На правах рукописи

ЕПИФАНОВ Николай Анатольевич

Методы реализации регрессионного тестирования по расширенным тестовым наборам

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание учёной степени кандидата технических наук

Санкт-Петербург – 2003

Работа выполнена в государственном образовательном учреждении высшего профессионального образования «Санкт-Петербургский государственный политехнический университет».

Научный руководитель: кандидат технических наук, профессор
Котляров Всеволод Павлович

Официальные оппоненты: доктор технических наук, профессор
Лисс Александр Рудольфович
кандидат технических наук
Пинаев Дмитрий Владимирович

Ведущая организация: НПО «Импульс».

Защита состоится «26» февраля 2004 года в 16^{00} на заседании диссертационного совета Д 212.229.18 при ГОУ ВПО «Санкт-Петербургский государственный политехнический университет» по адресу: 195251, Санкт-Петербург, Политехническая ул., 29, корпус 9, аудитория 325.

С диссертацией можно ознакомиться в библиотеке ГОУ ВПО «Санкт-Петербургский государственный политехнический университет».

Автореферат разослан «25» января 2004 года.

Учёный секретарь
диссертационного совета Д 212.229.18 **Шашихин** В.Н.

1. ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

1.1. Актуальность работы

Программы подвержены модификациям, как бы хорошо они ни были разработаны первоначально. В процессе разработки, на этапе тестирования и при последующей эксплуатации выявляются дефекты, которые должны быть исправлены. К переработке программ также приводят ошибочные и изменённые требования. Кроме того, новые области применения старых программ требуют новых функциональных возможностей, не учтённых в требованиях изначально. Контроль над изменениями – критический фактор для сохранения полезности программ.

С развитием программного обеспечения сопровождение становится наиболее дорогим компонентом разработки программных систем. Исследования Майерса, Линца, Свенсона и Шаха показали, что от 1/2 до 2/3 затрат на протяжении срока службы системы программного обеспечения уходит на сопровождение. На этап сопровождения приходятся и наибольшие затраты, связанные с исправлением обнаруженных ошибок. Проведённый Вейнбергом анализ самых дорогих ошибок за всю историю программирования показал, что худшие три из них были вызваны изменением ровно одной строки кода, который не был протестирован после внесения изменения.

Перечисленные обстоятельства иллюстрируют актуальность проблемы эффективного обнаружения и исправления дефектов программного изделия в процессе сопровождения.

Важным различием между регрессионным тестированием и тестированием в процессе разработки является наличие в процессе регрессионного тестирования готового набора тестов, доступного для повторного использования. Одна из стратегий регрессионного тестирования допускает повторный запуск всех этих тестов, но такой метод потребляет чрезмерно много времени и ресурсов. Альтернативный метод, выборочное повторное тестирование, выбирает из старого набора тесты, которые считает необходимыми для тестирования изменённой программы. Эти тесты дополняются, в случае необходимости, новыми, возможно для того, чтобы удовлетворить некоторый критерий покрытия кода. Выборочный подход даёт выигрыш тогда, когда стоимость отбора и прогона подмножества тестов меньше, чем стоимость выполнения тестов, которые удаётся опустить.

Во время, когда система подвергается изменению, возникает проблема определения множества тестов, подлежащих модификации, и способов их изменения. Инструментальные средства поддержки разработки программного обеспечения, успешно применяемые при тестировании, далеко не всегда применимы к регрессионному тестированию. Большинство коммерческих средств автоматизации тестирования обеспечивает не более чем возможность

сохранять существующие тесты и запускать их повторно после каждого изменения. У существующих инструментальных средств отсутствует возможность какого-либо автоматического отбора тестов, нет и средств автоматизации оценки требуемых для регрессионного тестирования затрат. Методики выборочного регрессионного тестирования пока ещё не нашли своего воплощения в программных средствах промышленного качества.

В существующей практике методы отбора регрессионных тестов основаны на субъективном выборе подмножества из существующего набора тестов. Если программа не тестируется адекватно существующим набором тестов, то маловероятно, что подмножество этого набора будет адекватным для тестирования изменённой версии программы. Более того, даже если существующий набор тестов был адекватен для тестирования предыдущей версии программы, то он не обязательно будет адекватен для тестирования модифицированной программы. Существующие методы отбора регрессионных тестов сосредоточены только на проблеме отбора тестов из существующего набора, в то время как перспективный универсальный метод регрессионного тестирования не должен ограничиваться исключительно существующими тестовыми наборами.

Наконец, существующие подходы к регрессионному тестированию обычно оставляют за бортом такие возможности, как уменьшение объёма тестируемой программы, методы упорядочения и функции предсказания целесообразности отбора тестов, так что пользователю не всегда ясны основания, которыми руководствуются при проведении выборочного регрессионного тестирования.

1.2. Цель работы

Диссертационная работа посвящена созданию методики, позволяющей производить отбор тестов для проведения регрессионного тестирования, а также указывать области исходного кода и группы функциональных требований, для которых необходима разработка дополнительных тестов. Методика основана на профилировании исходного кода системы и анализе путей в графе системы, активируемых отдельными тестами. Методика обеспечивает применение сопутствующих технологий, таких как предсказание целесообразности регрессионного тестирования.

1.3. Научная новизна работы

1. Разработана методика выборочного регрессионного тестирования. Предложенный подход является безопасным и целесообразным, т.к. он отбирает 100% тестов, обнаруживающих ошибки, и оправдывает расходы на собственное применение. Он применим для регрессионного тестирования больших промышленных программ объёмом до 10^7 строк кода, поскольку обеспечивает линейный рост времени работы алгоритма отбора в

зависимости от объёма тестируемой программы (является эффективным). Предложенная методика применима для широкого круга программ, написанных на языках C, C++ и C#.

2. Разработана методика порождения дополнительных тестов при регрессионном тестировании. Методика находит ситуации, «подозрительные» с точки зрения возможного наличия ошибок, и рекомендует пути их дальнейшего тестирования. Она применима как для случая модульного тестирования с применением структурных критериев покрытия кода, так и для системного тестирования, ориентированного на требования технического задания.
3. Разработана методика расширенного регрессионного тестирования. В отличие от алгоритма Ротермела – Харролд, методика предусматривает использование разработанного метода порождения дополнительных тестов, алгоритма упорядочения и функции предсказания целесообразности.
4. Создан инструментарий, поддерживающий процесс автоматизации регрессионного тестирования.
5. Предложенные методики и разработанный инструментальный пакет были применены при регрессионном тестировании промышленных программных проектов.

1.4. Апробация работы

Основные положения и результаты работы докладывались на следующих конференциях:

- XXX Юбилейная неделя науки СПбГТУ;
- XXXI неделя науки СПбГТУ;
- Motorola Technology Day 2000;
- Motorola Technology Day 2001;
- International Workshop on Program Understanding, 2003.

По материалам диссертации опубликовано 6 печатных работ.

Разработанные методы и программные средства внедрены в производственный процесс ЗАО «Моторола ЗАО», ЗАО «Северо-Западная Лаборатория» и научно-производственной фирмы «Юпитер».

1.5. Структура и объём работы

Работа содержит 4 главы, введение и заключение. Объём работы 149 страниц, количество иллюстраций – 39, список использованной литературы состоит из 143 наименований.

2. СОДЕРЖАНИЕ РАБОТЫ

В **первой главе** диссертации:

- проанализированы проблемы, характерные для этапа сопровождения программного обеспечения;
- обоснована необходимость регрессионного тестирования при сопровождении программ;
- перечислены цели и задачи регрессионного тестирования;
- рассмотрены виды регрессионного тестирования, которые сопоставлены с типами изменений кода, видами обычного тестирования и различными методами сопровождения;
- объяснены принципы выборочного регрессионного тестирования;
- приведена классификация методов выборочного регрессионного тестирования и классификация тестов в процессе отбора;
- детально рассмотрены особые виды регрессионного тестирования, такие как регрессионное тестирование межмодульных зависимостей и регрессионное тестирование объектно-ориентированных программ;
- рассмотрены технологии уменьшения объёма тестируемой программы, упорядочения тестов и предсказания целесообразности;
- приведён обзор методов отбора регрессионных тестов.

На основании анализа приведённых в главе данных сформулированы выводы о свойствах различных классов методов и средств поддержки регрессионного тестирования:

1. В литературе описывается довольно много разнообразных средств поддержки регрессионного тестирования. Большинство из них ориентировано на функциональное и структурное тестирование.
2. Методы префикса пути и классов данных поддерживают гибридное тестирование.
3. Большинство методов пригодно только для модульного, или только для интеграционного, или только для системного тестирования. Универсальных методов найдено не было.
4. Метод классов данных предусматривает порождение дополнительных тестов по критерию структурного тестирования.
5. Не найдено публикаций, посвящённых критериям порождения дополнительных тестов при функциональном тестировании.
6. Существующие модели оценки методов выборочного регрессионного тестирования не учитывают влияния тестируемых программ на преимущества и недостатки отдельных методов. Из практики известно, что для некоторых программ целесообразнее применение одного метода, для других – другого. Поэтому исследования с целью создания новых методов не потеряли своей актуальности.

7. Ни одно из рассмотренных в обзоре программных средств поддержки регрессионного тестирования не находится в промышленном использовании, что указывает на трудность его создания. Частичным объяснением этому может служить тот факт, что любое средство поддержки регрессионного тестирования должно работать в паре с программным средством поддержки обычного тестирования, поскольку при регрессионном и исходном тестировании должны использоваться одни и те же методы. Другая причина в том, что оценочное время обработки любым из рассмотренных методов большой промышленной программы неприемлемо велико.
8. Существующие методы регрессионного тестирования не используют методов упорядочения и функцию предсказания целесообразности.
9. Безопасные методы отбора тестов являются более универсальными, чем небезопасные.
10. Технология уменьшения объёма тестируемой программы непригодна для индустриального применения.

Таким образом, анализ существующих на сегодняшний день программных средств поддержки регрессионного тестирования позволяет констатировать отсутствие программных средств безопасного отбора тестов для больших проектов, отсутствие методов порождения дополнительных тестов для случая функционального регрессионного тестирования и неполноту существующих методик регрессионного тестирования, не учитывающих технологии, не связанные с отбором тестов, что и определило цели настоящей работы.

Во **второй главе** описан предложенный в работе метод покрытия точек использования неисполняемых определений, метод порождения новых тестов на основании «подозрительных» состояний и методика расширенного регрессионного тестирования, а также дана оценка предложенных методов.

Тестирование программы P по некоторому критерию C заключается в покрытии множества M покрываемых элементов m_j, соответствующих элементам и/или взаимосвязям между элементами некоторой модели программы: $M(P, C) = \{m_1, m_2 \dots m_k\}$. Пусть выполняемые тесты t_j составляют упорядоченное множество тестов $T = \{t_1, t_2, \dots, t_n\}$. Каждому тесту t_j соответствует некоторое подмножество покрываемых им элементов $MT(P, C, t_j)$. В этих условиях методика решения задачи выборочного регрессионного тестирования путём покрытия точек использования неисполняемых определений включает следующие этапы:

1. С помощью анализа профиля программы для каждого существующего теста t_j собирается информация о том, какие строки исходного кода продукта покрываются этим тестом, то есть создаётся множество $MT(P, C, t_j)$.

2. Множество MT(P, C, t_j) хранится под управлением системы контроля версий (например, CVS или ClearCase) и обновляется для каждой версии продукта.
3. Когда очередная версия продукта передаётся на тестирование, с помощью системы контроля версий создаётся множество добавленных, удалённых и изменённых строк исходного кода (множество ΔP):

$$\Delta P = (P' \setminus P) \cup (P \setminus P')$$

4. Множество ΔP расширяется за счёт раскрытия макроопределений и других подобных им неисполняемых операторов.
5. Для повторного прогона выбираются только такие тесты t_j, для которых во множестве покрываемых ими элементов множества MT(P, C, t_j) найдётся затронутый изменением элемент m_k. Такие тесты t_j образуют множество T':

$$T' = \{t_j \mid MT(P, C, t_j) \cap \Delta P \neq \varnothing\}$$

6. Множество T' упорядочивается с тем, чтобы те тесты, у которых количество изменённых строк в пути исполнения наибольшее, выполнялись первыми. Подход инвариантен к применению других критериев упорядочения.

Предложенный метод является безопасным и обеспечивает приемлемую для промышленных применений универсальность и точность, то есть избегает выбора тестов из T, на которых результат выполнения изменённой программы не будет отличаться от результата её первоначальной версии. Время работы метода оценивается как O(|T| * n). Методы отбора тестов, обладающие сходными характеристиками, в литературе не описаны.

Введём понятие состояния тестируемой программы s как совокупности значений некоторого подмножества глобальных и локальных переменных. При создании новых тестов будем рассматривать состояния программы перед запуском теста (s_0) и после его окончания (s_j). Информацию об этих состояниях необходимо собирать для каждого теста по результатам запуска на предыдущей версии продукта. В таком случае, методика порождения новых тестов при регрессионном тестировании на основе анализа «подозрительных» состояний сводится к следующей последовательности шагов:

1. Вычисляется список глобальных и локальных переменных, определяющих состояние программы s.
2. С помощью анализа профиля программы, полученного на предыдущей версии продукта i - 1, для каждого существующего теста t_j собирается информация о состояниях программы перед запуском теста и после его окончания (т.е. s_0 и s_j). Множество таких состояний обозначается S_{i-1}:

$$S_{i-1} = s_0 \cup \{s_j \mid \forall j\}$$

3. Вновь разработанные и выбранные из множества T’ регрессионные тесты выполняются на текущей версии продукта i. По аналогии с S_{i-1} вычисляется множество S_i, которое сохраняется под управлением системы контроля версий.
4. Множество новых по сравнению с предыдущими версиями состояний N_i является «подозрительным» с точки зрения наличия ошибок. Правило вычисления множества N_i описывается следующей формулой:

$$N_i = S_i \setminus S_{i-1}$$

5. Если множество N_i включает состояния, в которых дальнейшая работа продукта невозможна в соответствии со спецификацией, то необходимо проанализировать, создаются ли рассматриваемые состояния в результате выполнения тестов, проверяющих нештатные режимы работы продукта, или каких-либо других тестов. В последнем случае фиксируется ошибка. Нештатные состояния исключаются из множества N_i.
6. Если новых состояний, допускающих продолжение выполнения программы, не обнаружено, т.е. $N_i = \varnothing$, перейти к шагу 9.
7. Для каждого состояния множества N_i вычисляется вектор его отличия от исходного состояния s_0, т.е. переменные, изменённые по сравнению с s_0.
8. С учётом информации об изменённых переменных модифицируется множество изменённых строк исходного кода ΔP и используется метод покрытия точек использования неисполняемых определений.
9. Шаги 3-8 выполняются повторно до достижения состояния $N_i = \varnothing$ либо до истечения времени, отведённого на регрессионное тестирование. Руководствуясь методом разбиения результатов тестирования на классы эквивалентности, инженер, ответственный за тестирование, может также принять решение о прекращении цикла тестирования, если ни один тест из созданных на очередном этапе не принадлежит к новому классу эквивалентности.

Следует отметить, что при поддержке тестирования средствами автоматизации, когда новые тесты получаются автоматически в результате суперпозиции уже имеющихся, целесообразно в качестве исходных тестов, то есть тех «кирпичиков», из которых будут строиться тесты в дальнейшем, брать тесты, заключающие всего одну элементарную проверку. Это поможет избежать избыточности при многократном слиянии тестов.

Предлагается методика расширенного регрессионного тестирования, которая включает помимо стандартных компонентов функцию предсказания целесообразности, метод упорядочения тестов и вышеописанную методику порождения новых функциональных тестов. Методика состоит из следующих этапов:

1. Использование функции предсказания целесообразности. Если прогнозируемое количество выбранных тестов больше, чем порог целесообразности, провести повторный прогон всех тестов. В противном случае перейти к шагу 2.
2. Идентификация изменений в программе. Метод покрытия точек использования неисполняемых определений, в частности, предусматривает на этом этапе построчное сравнение двух версий программы и создание множества изменённых строк.
3. Отбор и упорядочение подмножества исходных тестов для повторного выполнения на изменённой программе. Разработанный безопасный метод выбирает тесты, покрывающие точки использования неисполняемых определений. Упорядочение тестов рекомендуется производить по критерию количества активируемых ими изменённых сущностей. Возможно также конкретное указание числа тестов, выполнения которых достаточно для соответствия какому-либо критерию минимизации.
4. Применение выбранного подмножества для регрессионного тестирования изменённой программы.
5. Создание дополнительных тестов при функциональном регрессионном тестировании. Например, метод «подозрительных» состояний требует порождения новых тестов для всех состояний, которые не наблюдались в ходе тестирования исходной программы. Согласно этому методу, новые тесты получаются в результате суперпозиции имеющихся тестов.
6. Применение новых тестов для тестирования изменённой программы и обновление базы данных тестов.

Конструктивность и эффективность предложенных методик подтверждена при их реализации и пилотировании на реальных программных системах.

В **третьей главе** рассматривается программная система поддержки регрессионного тестирования, описаны её компоненты, известные ограничения и сценарий использования, а также дана оценка программной реализации предложенных методов.

Структура системы поддержки регрессионного тестирования представлена на рис. 1.

Группа программ профайлера использует общий грамматический анализатор, обеспечивающий базовую функциональность. Программа profiler обнаруживает строки кода, содержащие использование изменённых макроопределений по всему тексту тестируемой программы; найденные строки присоединяются к множеству ΔP. Программы profivar и agent обеспечивают фиксацию в протоколе тестирования значений глобальных и видимых локальных переменных из функции тестируемой программы, содержащей цикл обработки

событий. Программа profiadd расширяет множество ΔP за счёт строк кода, где используются переменные, изменившиеся по сравнению с каким-либо известным состоянием программы.

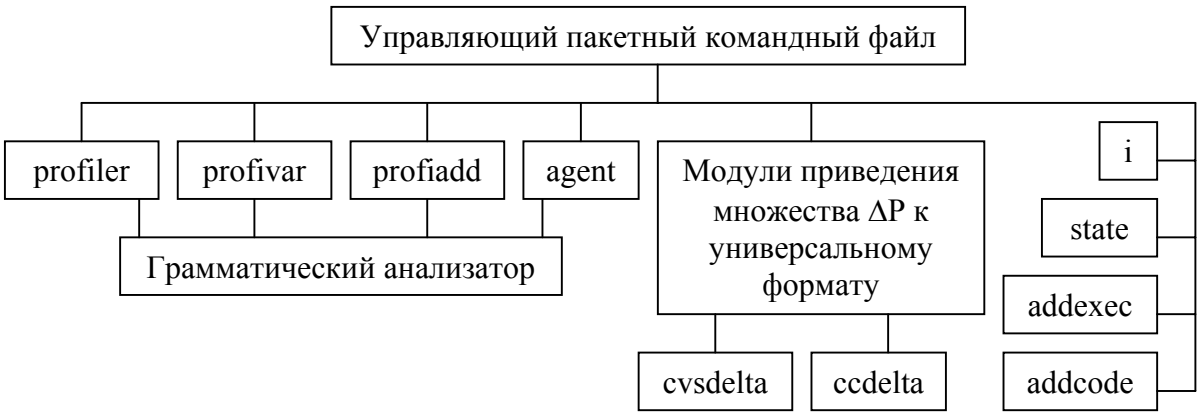

Рис. 1. Структура системы поддержки регрессионного тестирования.

Исходный код обеих версий тестируемой программы хранится под управлением системы контроля версий. Система поддержки регрессионного тестирования ориентирована на использование систем CVS и ClearCase. Средствами одной из этих систем создаётся файл различий между версиями. Модули приведения множества ΔP к универсальному формату (cvsdelta для CVS или ccdelta для ClearCase) вычисляют список добавленных, изменённых и удалённых строк исходного кода, который является удобной формой представления множества ΔP. Модуль получения списка тестов, подлежащих повторному запуску, реализованный как программа i, сопоставляет расширенное множество ΔP с результатами прогона тестов из множества T на предыдущей версии программы. Если в ходе выполнения какого-либо теста t_i получала управление хотя бы одна строка, входящая во множество ΔP, тест t_i отбирается для повторного запуска. Этот модуль позволяет также для каждого состояния указать тесты, запуск которых необходим. Модуль определения новых состояний и списка изменённых переменных, реализованный как программа state, анализирует список состояний программы, полученный в результате запуска тестов из множества T’ на профилированной версии программы, и для каждого ранее не наблюдавшегося состояния вычисляет список переменных, изменившихся по сравнению с каким-либо известным состоянием. Модуль создания списка рекомендованных новых тестов, реализованный как программа addexec, создаёт этот список в форме, удобной для восприятия человеком. Наконец, модуль автоматического добавления операторов вывода, реализованный как программа addcode, позволяет добавлять в тестируемую программу операторы вывода, необходимые для профилирования.

Реализация метода покрытия точек использования неисполняемых определений средствами системы поддержки регрессионного тестирования представлена на рис. 2.

Реализация метода «подозрительных» состояний средствами системы поддержки регрессионного тестирования представлена на рис. 3. Выходные данные каждой программы-обработчика доступны пользователю, что позволяет контролировать промежуточные результаты работы системы. К примеру, можно исключить из рассмотрения переменные, которые, хотя и изменяются в ходе выполнения программы, не влияют на её конечное состояние. Архитектура системы позволяет легко расширять функциональность; например, для поддержания какой-либо новой системы контроля версий достаточно просто создать программу, аналогичную cvsdelta и ccdelta. Остальные модули системы поддержки регрессионного тестирования можно использовать без изменений.

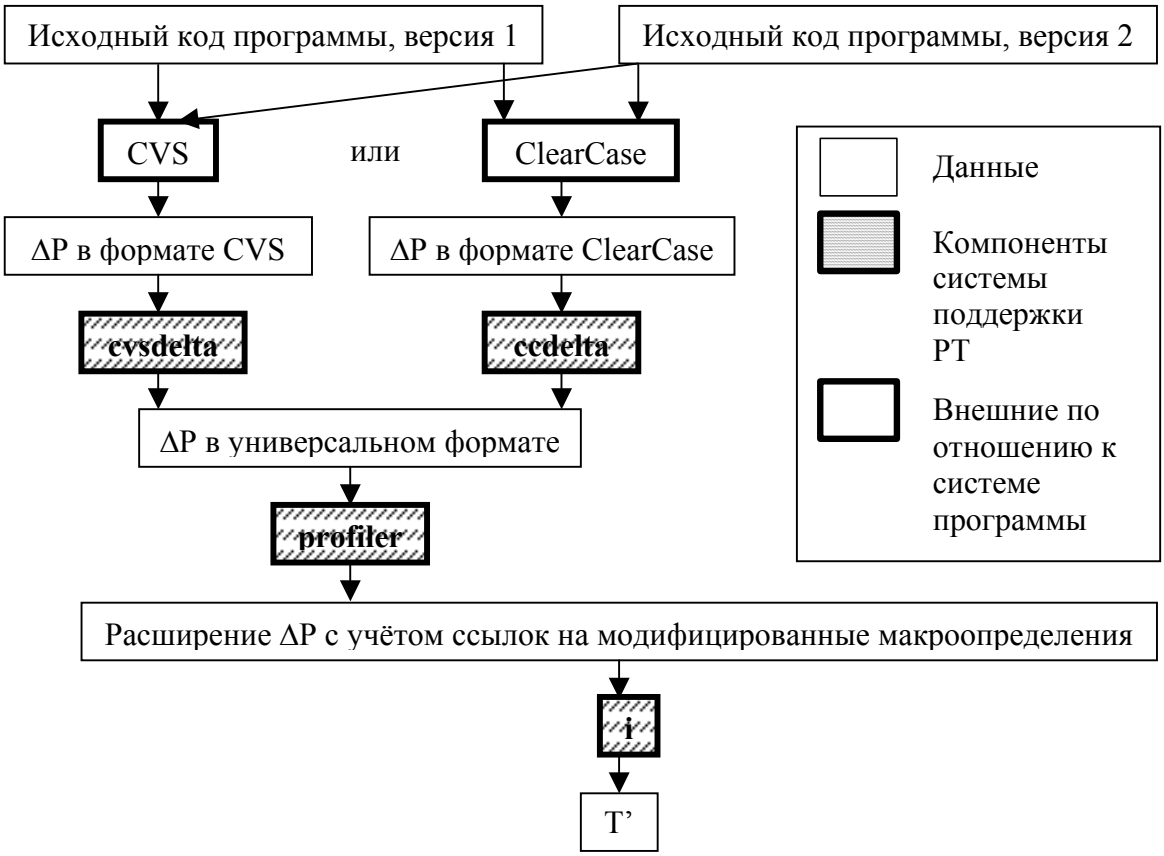

Рис. 2. Метод покрытия точек использования неисполняемых определений.

В **четвёртой главе** приводятся результаты анализа применения разработанных методов и средств поддержки регрессионного тестирования программных проектов.

При помощи предложенных методов были успешно определены наборы регрессионных тестов для трёх программных продуктов, использующихся уже в течение нескольких лет для управления телекоммуникационным оборудованием. Сведения об этих продуктах суммированы в таблице 1.

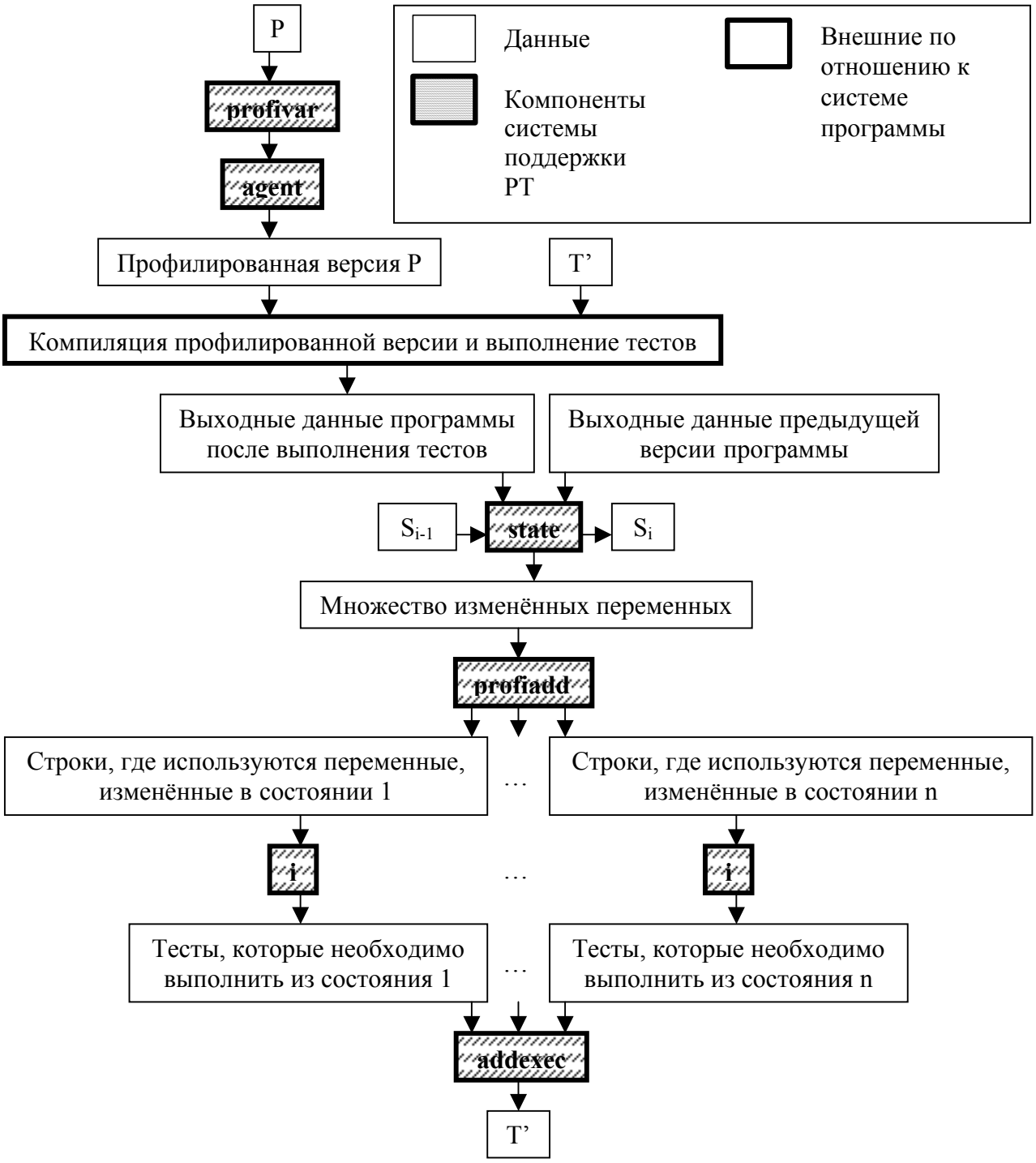

Рис. 3. Реализация метода «подозрительных» состояний.

	Область применения	Объём исходного кода
Проект 1	Управление базовой станцией	10 KLOC
Проект 2	Управление контроллером базовых станций	30 KLOC
Проект 3	Управление контроллером базовых станций	40 KLOC

Таблица 1. Проекты, использовавшиеся для исследований.

С целью оценки метода покрытия точек использования неисполняемых определений проводилась проверка его поведения в условиях наличия изменений различного объёма. Было также проведено его сравнение с другими известными методами. Результаты исследования на программе объёмом 10^5 строк кода приводятся на рис. 4, где показана зависимость количества исполняемых тестов от изменений в коде.

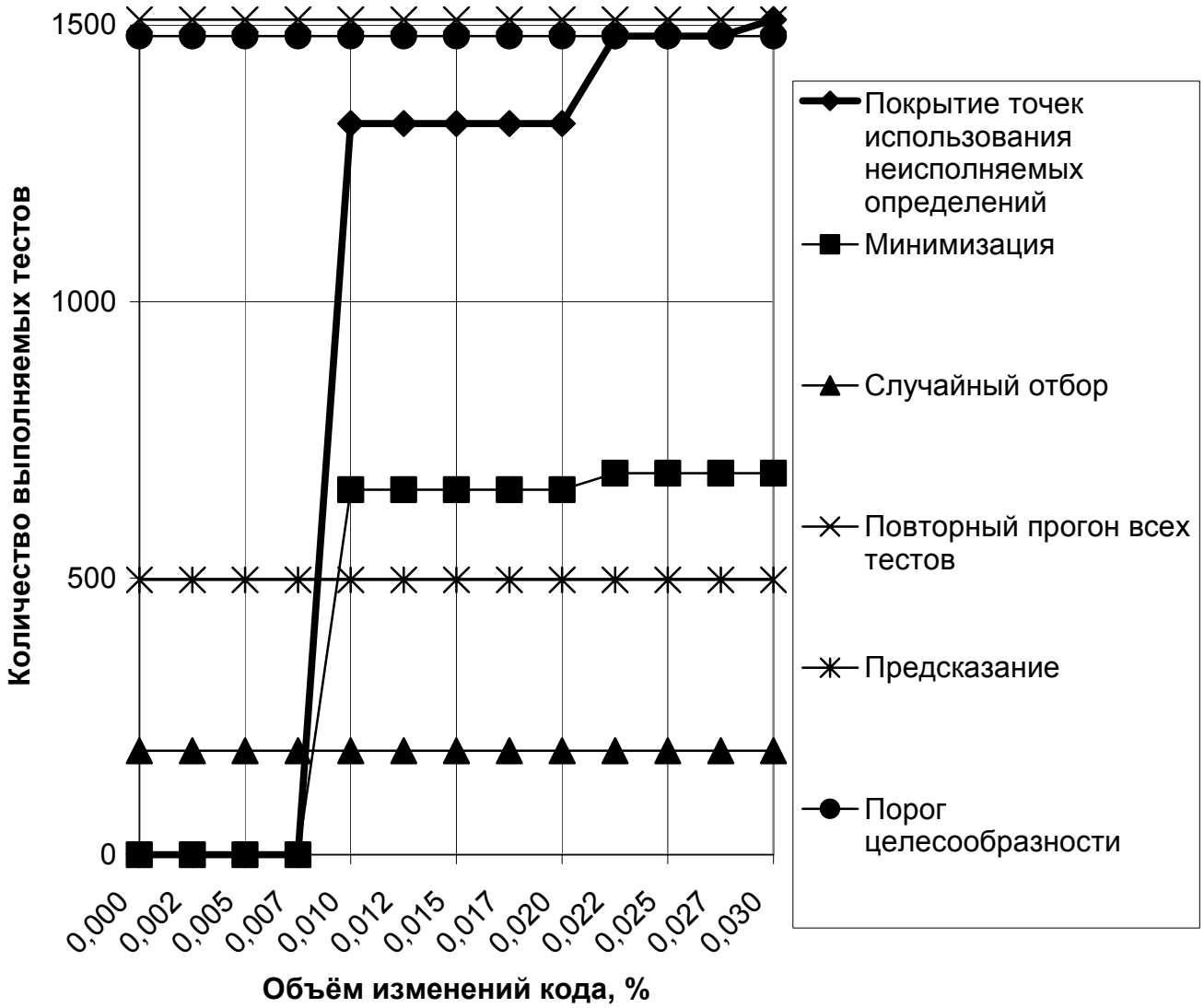

Рис. 4. Зависимость количества исполняемых тестов от изменений в коде.

Внесение изменений осуществлялось в соответствии с различными вариациями функциональных требований. Изменения оказывали влияние и на поток управления программы и на её поток данных. Ввиду того, что время исполнения одного теста намного превышало время работы системы поддержки регрессионного тестирования, значение порога целесообразности оказалось равным 1480, т.е. очень близким к 1510 – общему количеству тестов в наборе. Применение функции предсказания целесообразности показало, что среднее

количество выбранных тестов должно составить 497, то есть отбор тестов целесообразен. В этих условиях проводилось сравнение метода минимизации, случайного метода и метода повторного прогона всех тестов с предложенным в данной работе методом покрытия точек использования неисполняемых определений.

Из графика видно, что чем больше было сделано изменений в исходном коде продукта, тем большее количество регрессионных тестов требовалось для их покрытия с помощью детерминированного выборочного метода. Предложенный метод покрытия точек использования неисполняемых определений показывает приемлемую производительность и на большей части рассмотренного интервала позволяет экономить средства за счёт сокращения времени на тестирование. Другой детерминированный метод, метод минимизации, позволяет добиться ещё более впечатляющего сокращения объёма набора тестов, но не является безопасным и не может быть рекомендован для широкого круга проектов. Случайный метод позволяет добиться результатов, сходных с результатами метода минимизации, но не требует никаких затрат на анализ при отборе тестов. Этот метод также не является безопасным и на практике может составлять конкуренцию только методу минимизации. Наконец, метод повторного прогона всех тестов является безопасным, но в большинстве случаев не позволяет экономить средства так, как метод покрытия точек использования неисполняемых определений. В рассмотренном примере рекомендуется применять метод покрытия точек использования неисполняемых определений при объёме изменений до 0.02% от общего объёма кода и метод повторного прогона всех тестов при объёме изменений свыше 0.03%. При объёме изменений от 0.02% до 0.03% можно использовать любой из этих методов.

Сделанные наблюдения позволяют рекомендовать разработанный метод к использованию при регрессионном тестировании проектов того же класса, что и рассмотренный в примере, а именно для проектов, объём кода которых составляет 100 KLOC и более, а количество тестов в наборе превышает 10^3.

Предложенный в работе метод порождения новых тестов был исследован на предмет вычисления зависимости между объёмом изменений и числом рекомендованных новых тестов (рис. 5). Для проведения этого исследования использовалось то же прикладное ПО, что и для построения графика на рис. 4. Легко заметить, что в исследуемой системе для данных (небольших) объёмов изменений возникновение новых ситуаций не зависит от объема измененного кода. Изменения в коде программного продукта, не превышающие 0.1% от его объёма, приводят к одинаковым изменениям в его состоянии. Наличие случайных выбросов может показывать, что некоторые переменные в системе используются без

предварительной инициализации. Таким образом, метод регрессионного тестирования по расширенным тестовым наборам позволяет не только определить множество необходимых регрессионных тестов, но и может указать потенциально опасные места в программе.

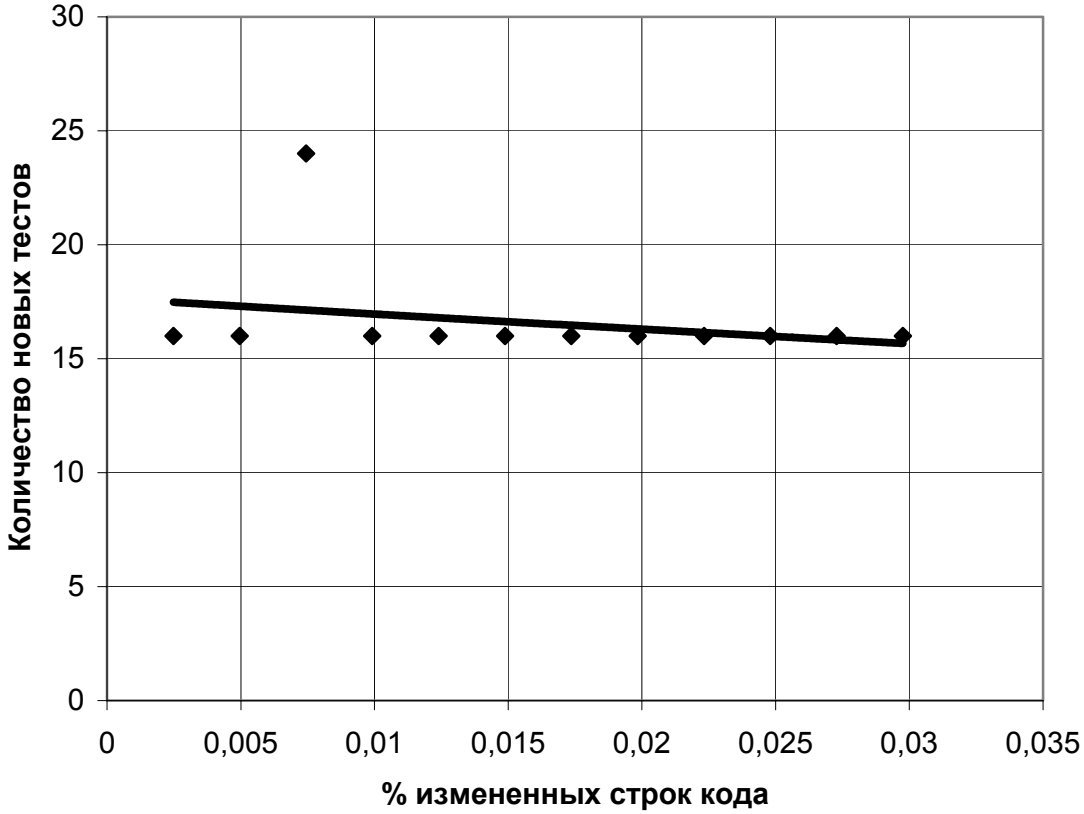

Рис. 5. Зависимость количества новых тестов от изменений в коде.

Рис. 6 показывает, что производительность системы поддержки регрессионного тестирования является линейной характеристикой. Это говорит о том, что при её работе не происходит постоянного накопления данных, которое могло бы привести к переполнению памяти и нестабильной работе программ. Таким образом, производительность системы может быть оценена как 42 килобайта в секунду. К примеру, программа, содержащая 10^5 строк и в среднем по 30 символов в строке, будет обрабатываться около 70 секунд.

В отношении тестируемых программных проектов были получены следующие основные результаты:

1. Удалось добиться сокращения времени на регрессионное тестирование программного проекта в среднем на 12% (1322 теста вместо 1510).
2. По результатам применения предложенных методов были разработаны новые тесты, которые позволили выявить 100% искусственно внесённых ошибок.

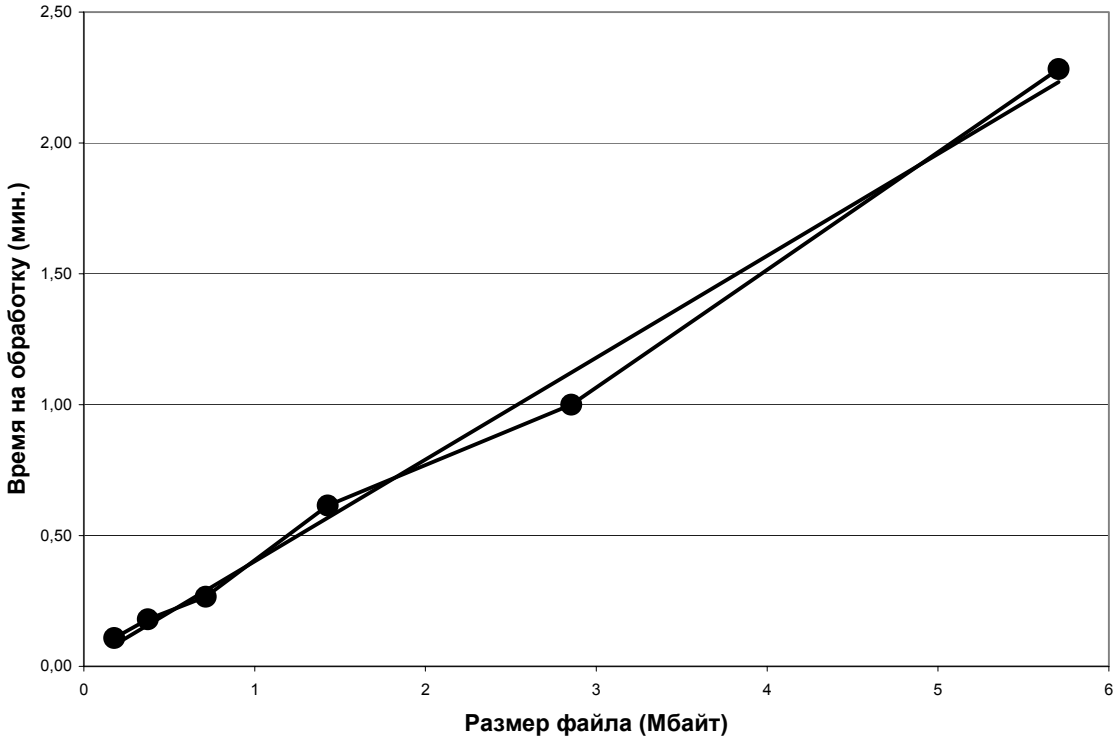

Рис. 6. Производительность системы поддержки регрессионного тестирования.

3. Регрессионное тестирование программного проекта осуществлялось за конечное число итераций (4) с использованием конечного числа тестов (1322 регрессионных и 48 новых).
4. Время работы программных средств поддержки регрессионного тестирования составило не более 2.5 минут, что оказалось приемлемым для тестируемого программного проекта и не отразилось на сроках его поставки.
5. Приемлемые требования программных средств по объёму памяти (не более 2.7% от объёма тестируемого проекта) не препятствовали их применению для регрессионного тестирования реальных программных систем.

В отношении самих предложенных методов и средств регрессионного тестирования были получены следующие основные результаты:

1. Получено подтверждение, что метод покрытия точек использования неисполняемых определений является работоспособным.
2. Подтверждена работоспособность и сходимость метода порождения новых тестов на основании анализа «подозрительных» состояний.
3. Подтверждена работоспособность методики расширенного регрессионного тестирования программного проекта. Важными свойствами предложенной методики являются учёт функции предсказания целесообразности и методов упорядочения тестов, а также применение новых методов отбора и порождения тестов.

4. Принципы построения системы поддержки регрессионного тестирования и способы программной реализации предложенных методов прошли проверку на практике. Результаты проверки показывают пригодность системы к использованию для автоматизации регрессионного тестирования больших промышленных программных проектов.

Результаты, полученные как по отношению к протестированным программным проектам, так и по отношению к методам и средствам поддержки регрессионного тестирования, подтверждаются набранной в ходе практической работы статистикой.

Разработанные методы и программные средства внедрены в производственный процесс ЗАО «Моторола ЗАО», ЗАО «Северо-Западная Лаборатория» и научно-производственной фирмы «Юпитер».

3. ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

Для достижения цели в работе были решены следующие задачи:

1. Проведён анализ существующих методов и средств регрессионного тестирования программного обеспечения и выбор требований к системе.
2. Разработан метод выборочного регрессионного тестирования. Этот метод является безопасным и целесообразным, то есть отбирает 100% тестов, обнаруживающих ошибки, и оправдывает расходы на собственное применение. Он применим для регрессионного тестирования больших промышленных программ объёмом до 10^7 строк кода, поскольку является эффективным и обеспечивает линейное возрастание времени выполнения в зависимости от объёма тестируемой программы. Метод применим для широкого круга программ, написанных на языках C, C++ и C#, то есть является универсальным.
3. Разработан метод порождения новых тестов на основании анализа «подозрительных» состояний. Метод находит ситуации, «подозрительные» с точки зрения возможного наличия ошибок, и выбирает методы их дальнейшего тестирования. Он применим как для случая модульного тестирования с применением структурных критериев покрытия кода, так и при системном тестировании, ориентированном на требования технического задания.
4. Разработана методика расширенного регрессионного тестирования. Эта методика учитывает как разработанные методы отбора и порождения тестов, так и не учтённые в стандартном алгоритме Ротермела – Харролд алгоритм упорядочения и функцию предсказания целесообразности.
5. Создан инструментарий, поддерживающий процесс автоматизации регрессионного тестирования. Объём исходного кода на языках C и C++ составил около 960 килобайт.

6. Предложенные методики и разработанный инструментальный пакет были применены при регрессионном тестировании промышленных программных проектов в ЗАО «Моторола ЗАО», ЗАО «Северо-Западная Лаборатория» и научно-производственной фирме «Юпитер».

Практические результаты, полученные в ходе работы, позволяют сделать вывод, что поставленные задачи решены, и констатировать достижение цели всей работы. По своей значимости работа относится к новым технологическим средствам поддержки регрессионного тестирования программных продуктов, выполняемым на основе исследований современного инструментария и процесса производства программного продукта. Приложения работы важны для практики и инвестируются конкретными заказчиками.

4. СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Котляров В.П., Епифанов Н. А., Некрасов А. О., Коликова Т. В. Основы современного тестирования программного обеспечения, разработанного на C#. СПб.: Нестор, 2003. 86 с.
2. Швецов В. В., Епифанов Н. А., Котляров В. П. Расширение свойств профайлера для использования в системе оптимизации регрессионного тестирования по частичным тестовым наборам // XXX неделя науки СПбГТУ. Материалы межвузовской научной конференции (ФТК и ИИСТ) / СПбГТУ. 2002. С. 19-20.
3. Швецов В. В., Епифанов Н. А., Котляров В. П. Система оптимизации регрессионного тестирования по частичным тестовым наборам // XXX неделя науки СПбГТУ. Материалы межвузовской научной конференции (ФТК и ИИСТ) / СПбГТУ. 2002. С. 18-19.
4. Nekrasov A. O., Epifanov N. A., Kotlyarov V. P. Code profile for parallel execution systems testing // International Workshop on Program Understanding. 2003. P. 102-106.
5. Epifanov N. A., Nekrasov A. O., Kotlyarov V. P. Use of extended test suites for regression testing // International Workshop on Program Understanding. 2003. P. 98-101.
6. Kotlyarov V. P., Epifanov N. A. Regression testing optimization // Motorola Technology Day. 2001. 8 p.