Предложены новые метрики исходного кода, которые могут быть использованы при прогнозировании дефектов и представляют различную информацию об эволюции исходного кода в программном обеспечении на стадии разработки.

4. Предложен новый метод прогнозирования дефектов в компонентах программного обеспечения на стадии разработки, который обладает следующими свойствами: независимость от языка программирования; отсутствие потребности в экспертных оценках и какой-либо другой информации, которая не может быть получена из системы контроля версий; независимость от наличия и количества версий системы.

5. Разработаны новые алгоритмы для извлечения информации из систем контроля версий Subversion и Git, расчёта на её основе различных метрик исходного кода.

6. Разработан набор утилит MSR Tools, включающий платформу (framework) для выполнения анализа эволюции исходного кода программного обеспечения на стадии разработки, а также утилиты для вычисления различных метрик исходного кода, визуализации и прогнозирования дефектов. Данный набор утилит размещён в сети Интернет (`http://msr.sourceforge.net`) и доступен вместе с исходными кодами под лицензией BSD. Также в сети Интернет размещены использованные в ходе экспериментов наборы данных, что делает все проведённые эксперименты легко воспроизводимыми с целью дальнейшего улучшения предложенного метода прогнозирования дефектов, разработки новых методов

или выполнения любых других исследований, связанных с изучением эволюции исходного кода программного обеспечения на стадии разработки.

7. Проведены вычислительные эксперименты, по результатам которых ряд гипотез о закономерностях эволюции исходного кода программного обеспечения на стадии разработки получил подтверждение.

8. Проведены вычислительные эксперименты, по результатам которых сделаны выводы о том, что точность предложенного метода прогнозирования дефектов является достаточной, для его использования при решении задач управления процессом разработки.

Список публикаций

Публикации в изданиях, включённых в Перечень ВАК:

1. Кирносенко, С. И. Идентификация исправляющих ревизий в системах контроля версий / С. И. Кирносенко, В. С. Лукьянов // Изв. ВолгГТУ. Серия «Актуальные проблемы управления, вычислительной техники и информатики в технических системах». Вып. 9 : межвуз. сб. науч. ст. / ВолгГТУ. - Волгоград, 2010. - № 11. - С. 146-149.

2. Кирносенко, С. И. Подсчёт плотности ошибок для различных артефактов кода / С. И. Кирносенко, В. С. Лукьянов // Международный отраслевой журнал «Век качества. Связь: сертификация, управление, экономика» / ООО «Азбука». - Москва, 2011. - № 4. - С. 70-72.

3. Кирносенко, С. И. Прогнозирование обнаружения дефектов в программном обеспечении / С. И. Кирносенко, В. С. Лукьянов // Программные продукты и системы / НТП «Фактор». - Тверь, 2011. - № 3. - С. 67-71.

Прочие публикации:

4. Кирносенко, С. И. Оценка надёжности программного обеспечения на основе итерационной модели // Технологии Microsoft в теории и практике программирования: матер. конф. (Нижний Новгород, 19-20 марта 2008 г.) / Нижегор. гос. ун-т им. Н.И. Лобачевского. - Н. Новгород, 2008. - С. 166-169.

5. Кирносенко, С. И. Подсчет плотности ошибок кода // Наука. Технологии. Инновации. Материалы всероссийской научной конференции молодых ученых в 7-ми частях. Часть 1. Новосибирск: НГТУ, 2008. — С. 15–16.

6. Кирносенко, С. И. Анализ надёжности программного обеспечения // XIII региональная конференция молодых исследователей Волгоградской области, г. Волгоград, 11-14 нояб. 2008 г.: тез. докл. / ВолгГТУ [и др.]. - Волгоград, 2009. - С. 204-205.

7. Кирносенко, С. И. Подсчёт плотности ошибок кода для различных видов исправляющих модификаций // 6-я Межрегиональная научно-техническая конференция студентов и аспирантов. Сборник трудов. Смоленск: Универсум, 2009. — С. 69–71.

8. Кирносенко, С. И. Подсчёт плотности ошибок кода / С. И. Кирносенко // Технологии Microsoft в теории и практике программирования : матер. конф. (Нижний Новгород, 11-12 марта 2009 г.) / Нижегород. гос. ун-т им. Н. И. Лобачевского. - Н. Новгород, 2009. - С. 193-197.

9. Кирносенко, С. И. Извлечение данных для подсчёта плотности ошибок кода // Приоритетные направления современной российской науки глазами молодых учёных. Рязань: Ряз. гос. ун-т им. С. А. Есенина, 2009. - С. 159-161.

10. Кирносенко, С. И. Коллективная разработка и надёжность / С. И. Кирносенко, В. С. Лукьянов // XIV региональная конференция молодых исследователей Волгоградской области (Волгоград, 10-13 нояб. 2009 г.) : тез. докл. / ВолгГТУ [и др.]. - Волгоград, 2010. - С. 204-206.

11. Кирносенко, С. И. Представление истории изменений исходного кода для задач подсчёта метрик // Актуальные вопросы современной техники и технологии. Сборник докладов. Липецк: Издательский центр «Де-факто», 2010. - С. 39-41.

12. Кирносенко, С. И. Классификация ревизий в системах контроля версий // Теоретические вопросы разработки, внедрения и эксплуатации программных средств: материалы Всероссийской научно-практической конференции. Орск: Издательство ОГТИ, 2011. - С. 45-47.

Подписано в печать 25.01.2012 г. Заказ № 30 . Тираж 100 экз. Печ. л. 1,0
Формат 60 x 84 1/16. Бумага офсетная. Печать офсетная.

Типография ИУНЛ
Волгоградского государственного технического университета.
400005, г. Волгоград, просп. им. В.И. Ленина, 28, корп. №7.