На правах рукописи

Кичигин Дмитрий Юрьевич

МЕТОД РЕДУКЦИИ ТЕСТОВОГО НАБОРА ДЛЯ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ

Специальность 05.13.11 –
математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва - 2010

Работа выполнена в Институте системного программирования РАН.

Научный руководитель: доктор физико-математических наук
Петренко Александр Константинович

Официальные оппоненты: доктор технических наук, профессор,
Позин Борис Аронович
кандидат физико-математических наук
Иванов Денис Владимирович

Ведущая организация: Научно-исследовательский институт системных исследований РАН

Защита диссертации состоится «17» июня 2010 г. в 15 часов на заседании диссертационного совета Д.002.087.01 при Институте системного программирования РАН по адресу:

109004, Москва, А.Солженицына 25, Институт системного программирования РАН, конференц-зал.

С диссертацией можно ознакомиться в библиотеке Института системного программирования РАН.

Автореферат разослан «15» мая 2010 г.

Ученый секретарь
диссертационного совета
кандидат физико-математических наук  Прохоров С.П.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. Для современного программного обеспечения, характеризующегося большим объемом и богатой функциональностью, задача обеспечения качества, в частности, корректности и надежности, становится все сложнее. Современные программные комплексы базируются на принципе модульности: создаваемое ПО разбивается на модули, которые разрабатываются либо силами одного коллектива разработчиков, либо разрабатываются и поставляются независимыми производителями, а затем интегрируются вместе. Модульный подход не только позволяет облегчить и оптимизировать процесс разработки ПО, но также систематизировать и повысить эффективность тестирования разрабатываемого ПО, так как тестирование можно проводить на уровне модулей, подсистем и системы в целом. При этом качество и надежность программного обеспечения в значительной степени определяются корректностью взаимодействия различных модулей программного обеспечения между собой.

Одним из основных процессов, направленным на обеспечение корректности взаимодействия модулей при создании и последующих модификациях программного обеспечения, является проведение интеграционного и, особенно, регрессионного интеграционного тестирования. Интеграционным тестированием называется тестирование взаимодействия между интегрируемыми компонентами программного и/или аппаратного обеспечения. Регрессионным тестированием называется повторное тестирование системы или компонента с целью убедиться, что внесенные изменения не привели к неожидаемым эффектам, и что система или компонент продолжает удовлетворять своей спецификации. Регрессионное интеграционное тестирование должно осуществляться регулярно по мере внесения модификаций в ПО. Для проведения регрессионного интеграционного тестирования должны использоваться

специализированные интеграционные тесты, целью которых является проверка взаимодействия интегрируемых модулей. Однако на практике создание таких тестов достаточно сложно и трудоемко, поэтому распространенным подходом является использование существующих регрессионных тестов, изначально созданных для модульного и системного тестирования. Такой подход позволяет сэкономить ресурсы на создании специализированных регрессионных интеграционных тестов, однако является неэффективным: так как используемые в этом случае регрессионные тесты изначально не создавались специально для интеграционного тестирования, на практике только часть таких запускаемых тестов тестирует взаимодействие между интегрируемыми модулями, а оставшаяся часть тестов работает «вхолостую», лишь потребляя ресурсы. Ситуация значительно усугубляется в случае регрессионного интеграционного тестирования крупных программных комплексов: как правило регрессионные тесты для такого программного обеспечения достаточно сложны и требуют больших ресурсов, что значительно увеличивает потери ресурсов на запуске тестов, не тестирующих требуемое взаимодействие между интегрируемыми модулями.

Для решения этой проблемы используется подход, известный как редукция тестового набора, и заключающийся в отборе набора тестов меньшего размера, но при этом, желательно, с неменьшей способностью обнаруживать ошибки, чем у исходного набора тестов. Традиционные способы редукции набора тестов заключаются в «отсеивании» тестов из первоначального набора таким образом, чтобы сохранялся уровень его адекватности в терминах некоторого выбранного критерия адекватности. Для оценки адекватности набора тестов, большинство существующих методов используют метрики адекватности, основывающиеся на статическом анализе и/или инструментировании исходного текста ПО, что обусловливает их ограничения:

1. Неприемлимость использования для программного обеспечения большого объема. Инструментирование исходного кода может быть достаточно нетрудоемким для программного обеспечения, состоящего из нескольких модулей, однако инструментирование исходного кода для ПО, состоящего из нескольких тысяч модулей, гораздо более сложная и затратная процедура.
2. Невозможность применения для тестирования взаимодействий с участием повторно используемых бинарных программных компонент. Существенным трендом в разработке современного программного обеспечения является использование повторно используемых программных компонент. Такие компоненты часто разрабатываются независимыми разработчиками и поставляются, как правило, без исходного кода в бинарном виде. Это делает невозможным применение методов, основанных на анализе или инструментировании исходного кода программы, что, вследствие постоянного увеличения количества и разнообразия функциональности доступных повторно используемых бинарных программных компонент, является очень серьезным ограничением.

Вышеприведенные ограничения существенно сужают область применения существующих методов редукции набора тестов, и делают актуальной разработку новых методов, не требующих статический анализ или инструментирование исходного кода модулей и способных работать с бинарными программными компонентами, поставляющимися без исходного кода.

Целью диссертационной работы являлась разработка метода редукции тестового набора для регрессионного интеграционного тестирования, не требующего инструментирование исходного кода модулей и способного работать с бинарными программными компонентами, поставляющимися без исходного кода. Для достижения этой цели необходимо было решить следующие задачи:

1. Исследовать виды взаимодействия между модулями ПО и типичные для этих видов интеграционные ошибки;
2. Разработать модель процесса взаимодействия интегрируемых модулей, адекватную задаче анализа взаимодействия модулей при запуске различных тестов;
3. Разработать метод редукции регрессионного тестового набора, основанный на сравнении моделей процесса взаимодействия, построенных на информации, полученной при прогоне тестового набора;
4. Разработать экспериментальную систему редукции набора тестов; испытать метод и оценить его эффективность на наборах тестов, использующихся для тестирования индустриального программного обеспечения.

Научная новизна предложенного в работе метода заключается в том, что впервые для редукции набора тестов используется модель взаимодействия на основе техники «скользящего окна» по трассе вызовов интерфейсных функций, выполненных во время работы программы на тесте. Метод обеспечивает существенное сокращение размеров тестового набора без потери качества тестирования, что обосновано приведенным исследованием.

Практическая значимость работы. Разработанный в работе метод редукции тестового набора значительно расширяет область применения методов редукции наборов тестов, делая их применимыми при регрессионном тестировании взаимодействий с участием готовых программных компонент, поставляемых без исходного кода. Практическая ценность предлагаемого метода доказана успешным применением метода для редукции набора тестов для регрессионного тестирования взаимодействий осуществляемых через интерфейсы функций со скалярными параметрами и указателями. Проведенный на примере вычислительной системы на базе распространенной операционной системы SUSE Linux

Enterprise Desktop v10.2 анализ структуры интерфейсов системного и прикладного программного обеспечения, показал, что доля таких интерфейсов составляет порядка 96% от общего числа программных интерфейсов, что подтверждает широкую область применения предлагаемого метода. Кроме того, проведенные исследования подтвердили, что данный метод также может быть успешно адаптирован к интерфейсам функций, содержащих не только скалярные параметры и указатели, но и любые другие параметры.

В качестве основных **методов исследования** в данной работе использовались математическое моделирование, математическая логика, элементы теории множеств и общей алгебры. В качестве теоретической основы для моделирования процесса межмодульных взаимодействий и построения отношения эквивалентности между процессами использовался аппарат теории множеств и методы построения отношений эквивалентности между элементами разнообразных типов данных. Для оценки количественных характеристик предложенного метода редукции и сравнения предложенного метода с другими методами редукции использовался метод статистического эксперимента в комбинации с методом подсева искусственных ошибок.

Апробация работы и публикации. Результаты работы докладывались:

- Seventh International Andrei Ershov Memorial Conference «Perspectives of System Informatics» (PSI'09), Новосибирск, 2009.
- на секции Software Testing международной конференции SYRCoSE 2007 (The First Spring Young Researchers Colloquium on Software Engineering), Москва, 2007.
- на научном семинаре Института Системного Программирования РАН, Москва, 2006, 2007, 2009.

– на научном семинаре The Knowledge Management Research Group, Loughborough University, Loughborough, United Kingdom, 2005.

Результаты диссертации опубликованы в 4 печатных работах.

Структура и объем работы. Диссертация состоит из введения, 3 глав, заключения, списка литературы и 1 приложения. Список литературы включает 86 названий. Общий объем диссертации составляет 141 страницу. Объем приложения составляет 7 страниц.

СОДЕРЖАНИЕ РАБОТЫ

Введение содержит общую характеристику проблемы, обоснование необходимости проведения исследований по теме диссертации и ее актуальности. В нем сформулированы цель и задачи диссертационной работы, рассмотрен круг задач, решение которых положено в основу диссертационной работы.

В **первой главе** рассматриваются особенности редукции набора тестов для регрессионного интеграционного тестирования и также рассматриваются существующие методы редукции набора тестов.

В первой части главы рассматривается понятие регрессионного тестирования, заключающегося в повторном тестировании пограммного обеспечения с целью убедиться что внесенные изменения не привели к неожидаемым эффектам, и необходимость проведения регрессионного тестирования в процессе сопровождения программного обеспечения. Отмечается, что важной особенностью регрессионного тестирования является наличие существующего набора тестов. Формулируются принципы выборочного регрессионного тестирования и показывается перспективность этого подхода для повышения эффективности регрессионного тестирования, достигаемого вследствие уменьшения затрат ресурсов, требуемых для достижения целей регрессионного тестирования. Также приводится классификация методов редукции набора тестов для проведения выборочного тестирования и рассматриваются принципы оценки таких методов. После этого подробно рассматриваются особенности выборочного тестирования при проведении регрессионного интеграционного тестирования с использованием бинарных программных компонент, в частности, делается вывод о том, что в реальных крупных системах встречается дополнительное ограничения на доступ к исходному коду таких компонент, что, в свою очередь, предъявляет дополнительные требования к методам редукции набора тестов.

Во второй части главы приводится детальный обзор существующих методов редукции набора тестов для регрессионного интеграционного тестирования. Отмечается, что в литературе встречается мало методов редукции для интеграционного тестирования. Большинство методов редукции набора тестов для интеграционного тестирования является адаптацией существующих методов для модульного тестирования. Показывается, что используемые методами метрики покрытия основываются на анализе исходного кода, вследствие чего для работы этих методов редукции требуется наличие исходного кода тестируемого программного обеспечения.

На основании приведенных в главе данных и проведенного анализа существующих методов, формулируется вывод о том, что традиционные методы редукции набора тестов оказываются неприменимыми в условиях регрессионного интеграционного тестирования с использованием программных компонент, поставляющихся только в бинарном виде, и, следовательно, оказываются не вполне адекватны рассматриваемой задаче редукции набора тестов. Делается вывод о необходимости создания нового метода редукции набора тестов, способного работать в условиях ограниченного доступа к исходному коду программного обеспечения, который и определил цели настоящей работы.

Вторая глава посвящена описанию разработанного метода редукции набора тестов для интеграционного тестирования.

В начале главы описывается класс рассматриваемых в данной работе взаимодействий и приводится классификация интеграционных ошибок.

После этого приводится описание разработанного метода редукции. В основе предлагаемого метода редукции лежит эвристический алгоритм минимизации исходного набора тестов. Данный алгоритм базируется на предположении, что, если покрытие точек взаимодействия между интегрируемыми модулями на двух тестах одинаково, то возможности этих тестов по выявлению интеграционных ошибок между этими

интегрируемыми модулями также одинаковы. В частности, если тесты вообще не задействуют точки взаимодействия между модулями, то такие тесты не представляют пользы при тестировании взаимодействия. Метод основан на этих предположениях и состоит в «отсеивании» тех тестов, которые создают эквивалентные покрытия точек взаимодействия модулей. В качестве точек взаимодействия принимаются интерфейсные функции интегрируемых модулей. Для оценки покрытия точек взаимодействия используются модели процесса взаимодействия модулей, в основе которых лежат последовательности вызовов интерфейсных функций интегрируемых модулей, произведенных во время работы программного обеспечения на наборе тестов.

Описание метода состоит из трех частей: вначале описывается модель анализируемого процесса взаимодействия; затем описывается способ сравнения моделей; в завершение описывается алгоритм редукции, использующий модели взаимодействия и предложенный способ сравнения моделей для принятия решения какой тест следует «отсеять» из исходного тестового набора, а какой – нет.

Описание модели процесса взаимодействия начинается с определения основных используемых терминов: таких как понятие *интерфейсной функции*, *трассы взаимодействия* двух модулей на тестовом варианте, и *последовательности* интерфейсных функций длины *K*. *Интерфейсными функциями* называются функции, входящие в состав программного интерфейса модуля программного обеспечения. *Трассой взаимодействия* модулей *A* и *B* на тесте *t* называется последовательность записей, каждая из которых соответствует вызову интерфейсной функции модуля *B* модулем *A* при выполнении тестируемого программного обеспечения на тесте *t*. *Последовательностью* интерфейсных функций длины *K* называется любая непрерывная последовательность вызовов интерфейсных функций длины *K*, встречающаяся в трассе взаимодействия. После этого определяется *модель процесса взаимодействия* модулей на тесте *t* (далее по тексту *модель*

взаимодействия) как *множество всех возможных последовательностей* длины *K*, встречающихся в трассе взаимодействия между модулями *A* и *B* на тесте *t*.

Для построения модели взаимодействия используется техника «скользящего окна» с размером *K* (размер окна соответствует длине выделяемых последовательностей). Согласно этому подходу, выделение последовательностей происходит следующим образом: в качестве первой последовательности из трассы выбираются *K* идущих подряд интерфейсных функций, начиная с первой функции в трассе; в качестве второй последовательности выбираются *K* идущих подряд интерфейсных функций, начиная со второй функции; и так далее, пока не будет пройдена вся трасса. В качестве примера построения модели взаимодействия модулей, рассмотрим следующую трассу интерфейсных функций:

fopen fread fread fread fread fwrite fclose

тогда результатом выделения последовательностей с помощью скользящего окна размера 4 будет следующее множество последовательностей, являющееся моделью взаимодействия (Рис.1):

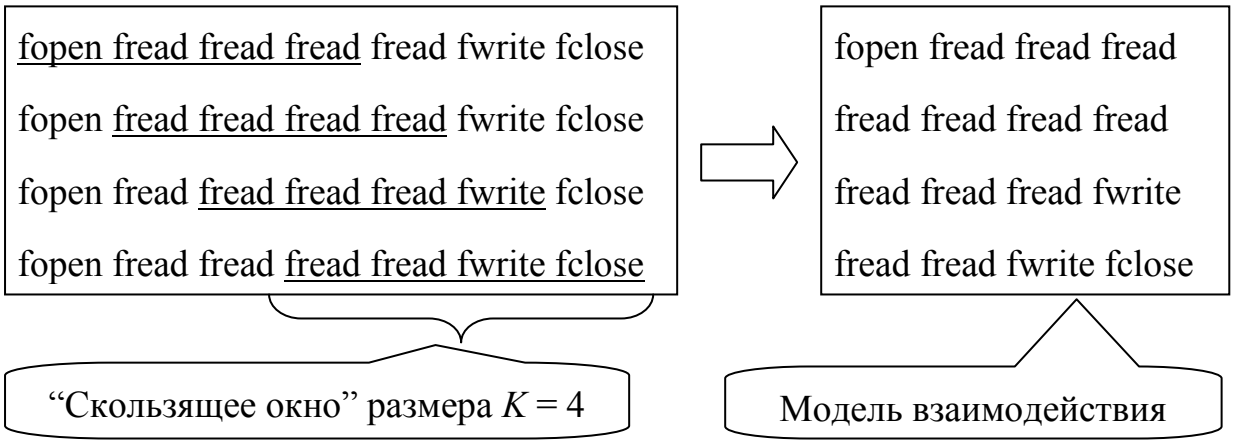

Рис.1. Построение модели взаимодействия модулей.

В качестве способа сравнения моделей взаимодействия было предложено отношение эквивалентности между моделями, учитывающее значения параметров функций, переданных во время выполнения тестируемой программы на тесте, и определяемое с помощью формулы:

$$\delta(M_1, M_2) = \prod_{s1 \in M_1} [\sum_{s2 \in M_2} \prod_{k=1}^{K} \delta(f_{k1}, f_{k2})] \times \prod_{s2 \in M_2} [\sum_{s1 \in M_1} \prod_{k=1}^{K} \delta(f_{k2}, f_{k1})] \qquad (*)$$

где M_1, M_2 – модели взаимодействия; s_1, s_2 - последовательности интерфейсных функций, входящие в состав моделей; K – длина последовательности интерфейсных функций; f_{ki} – вызов интерфейсной функции, имеющий порядковый номер k в последовательности s_i; $\delta(f_1, f_2)$ – функция, задающее отношение эквивалентности между вызовами функций f_1 и f_2, и определяемое с помощью формулы:

$$\delta(f_1, f_2) = \begin{cases} \prod_{i=1}^{n} \delta(x_i, y_i), \text{ если } \delta(\text{name}(f_1), \text{name}(f_2)) = 1 \\ 0, \text{ если } \delta(\text{name}(f_1), \text{name}(f_2)) = 0 \end{cases}$$

где f_1, f_2 – вызовы функций; x_i, y_i, $i = 1..n$ – значения параметров функций f_1 и f_2 соответственно; name(f) – имя функции f; $\delta(\text{name}(f_1), \text{name}(f_2))$ - функция, задающая отношение эквивалентности для имен функций, и равная 1 если имена функций совпадают и 0 в противном случае. В качестве отношений эквивалентности для значений параметров функций используются следующие отношения:

- Скалярные номинальные параметры: $\delta(x,y)$, где δ – отношение равенства;
- Скалярные числовые параметры: $\delta(x,y) = \delta(\text{interval}(x), \text{interval}(y))$, где $\delta(\text{interval}(x), \text{interval}(y))$ – отношение равенства (совпадения) между интервалами, которым принадлежат значения параметров;
- Указатели: $\delta(p_1, p_2) = 1$, если p_1 = NULL и p_2 = NULL, либо если p_1 != NULL и p_2 != NULL; в любых других случаях $\delta(p_1, p_2) = 0$;
- Другие типы: $\delta(x,y) \equiv 1$.

Формулируется и доказывается теорема о том, что отношение, заданное формулой (*), является отношением эквивалентности, а именно, удовлетворяет свойствам рефлексивности, симметричности и транзитивности, в дальнейшем используемым в алгоритме редукции.

Важным достоинством предложенного отношения эквивалентности между моделями взаимодействий является то, что оно позволяет задавать любые способы учета значений параметров интерфейсных функций, которые могут быть представлены в виде отношения эквивалентности. В частности, предложенное отношение эквивалентности можно использовать и для того, чтобы различать другие (т.е. не скалярные и не-указатели) типы параметров функций, что означает, что предлагаемый метод также может быть адаптирован, чтобы учитывать значения других типов параметров интерфейсных функций. Для иллюстрации этого факта приводится пример адаптации предложенного метода редукции для учета параметров функций, принадлежащих строковому типу данных.

Используемый в методе алгоритм редукции заключается в последовательном переборе тестов из исходного набора тестов, для каждого из которых принимается решение поместить данный тест в сокращенный набор тестов или нет, и состоит из следующих шагов. На первом шаге выбирается очередной тест из исходного набора тестов и на нем запускается тестируемое программное обеспечение. Во время работы программного обеспечения на выбранном тесте происходит сбор трассы взаимодействия интегрируемых модулей и построение модели взаимодействия интегрируемых модулей. Если построенная модель взаимодействия оказывается пустой, то это означает, что на данном тесте интегрируемые модули не взаимодействовали и, в таком случае, данный тест пропускается (не помещается в сокращенный набор тестов) и алгоритм возвращается к первому шагу для выбора следующего теста из исходного набора тестов. В противном случае, если построенная модель взаимодействия на тесте не пуста, то происходит сравнение модели с моделями взаимодействия,

построенными на предыдущих итерациях алгоритма. Для сравнения моделей используется отношение эквивалентности между моделями. В случае если обнаруживается модель взаимодействия, построенная на одном из предыдущих тестов и эквивалентная текущей модели, то тест также пропускается. Если же, наоборот, среди моделей, построенных на предыдущих итерациях алгоритма, не находится модели эквивалентной текущей модели, то такой тест заносится в сокращенный набор тестов. После этого алгоритм возвращается к первому шагу для выбора очередного теста из исходного набора тестов. Когда исходный набор тестов оказывается пройденным, алгоритм редукции завершает свою работу и сокращенный набор тестов считается построенным.

Описанный выше алгоритм редукции обладает вычислительной сложностью, максимальная оценка которой составляет $O(T^2 \times N^{2K} \times K)$, при этом максимальный объем памяти, используемой для хранения модели в памяти компьютера, составляет $O(T \times N^K \times K)$, где: T - количество тестов; K – длина последовательности интерфейсных функций; N - количество интерфейсных функций. Для улучшения этих показателей была проведена оптимизация метода, заключающаяся в том, что табличное представление модели взаимодействия было заменено древовидным, при котором модель взаимодействия представляется в форме дерева, где каждой последовательности вызовов интерфейсных функций соответствует путь в дереве от корня до одного из листовых узлов. Такое представление позволило уменьшить сложность алгоритма до $O(T^2 \times N^K \times \log(N) \times K)$, а максимальный объем используемой памяти до $O(T \times (N^{K+1} - N)/(N-1))$.

После описания метода редукции приводятся условия применения, методика применения, и область применимости предложенного метода.

В конце главы приводится описание экспериментального исследования предложенного метода. Экспериментальное исследование включало три серии экспериментов, целями которых являлись:

- Оценка основных характеристик метода, таких как коэффициент сокращения набора тестов и коэффициент обнаружения ошибок, и сравнение с методами случайной редукции и редукции пустых трасс;
- Оценка влияния длины последовательности вызовов функций (параметр K) на работу метода и оценка затрат ресурсов при работе алгоритма редукции;
- Исследование адаптируемости метода для учета параметров функций других типов.

Первая серия экспериментов была направлена на исследование основных характеристик метода и сравнение с методом случайной редукции и с методом редукции пустых трасс, базирующемся на известном методе регрессионного интеграционного тестирования, отсеивающем только те тесты, которые не затрагивают взаимодействие модулей. При анализе методов редукции набора тестов важны следующие характеристики:

1. Коэффициент сокращения набора. Это одна из наиболее существенных характеристик работы методов редукции, так как показывает, насколько сокращается первоначальный набор тестов и, соответственно, связанные с ним затраты. Чем больше коэффициент сокращения набора тестов, тем сильнее уменьшаются затраты на его выполнение и проверку результатов, тем самым уменьшая затраты на проведение регрессионного тестирования в целом.
2. Коэффициент обнаружения ошибок. Основной опасностью необоснованной редукции набора тестов является возможное уменьшение способности сокращенного набора тестов обнаруживать ошибки вследствие удаления тестов, выявляющих ошибки. Слишком большие потери в способности обнаружения ошибок могут свести на нет целесообразность редукции набора тестов. Поэтому важно, чтобы в результате работы метода способность обнаружения ошибок сокращенным набором по возможности не уменьшалась, а если и уменьшалась, то незначительно. Коэффициент обнаружения ошибок

отражает в процентном выражении способность сокращенного набора тестов обнаруживать ошибки по сравнению с исходным набором тестов.

Для оценки вышеперечисленных характеристик предложенного метода редукции и для его сравнения с методами случайной редукции и редукции пустых трасс была проведена серия экспериментов на ряде тестовых наборов, сгенерированных из тестов тестового набора GNU Assembler. В Таблице 1 приведены основные результаты эксперимента.

Анализ результатов проведенных экспериментов продемонстрировал высокую эффективность метода: коэффициент сокращения исходных тестовых наборов составлял в среднем 52%, при этом коэффициент обнаружения ошибок сокращенных наборов не уменьшался и составлял 100% от уровня исходных наборов. Предложенный метод по совокупности характеристик превзошел методы случайной редукции и редукции пустых трасс: метод был на 20% эффективнее метода случайной редукции в части обнаружения ошибок, и на 42% эффективнее метода редукции пустых трасс в части сокращения исходного набора тестов.

Индикатор	**Исходный набор**	**Предло-женный метод**	**Метод редукции пустых трасс**	**Метод случайной редукции**
Размер набора (в среднем)	60	**26**	54	26
Коэффициент сокращения (в среднем, в %)	-	**на 52%**	на 10%	на 52%
Количество обнаруженных ошибок (в среднем)	16	**16**	16	12
Коэффициент обнаружения ошибок (в среднем, в % от кол-ва ошибок, обнаруженных исходным набором)	100%	**100%**	100%	80%

Таблица 1. Результаты эксперимента.

Вторая серия экспериментов была направлена на исследование адаптируемости метода для учета параметров функций типов данных, не учитываемых оригинальным методом (т.е. не скалярных параметров и не указателей). Для проведения эксперимента в качестве примера выбран строковый тип параметров и была построена модификация метода, где в дополнение к скалярным параметрам и указателям также учитывались значения строковых параметров. Для этого было предложено отношение эквивалентности между строками, которое было определено как отношение равенства между их длинами:

$$\delta(x, y) = \delta(length(x), length(y))$$

где: x, y – строковые параметры; $length(x)$ – длина строки x.

Для сравнения характеристик оригинального и модифицированного методов была проведена серия экспериментов на ряде тестовых наборов, также сгенерированных из тестов тестового набора GNU Assembler, где в качестве тестируемого взаимодействия были выбраны модули, большинство интерфейсных функций которых содержало строковые параметры. В Таблице 2 приведены результаты сравнения характеристик оригинального и модифицированного метода:

Индикатор	**До модификации**	**После модификации**
Коэффициент сокращения (в среднем, в %)	на 88%	на 79%
Коэффициент обнаружения ошибок (в среднем, в % от количества обнаруженных исходным набором ошибок)	91%	100%

Таблица 2. Результаты эксперимента.

Результаты данного эксперимента подтвердили хорошую адаптируемость для учета параметров функций других типов: коэффициент обнаружения ошибок наборов тестов, построенных с помощью модифицированного метода, значительно повысился по сравнению с

оригинальным методом и составил 100%, т.е. сокращенные тестовые наборы обнаруживали столько же ошибок, сколько и исходные тестовые наборы. Коэффициент сокращения наборов тестов, построенных с помощью модифицированного метода, уменьшился по сравнению с оригинальным методом, что является ожидаемым результатом, так как, вследствие учета значений дополнительных типов параметров, модифицированный метод способен лучше различать процессы взаимодействия, и, поэтому, размеры построенных им наборов оказываются не меньше, чем размеры наборов, построенных оригинальным методом.

Третья серия экспериментов была направлена на оценку влияния длины последовательности интерфейсных функций *K* на работу метода и на оценку затрат ресурсов при работе метода.

Длина последовательностей интерфейсных функций *K* является важным параметром предложенного метода редукции. Этот параметр влияет на работу метода следующим образом: чем больше длина последовательностей, тем больше процессов взаимодействий будут считаться различными. В результате коэффициент сокращения набора тестов уменьшится, но зато также уменьшатся (по сравнению с методом, использующим меньшую длину последовательностей) потери в способности обнаружения ошибок. Меньшая длина последовательностей, соответственно, приведет к увеличению коэффициента сокращения и также к увеличению потерь в обнаружении ошибок. Длина последовательностей также влияет на затраты ресурсов при работе метода: при большей длине последовательностей требуется больший объем памяти для хранения моделей взаимодействия и большие временные затраты на работу метода.

Для оценки влияния длины последовательностей интерфейсных функций *K* проведено сравнение характеристик различных модификаций метода, со значениями *K*, равными 1,2,3,6,10, и 15. Результаты сравнения приводятся в Таблице 3:

Индикатор \ Длина *K*	**1**	**2**	**3**	**6**	**10**	**15**
Коэффициент сокращения (в среднем, в %)	74%	61%	57%	54%	52%	45%
Коэффициент обнаружения ошибок (в среднем, в % от кол-ва ошибок, обнаруженных исходным набором)	92%	99%	100%	100%	100%	100%
Время работы (в среднем)	2 сек	5 сек	9 сек	27 сек	1.2 мин	6.5 мин
Объем используемой памяти (Мб, в среднем)	9.6	9.7	9.8	10.6	12.2	15.3

Таблица 3. Результаты эксперимента.

Результаты эксперимента подтвердили, что бОльшая длина последовательностей функций *K* улучшает способность метода обнаруживать ошибки. Но, при этом, уменьшает коэффициент сокращения набора тестов и увеличивает затраты ресурсов на проведение тестирования. Аналогично, меньшая длина последовательностей ухудшает способность метода обнаруживать ошибки, и, одновременно, увеличивает коэффициент сокращения набора тестов и уменьшает затраты ресурсов на проведение тестирования. Результаты показали, что при значениях *K* больше 6, таких как 10 и 15, использование метода становилось достаточно затратным: значительно возрастают временные затраты на проведение редукции, что, однако, не приводит к увеличению уровня обнаружения ошибок, так как коэффициент обнаружения ошибок уже достиг 100%. Также было замечено, что для тестируемых программ коэффициент обнаружения ошибок достигает 100% уже при $K = 3$, однако бОльшие значения *K*, такие как 6, добавляют «запас прочности» методу при незначительном увеличении затрат ресурсов.

В **третьей главе** приведены результаты апробации метода редукции набора тестов и программная реализация метода редукции.

Апробация разработанного метода редукции проводилась на задаче сокращения индустриального тестового набора LSB Application Battery. Тестовый набор LSB Application Battery используется в процессе

сертификации Линукс-подобных операционных систем на соответствие спецификации Linux Standard Base (LSB) и предполагает достаточно высокую степень интерактивного взаимодействия с пользователем. Для проведения апробации использовалась версия 3.1 набора LSB Application Battery, которая представляет собой коллекцию из 15 общеиспользуемых приложений. Проведение тестирования с помощью этой версии набора занимает около двух часов на машине с конфигурацией Intel Pentium 4-M CPU 2.00 GHz / 448 MB RAM.

При проведении апробации, в дополнение к основным характеристикам методов редукции, таким как коэффициент редукции набора тестов и коэффициент обнаружения ошибок, также был оценен выигрыш временных затрат на проведение тестирования, который дает использование сокращенного набора тестов при выполнении на нем тестируемого программного обеспечения. Результаты апробации предложенного метода на наборе LSB Application Battery представлены в Таблице 4:

Индикатор	**Исходный набор**	**Сокращенный набор**
Размер сокращенного набора (кол-во тестов)	81	9
Коэффициент сокращения	9 раз	
Кол-во обнаруженных ошибок	6	6
Изменение уровня обнаружения ошибок	0%	
Время выполнения	1 час 58 минут	10 минут
Коэфф. сокращения времени выполнения	11.8 раз	
Время работы метода редукции	1 минута	
Выигрыш по времени	1 час 47 минут / 10.7 раз	

Таблица 4. Результаты апробации метода.

Результаты апробации подтвердили высокую практическую ценность метода: в результате работы метода было достигнуто сокращение исходного тестового набора в 9 раз, что привело к сокращению времени тестирования с помощью сокращенного набора тестов в 11.8 раз (до 10 минут). Затраты на работу метода редукции составили 1 минуту, таким образом суммарная экономия времени от использования метода редукции составила 1 час 47 минут или 10.7 раз от времени тестирования с помощью исходного набора тестов. При этом способность тестового набора обнаруживать ошибки не уменьшилась: сокращенный тестовый набор обнаруживал столько же ошибок, сколько и исходный тестовый набор.

На основе данных результатов можно сделать вывод о том, что предлагаемый метод редукции набора тестов показал успешные результаты, значительно сократив исходный тестовый набор без потерь в уровне обнаружения ошибок. Причем такое сокращение обеспечило значительный выигрыш во временных затратах на проведение регрессионного тестирования.

Далее описывается программная реализация метода редукции и отмечается, что:

1. Архитектура программной реализации позволяет достаточно легко адаптировать реализацию метода для работы с программным обеспечением, выполняемым в различных операционных системах/средах выполнения, таких как Linux/UNIX, MS Windows, .NET, Java и др.;
2. Выбранная архитектура позволяет с небольшими затратами адаптировать экспериментальную систему для учета других, более сложных типов параметров функций (т.е. не скалярных параметров и не указателей) и для изменения способа учета типов параметров функций, уже учитываемых в разработанном методе;
3. Реализация модуля анализа взаимодействий позволяет собирать трассы взаимодействий интегрируемых модулей за один проход, во время

первоначального тестирования программного обеспечения на исходном наборе тестов. При регрессионном интеграционном тестировании достаточно часто встречается ситуация, когда меняется не один, а сразу несколько модулей, что требует проверки нескольких разных взаимодействий между модулями. В этом случае, возможность единовременного сбора трасс взаимодействий на полном наборе тестов позволяет значительно сократить издержки при построении регрессионных наборов для последующих регрессионных тестирований разных взаимодействий;

4. Программная компонента, реализующая предложенный метод, поддерживает программный интерфейс для интеграции в различные пользовательские приложения.

В **заключении** представлен обзор результатов, полученных в рамках настоящей диссертационной работы.

Основные результаты работы

В работе были получены следующие основные результаты, которые выносятся на защиту:

1. Разработана новая модель процесса взаимодействия модулей на основе метода «скользящего окна» по трассе вызовов интерфейсных функций;
2. Предложено и обосновано отношение эквивалентности между процессами взаимодействия модулей на основе предложенной модели и дано формальное доказательство его корректности;
3. Разработан новый метод редукции регрессионного тестового набора, основанный на сравнении моделей процесса взаимодействия, построенных по трассе вызовов интерфейсных функций;
4. Разработана экспериментальная система, реализующая метод редукции тестового набора.

Проведено экспериментальное исследование предложенного метода на наборе тестов, использующихся для тестирования индустриального программного обеспечения. Статистическое исследование показало, что метод обеспечивает существенное сокращение размеров тестового набора без потери качества тестирования.

Результаты диссертации опубликованы в работах:

[1] Кичигин Д.Ю. Метод редукции тестового набора для регрессионного интеграционного тестирования // Журнал РАН «Программирование», Номер 5, 2009, С. 57-69.

[2] Dmitry Kichigin. A Method for Test Suite Reduction for Regression Testing of Interactions between Software Modules // In Proceedings of PSI'09, Novosibirsk, 2009. LNCS, Springer: Volume 5947/2010 pp. 177-184, 2010.

[3] D. Kichigin. Test Suite Reduction for Regression Testing of Simple Interactions between Two Software Modules // In Proceedings of Spring Young Researchers Colloquium on Software Engineering (SYRCoSE 2007). Volume 2, ISP RAS. pp. 31-37, 2007.

[4] Кичигин Д.Ю. Об одном методе сокращения набора тестов // Сборник трудов ИСП РАН, том 13, часть 1, М: ИСП РАН, 2007, С.79-92.