На правах рукописи

Тютин Борис Викторович

МЕТОДЫ АВТОМАТИЗАЦИИ РАСПРЕДЕЛЁННОГО ТЕСТИРОВАНИЯ РЕАКТИВНЫХ СИСТЕМ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Санкт-Петербург - 2013

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

Научный руководитель: **Котляров Всеволод Павлович,**
кандидат технических наук, доцент

Официальные оппоненты: **Воробьёв Владимир Иванович,**
д.т.н., профессор, заведующий Лабораторией информационно-вычислительных систем ФГБ УН «СПИИРАН»

Гаврилова Татьяна Альбертовна,
д.т.н., профессор, заведующая кафедрой информационных технологий в менеджменте Высшей школы менеджмента ФГБОУ ВПО «СПбГУ»

Ведущая организация: ОАО «НПО «Импульс»

Защита состоится «15» мая 2014 г. в 14:00 на заседании диссертационного совета Д 212.229.18 при ФГБОУ ВПО «Санкт-Петербургский государственный политехнический университет» по адресу: 195251, Санкт-Петербург, Политехническая ул., д.29, 9 уч. корп., ауд. 325.

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

Автореферат разослан «___» апреля 2014 г.

Ученый секретарь
диссертационного совета
Д 212.229.18, к.т.н., доцент **Васильев Алексей Евгеньевич**

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. В настоящее время в процессе производства программного продукта (ПП) много внимания уделяется обеспечению его качества – соответствию начальным требованиям, эффективности решения поставленной задачи, надёжности работы. В силу особенностей решаемых задач непосредственная выгода от улучшения характеристик создаваемых информационных систем для заказчика часто неочевидна и скрыта. В первую очередь, это связано с тем, что существующие инструменты и методы проверки программных продуктов ориентированы на разработчиков, а не конечных пользователей и заказчиков. Это делает полученную информацию о свойствах программы недоступной для всех заинтересованных лиц, что приводит к снижению эффективности их взаимодействия.

Наиболее популярным инструментом управления качеством программного продукта является тестирование. Актуальная задача уменьшения его стоимости решается с помощью автоматизации путем объединения в технологическом процессе различных готовых инструментов. Существенным недостатком такого подхода является сложность интеграции разнородных средств автоматизации тестирования в существующий процесс разработки и их стыковки с уже использующимся инструментарием. Выполнение данной задачи существенно усложняется тогда, когда требуется обеспечить связь тестирования с исходными требованиями.

Помимо развития подходов к управлению качеством появление новых технологий в сфере создания программного обеспечения (ПО) расширяет круг возможных областей применения средств автоматизации тестирования, а также распределённых тестовых стендов. Одним из наиболее молодых и интенсивно развивающихся направлений являются кластерные и облачные вычисления. Принципы, лежащие в основе предоставления программных средств и комплексных решений как сервиса (SaaS – software as a service, и IaaS – infrastructure as a service) могут быть распространены на автоматизацию тестирования.

Современный уровень развития средств автоматизации тестирования позволяет путём исследований в области параллельных алгоритмов решить проблемы ускорения выполнения тестового набора. Использование формальных нотаций для работы с требованиями позволяет снять ограничения существующих подходов к построению тестового набора, а также представлять информацию о тестировании в более высокоуровневом виде. В связи в этим тематика работы актуальна, исследование востребовано с точки зрения науки и практики.

Степень разработанности темы. Проблемам формализации и автоматизации методов верификации и тестирования, а также из применения на практике посвящены работы таких авторов, как Липаев В.В., Петренко А.К., Карпов Ю.Г, Летичевский А.А.. Среди зарубежных авторов наиболее значимыми с точки зрения близости к теме работы можно назвать труды R. Alura, H. Wenhong, B. Bollig, K. Beck, M. Fewster, R. Osherove и других.

Указанными авторами были опубликованы как фундаментальные работы, сформировавшие теоретическую базу для дальнейших изысканий в области автоматизации тестирования и верификации ПО с применением формальных моделей, так и труды, обобщающие богатый опыт индустриального производства программного продукта. В научных статьях, издаваемых в течение последнего десятилетия, освещаются вопросы автоматического создания тестов и анализа спецификаций.

Однако в данных работах недостаточно рассмотрены вопросы интеграции современных подходов к тестированию, предлагаемые авторами методы автоматизации построения тестового набора обладают известными ограничениями на способ описания тестовых сценариев. При детальной проработке аспектов верификации не до конца раскрывается потенциал её объединения с тестированием.

Цели и задачи диссертационной работы. Целью диссертации является разработка группы методов, автоматизирующих этапы тестирования ПО, начиная с получения тестового набора на основе формальных спецификаций и заканчивая формированием результатов тестирования и их анализом, а также реализация средств поддержки, предоставляющих инструментарий для организации тестирования на различных этапах жизненного цикла ПО — модель, код, исполняемая программа.

Для достижения поставленной цели в работе были решены следующие задачи:

1) Разработка метода создания тестов, основанного на DDT (Data-Driven Testing) и KDT (Keyword-Driven Testing), и имеющего возможность выполнения нелинейных сценариев;
2) Определение способа взаимодействия тестирующей системы с моделью и её компонентами (интеграционное и модульное тестирование) на разных уровнях;
3) Создание метода построения параметризованных тестов с определением набора значений параметров отдельно от тестовых сценариев;
4) Формулировка методов масштабирования выполнения тестового набора;
5) Создание средств автоматизации тестирования, реализующих разработанные методы;
6) Разработка метода автоматизированного тестирования, основанного на созданных программных средствах, и интегрирующего результаты исследования.

Научные результаты и их новизна.

1) Разработан метод создания параметризованного тестового набора на основе формальных спецификаций, сочетающий принципы тестирования, управляемого данными, и тестирования на основе ключевых слов, свободный от ограничений на описание нелинейного поведения и параллельных взаимодействий в тестовых сценариях;

2) Создан метод автоматизации выполнения тестового набора на основе плана тестирования. Его научная новизна заключается в применении символьных поведенческих трасс, которые затем автоматически конкретизуются и выполняются согласно плану тестирования;
3) Предложен метод анализа результатов тестирования и коррекции тестов, автоматизирующий идентификацию точек останова в ходе выполнения теста и отображение поведенческих трасс, ведущих к этой точке, на исходной формальной модели системы;
4) Предложено решение задачи масштабирования выполнения тестового набора с помощью разработанного метода кластеризации группы тестов по критериям её объёма и суммарной сложности тестов и метода параллельного выполнения тестов. Реализация методов автоматически обеспечивает изоляцию среды выполнения тестов, тем самым, исключая их взаимное влияние и снижая сложность автоматизации тестирования;
5) На основе результатов работы создан метод автоматизированного тестирования, интегрирующий анализ требований, метод тестирования, управляемого данными, метод тестирования, основанного на ключевых словах, и автоматизацию масштабирования выполнения тестового набора.

Теоретическая и практическая значимость работы. Созданы методы, автоматизирующие различные этапы основанного на тестировании процесса контроля качества ПО. Теоретической основой методов являются разработанные модель параллельного запуска тестирования и расширение аппарата символьной верификации, разработанного А.А. Летичевским. Предложенные методы формализованы до уровня алгоритмов, на основе которых созданы инструментальные средства поддержки. Последние базируются на технологии верификации VRS и технологии автоматизации тестирования TAT. На основании объединения разработанных методов была создана методика автоматизации тестирования. Использование плана тестирования и конкретизации параметров тестов позволило упростить автоматизацию тестирования и снизить сложность выполнения тестовых процедур. Разработанные методы и средства масштабирования тестирования решили задачу автоматизации настройки и запуска больших тестовых наборов.

Разработанные методы, алгоритмы и программные средства могут быть отчуждены для использования в сторонних методах и информационных системах. В рамках четырёх проектов по тестированию ПО применение результатов исследования позволило получить сокращение времени фазы тестирования в пределах 50-70%.

Исследование проводилось при поддержке в форме гранта Правительства Санкт-Петербурга для студентов, аспирантов вузов и академических институтов, расположенных на

территории Санкт-Петербурга (2013 г.), а также стипендии Президента Российской Федерации (2012 г.).

Предметом исследования являются методы и инструментальные средства автоматизации процесса контроля качества программного продукта (ПП).

Методология и методы исследования. Для решения поставленных в работе задач используются теория графов, теория алгоритмов в области параллельного программирования, теория конечных автоматов, теория инсерционного программирования, аппарат формальных спецификаций. Применяются стандарты языков Use Case Maps (UCM) и Message Sequence Charts (MSC), а также стандарт Message Passing Interface (MPI).

Основные результаты и положения, выносимые на защиту:

— Методы автоматизации этапов тестирования ПО:
  1) автоматическое создание параметризованного тестового набора;
  2) автоматическое тестирование на основе плана;
  3) автоматизированный анализ результатов тестирования;
  4) автоматизированное масштабирование выполнения тестирования;
— Реализация современных подходов к тестированию на базе символьной верификации;
— Адаптация технологии VRS/TAT для использования в облачных и кластерных вычислительных системах;
— Результаты экспериментальных исследований ускорения выполнения тестового набора при масштабировании запуска.

Обоснованность и достоверность полученных результатов обеспечивается корректным использованием теории графов и теории алгоритмов, использованием формальных моделей проектирования параллельных вычислений, сравнением результатов применения в индустриальных проектах разработанных методов и программных средств и аналогичных существующих подходов, соответствие данных практических экспериментов теоретическим оценкам.

Апробация работы. Основные положения и результаты диссертационной работы были представлены на научных конференциях “Технологии Microsoft в теории и практике программирования” (СПб, 2012, 2011, 2010), 6th Spring/Summer Young Researchers’ Colloquium on Software Engineering (Пермь, 2012), XXXIX неделя науки СПбГПУ (СПб, 2010).

Публикации. По теме диссертационной работы было опубликовано 10 печатных работ, 5 из них в изданиях из перечня ВАК.

Внедрение. Разработанные методы и инструменты поддержки внедрены в компаниях ЗАО «Моторола Солюшнз», ООО «Научно-Технический Центр «Северо-Западная Лаборатория» и использованы при разработке учебно-методического комплекса СПбГПУ по

курсам “Технология разработки программного обеспечения” и “Технология параллельных вычислений” на кафедре “Информационные и управляющие системы”. Практическое использование представляемых на защиту результатов подтверждено соответствующими актами о внедрении.

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения, списка литературы и семи приложений. Объём диссертации – 144 страницы машинописного текста, объем приложений – 41 страница, диссертация содержит 43 рисунка, 22 таблицы, список литературы состоит из 106 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность темы исследования; сформулированы цели и задачи работы; описаны основные аспекты её научной и практической значимости, апробация и публикация результатов; приведена структура изложения материала диссертации.

В **первой главе** проведён обзор критериев тестирования, применяемых при формировании тестового набора. Изучены свойства актуальных подходов к тестированию, сформулированы их ограничения и область применения. Исследованы технологии параллельного программирования и возможности их применения в тестировании. На основании обзора инструментов автоматизации тестирования делается вывод о том, что актуальные задачи в области автоматизации тестирования на данный момент решены лишь частично.

Современные подходы к построению тестового набора имеют ряд ограничений и недостатков. Подход к тестированию, основанный на данных (DDT), не подразумевает возможность создания сложной структуры тестов. Реализующее его ПО не обладает возможностью редактирования тестовых сценариев в графическом виде. Подход к тестированию, основанный на ключевых словах (KDT), также имеет указанные недостатки. Общим недостатком обоих подходов является отсутствие поддержки нелинейных конструкций в коде тестовых сценариев. Однако важным преимуществом, которое предоставляют DDT и KDT, является гибкость системы тестирования и возможность её интеграции со сторонними средствами.

На основании анализа в работе сформулированы проблемы интеграции отдельных компонент и подсистем в технологические цепочки цикла тестирования. Определены основные ограничения к применению существующих систем автоматизации тестирования:

- ориентация на те или иные технологии создания ПО;
- отсутствие взаимодействия со средами разработки ПО;
- тесная связь с одной определённой технологией создания ПО.

Технология VRS/TAT выбрана из рассмотренных по критерию полноты функциональности и возможности устранения выявленных ограничений. Определен ряд недостатков данной технологии: отсутствие поддержки распределённого тестирования, недостаточно полная реализация современных подходов к тестированию, невозможность выполнения тестов в кластерной и облачной среде.

На основании обнаруженных пробелов и ограничений в области автоматизации тестирования сформулирована цель работы и задачи, решение которых необходимо для ее достижения.

Во **второй главе** представлены методы, разработанные для устранения недостатков существующих подходов к автоматизации тестирования. Они автоматизируют процесс тестирования, начиная с этапа создания тестового набора и заканчивая получением и анализом результатов выполнения тестовых сценариев.

Основой для написания тестов служат требования. При автоматизации тестирования их необходимо привести к формальному представлению. Для создания формальной спецификации в работе были использованы нотации UCM, Base Protocols (BP) и MSC, а также технология VRS. Она основана на принципах инсерционного программирования, сформулированных в работах А.А. Летичевского, и позволяет автоматически создавать поведенческие трассы по модели системы в нотации BP. Трасса получается, как результат соединения нескольких базовых протоколов в последовательную цепочку и может быть записана в виде следующей формулы:

$$s_0 \xrightarrow{BP_1} s_1 \xrightarrow{BP_2} s_2 \dots s_{n-1} \xrightarrow{BP_N} s_n$$

где BP_1 … BP_n – множество базовых протоколов, S_0 – начальное состояние, S_n – конечное состояние. Подобная трасса представляет одно из возможных поведений системы, и может служить основой для тестовых сценариев. Создание модели системы в нотации BP является достаточно трудоёмким процессом, поэтому форматом спецификации системы, создаваемой вручную, был выбран UCM. Автоматическое преобразование модели системы из UCM в BP осуществляется с помощью инструмента UCM2FM. Алгоритмы и реализация данного инструмента, а также методы формализации, основанные на его использовании, предложены в работах И.В. Никифорова.

В **третьей главе** разработанные методы формализованы до уровня алгоритмов, на основании которых реализованы программные инструменты на базе технологий TAT и VRS.

Разработанный в ходе данного исследования **метод создания параметризованного тестового набора на основе требований к продукту** интегрирует формализацию требований и построение поведенческих трасс, позволяя автоматически получить тестовые сценарии на основании формальной модели требований к ПО.

Данный метод состоит из следующих шагов:

1) Формулирование требований к программному продукту и построение формальной модели системы путём преобразования требований в совокупность элементов UCM диаграмм:

$$\forall REQ \in R, TRACE = \{u_i\}, u_i \in U, U = U_{start} \cup U_{end} \cup U_{int}$$

где R – множество исходных требований, U – модель системы в нотации UCM, TRACE – отображение требования на набор элементов модели u_i. Данный шаг разработанного метода опирается на исследования диссертационной работы И.В. Никифорова;

2) Анализ формальной модели и определение ключевых параметров функционирования системы;
3) Построение модели системы в нотации BP. Этот и предыдущий шаги выполняются на основе технологии UCM2FM, разработанной в рамках исследований Никифорова И.В.;
4) Построение средствами VRS символических поведенческих MSC трасс, покрывающих модель, на основании критериев, задаваемых с помощью гидов;
5) Преобразование полученных поведенческих трасс в тесты.

В рамках данного метода был разработан подход к интерпретации MSC, обеспечивающий поддержку нелинейного поведения и параллельных взаимодействий для реализации концепции отделения данных от логики тестов. Данный подход базируется на исследованиях в области интерпретации и анализа MSC (M. Yannakakis, A. Roychoudhury) и фундаментальных работах Э. Кларка и Б. Боэма. Основными его аспектами являются:

1) Интерпретация сущности в тесте как независимого процесса;
2) Выполнение блоков конструкции parallel независимо с сохранением порядка следования событий. Условие перехода может быть представлено в следующем виде:

$$s \rightarrow \{s_i^1 : E_i^1, \ldots s_k^N : E_k^N\} \rightarrow s', \forall E \in PAR$$

где i,..k – индексы текущих возможных событий для каждого блока, s и s’ – текущее и последующее состояние поведенческой трассы, E_k^j – очередное событие в блоке, s_k^j – текущее состояние при условии выполнения очередного события в блоке;

3) Выполнение событий concurrent region в произвольном порядке:

$$s \rightarrow \{s_1 : E_1, \ldots s_N : E_N\} \rightarrow s', \forall E \in COREGION$$

Скорость разработки тестового набора была повышена путём параметризации тестов за счёт их получения из символьных поведенческих трасс. При таком подходе трассы представляются в виде двоек L = <B, R_B >, где B - базовый протокол, R_B - ограничения на параметры этого протокола. С помощью аппарата предикатных трансформеров, описанного в работах А.А. Летичевского, для трасс может быть получена таблица конкретизации параметров, которая была применена для автоматизации формировании плана тестирования.

Для интеграции данного подхода в разработанный метод был реализован алгоритм преобразования параметризованных поведенческих трасс в тесты. Пусть X - множество инстанций окружения, Y – множество тестируемых компонент системы, S – множество событий в тесте, λ(s) – функция получения имён сущностей, относящихся к событию s. Возможны следующие действия над набором сущностей в тесте:

1) Объединение нескольких сущностей в одну:

$$f(M): M \rightarrow \{SUT\}, Y' = f(Y), S' = S - \{s, (\forall x \in \lambda(s)) x \in Y\}$$

где f(M) – преобразование множества сущностей в один элемент,

2) Замещение тестируемых компонент автоматически создающимися заглушками с поведением, определяемым тестовым сценарием:

$$X = X \cup Y', Y' = \{Y_a, Y_m ...\}, Y_a \rightarrow Stub_a, Y_b \rightarrow Stub_b, ...$$

Данные действия автоматизированы в рамках разработанного метода и позволяют автоматически получить параметризованный тестовый набор, готовый к запуску. В рамках созданного **метода автоматизации выполнения тестового наборы на основе плана тестирования** осуществляется автоматическое конфигурирование тестового набора для запуска, конкретизация параметров тестов, построение тестового набора и запуск тестирования. Метод состоит из следующих шагов:

1) Построение таблицы параметров тестового набора. Данный этап выполняется с применением технологии VRS. Базовый протокол B(x) можно определить как выражение вида:

$$\forall x(\alpha(x) \rightarrow < P(x) > \beta(x)),$$

где x – список параметров протокола, α(x) и β(x) - логические формулы, определяющие пре- и постусловие B(x). Для получения таблицы фактических ограничений на параметры сигналов происходит последовательное применение прямого предикатного трансформера pt, позволяющего получить текущее состояние системы на основе её первоначального состояния и поведенческой трассы, ведущей к текущей точке:

$$B(x) = \forall x(\alpha(x) \rightarrow < P(x) > \beta(x)), E' = pt(E \wedge \alpha(x), \beta(x))$$

2) Уточнение пользователем способов генерации тестовых данных на основании таблицы параметров тестового набора (Рис. 1);

Решением задачи автоматизации конкретизации тестовых сценариев является созданный в ходе исследования модуль MscSubst, взаимодействующий с модулем конкретизации трасс VRS. VRS по набору символьных трасс создает таблицу конкретизации, которая служит основой тестового плана. Данный план детализируется

пользователем, который определяет желаемые значения переменных в терминах команд управления: R, M, L, O,C.

Var name	Signal name	Type	User option	Range	Value
Speed_Value	Current_Speed	I	L	-2147483648 <=Speed_Value &Speed_Value <=2	-2147483648
Speed_Value	Current_Speed	I	M	4<=Speed_Value&Speed_Value<=2147483647	107374180
Driving_Mode	MCS_Position	TRAIN_OPERATION_MODE	C	_AUTO,_CS,_RM_REV	_AUTO

Рис. 1. Пример таблицы конкретизации

Команда R управляет подстановкой значения равного правой границе интервала допустимых значений (ИДЗ) соответствующей переменной:

$$\forall R_B : \forall p_i \in P, p_i = RIGHT\ (RANGE_i)$$

Команда L управляет подстановкой левого значения ИДЗ:

$$\forall R_B : \forall p_i \in P, p_i = LEFT\ (RANGE_i)$$

Команда M предусматривает вычисление и подстановку среднего значения ИДЗ:

$$\forall R_B : \forall p_i \in P, p_i = AVG\ (RANGE_i)$$

Команда O предусматривает подстановку значения, находящегося за границами ИДЗ:

$$\forall R_B : \forall p_i \in P, p_i = X, X \notin RANGE_i$$

Команда C управляет подстановкой конкретного значения, определенного пользователем;

3) Генерация наборов значений параметров тестового набора. На основании полученного плана тестирования разработанные в ходе исследования инструмент MscSubst создаёт тестовые трассы с конкретными значениями согласно строкам таблицы:

$$TRACE_j \rightarrow \{TEST_n\} : p_i = CMD_n(RANGE_n),\ p_i \in \{R_{B_k}\},\ B_k \in TRACE_j$$

4) Генерация кода тестового набора на основании наборов тестовых данных и тестовых сценариев;
5) Выполнение тестового набора;
6) Представление результатов выполнения тестового набора с разделением тестов на прошедшие и проваленные.

Информация, полученная в результате запуска тестов, может быть использована для получения статистических и диагностических данных. Первые отражают количество пройденных и проваленных тестов. Второй тип данных используется для локализации дефектов, обнаруженных в ходе тестирования. Их поиск нетривиален, сложен для автоматизации и зачастую требует глубокого анализа полученной информации. В общем случае для её решения требуется участие человека. Рассматриваемый в данной работе **метод**

автоматизации анализа результатов тестировании и коррекции тестов частично решает данные задачи и состоит из следующих этапов:

1) Внедрение маркеров в тесты и исходные требования,
2) Выполнение тестирования,
3) Формирование вердиктов и диагностической информации на основе журналов тестирования с внедрёнными маркерами,
4) Извлечение маркеров и анализ точки завершения тестов,
5) Визуализация ошибок в тестах и требованиях,
6) Редактирование требований, тестов и наборов тестовых данных.

Метод позволяет при редактировании требований и тестов к программному продукту пользоваться результатами предыдущего прогона без необходимости их ручного анализа.

Для реализации рассматриваемого метода был разработан инструмент MscMarker, который осуществляет преобразование служебных пометок, содержащихся в поведенческих сценариях и комментариях языка MSC. Он написан на C++ и использует библиотеку ядра TAT для работы с объектным представлением языка MSC.

$$\forall MARK \in \{B, U\} \exists MARC \ (OUTCOME) : MARK \ (L) -> TEST_i$$

Из результатов тестирования можно автоматически извлечь имена базовых протоколов, между которыми произошла ошибка, а также тэг соответствующего элемента UCM диаграммы. Для отображения данной информации в средствах визуализации UCM был разработан инструмент MscGather. Он извлекает информацию о точке останова и формирует на её основе файл с пометками, которые могут быть использованы средствами визуализации BP и UCM.

При выполнении больших тестовых наборов становится актуальной задача уменьшения временных затрат. В области распределённой обработки информации для решения этой задачи используются технологии масштабирования. Применительно к тестированию понятие масштабирования можно определить как способность тестирующей системы ускорять выполнение тестового набора за счет параллельного выполнения его частей.

В ходе данного исследования были разработаны **два метода масштабирования выполнения тестового набора** – метод кластеризации группы тестов и метод параллельного запуска тестовых сценариев.

В контексте задач тестирования понятие кластеризации можно определить как разбиение набора тестовых сценариев на отдельные группы тестов. При этом необходимо обеспечить выполнение связанных между собой тестов в рамках одного кластера в необходимом порядке. В работе предложены следующие критерии кластеризации:

- Критерий количества тестов в группах;

- Критерий суммарной сложности тестов в группах. Данную величину можно выразить следующей формулой:

$$C = \sum_{i=1}^{N} n_i * c_i$$

где C – суммарная сложность тестов, ni – множитель для конкретного события или управляющей конструкции в тестовом сценарии, ci – весовой коэффициент, соответствующий выражению, N – общее количество событий в тесте.

Созданный в ходе выполнения данной работы **метод кластеризации тестового набора** состоит из следующих шагов:

1) Определение тестов, входящих в тестовый набор;
2) Определение взаимозависимостей между тестами;
3) Кластеризация тестового набора по одному из критериев, предложенных в работе;
4) Сборка и выполнение частей тестового набора;
5) Фиксация результатов выполнения тестирования.

Для реализации данного метода был разработан модуль MscSplit. Он был выполнен в виде сценария на языке Perl. Модуль кластеризации интегрирован в инфраструктуру системы TAT. Он использует служебные переменные среды, библиотеки работы с MSC и конфигурационными файлами, общие для всех инструментов.

После успешного завершения этапа кластеризации, так же, как и при его отсутствии, возможен параллельный запуск тестов. При выполнении отдельных тестов на различных процессорах системы необходимо сохранить свойства изолированности тестов и воспроизводимости тестового набора.

Предлагаемый в данной работе **метод параллельного выполнения тестового набора** состоит из следующих шагов:

1) Определение тестов, входящих в тестовый набор;
2) Определение зависимостей между тестами двух типов: необходимость совместного выполнения и невозможность совместного выполнения;
3) Сборка тестового набора;
4) Выполнение тестового набора в многопроцессорной вычислительной среде;
5) Фиксация результатов выполнения тестирования.

Данная функциональность была реализована путём изменения генераторов кода тестового набора с целью использования библиотеки, реализующей стандарт MPI. Выполнение поставленной в рамках данного аспекта задачи было разделено на два этапа: разработка параллельного алгоритма выполнения тестового набора и его реализация в коде.

Наиболее соответствующим задаче выполнения тестовых сценариев является подход, основанный на распараллеливании по данным. Минимальной подзадачей при запуске тестового набора является исполнение содержащихся в отдельном тесте команд. Цепочки взаимозависимых тестов объединяются в единую подзадачу. В рамках рассматриваемого подхода между подзадачами происходит обмен результатами выполнения тестов, вердиктами, сообщениями об ошибке и также журналами событий.

Выполнение подзадач равномерно распределяется между имеющимися процессами, при этом каждый процесс получает число подзадач, определяющееся следующей формулой N = C/p, где C – количество подзадач, полученных из тестового набора, p – количество процессов. Масштабирование производится путём пересчёта количества подзадач, получаемых одним процессом, и определение части тестового набора, подлежащей исполнению на конкретном процессе. Данный набор определяется следующей формулой:

$$T = \{t_i, t_{i+1} \dots t_{i+N}\},. \; i = N * n$$

где n – порядковый номер процесса, начинающийся с нуля, t -подзадача. Определённый таким образом алгоритм параллельного выполнения тестового набора был реализован в коде инструментов TAT. Была добавлена возможность осуществлять генерацию кода тестового набора как с поддержкой возможностей MPI, так и без таковой.

На основании инструментов, созданных для автоматизации кластеризации тестового набора возможна организация работы с ним как с сервисом путём удалённой сборки и запуска тестов, что позволяет разворачивать технологию в облачной инфраструктуре.

Разработанные методы автоматизации этапов тестирования позволяют в совокупности сформировать единую технологическую цепочку, реализующую автоматизацию контроля качества на протяжении всего жизненного цикла ПП. Она реализует **метод автоматизированного тестирования с обратной связью, основанный на анализе требований**, который разделён на тринадцать этапов. Этапы выполняются последовательно, в цепочку включены обратные связи, реализованные с помощью инструментов поддержки. Укрупняя, шаги метода можно определить следующим образом:

1) Автоматическое построение параметризованного тестового набора на основе формальной спецификации;
2) Автоматизированное формирование плана тестирования;
3) Автоматическая сборка и запуск тестового набора на основе плана тестирования с возможностью масштабирования;
4) Автоматизированная обработка результатов тестирования, коррекция требований и тестовых данных.

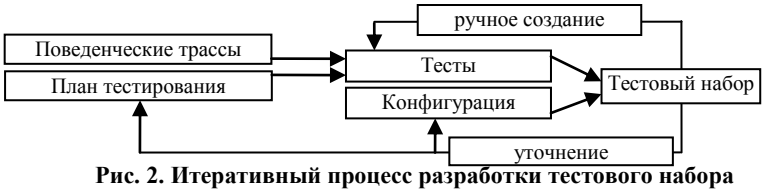

Рис. 2. Итеративный процесс разработки тестового набора

Предложенный метод тестирования позволяет организовать итеративную процедуру выполнения ряда этапов (Рис. 2). Такая модель цикла тестирования позволяет уменьшить временные затраты, а также уточнять и дополнять тесты в процессе разработки программного продукта.

В **чётвёртой главе** представлены результаты практического использования разработанных методов в рамках решения задачи тестирования в четырёх проектах. ModemInterface – проект, реализующий часть стандарта организации спутниковой связи для систем передачи видеосигнала, основан на 121 требовании. Для организации тестирования потребовалось создание библиотек-драйверов для передачи информации через механизм сокетов и файловый вывод. GMCPanel – проект, реализующий оконный пользовательский интерфейс модуля конфигурирования телекоммуникационного ПО для агрегации и обработки данных. Тестирование было основано на 54 требованиях и реализует проверку части интерфейса, отвечающей за редактирование численных параметров. Проект тестирования сервера SMTP-протокола предполагал проверку 24 требований, описывающих правила отправки электронных сообщений с хоста на сервер. EBROKER - проект тестирования веб-сервисов биржи обмена электронных валют, в рамках которого на основании WSDL-спецификации было определено 34 требования, покрывающих базовые сценарии взаимодействия клиентских приложений с сервисом.

Первым шагом, который выполнялся для каждого из проектов, являлось получение параметризованного тестового набора. Конкретизация и адаптация полученных трасс для получения тестовых сценариев, пригодных для генерации кода тестового набора, производилась с использованием разработанных инструментов. При наличии необходимых статистических данных осуществлялась оценка трудоёмкости создания аналогичного тестового набора в рамках других технологий тестирования. Запуск тестирования осуществлялся с масштабированием и без такового. Построение и запуск тестового набора проводилось как в рамках использования технологии TAT как сервиса, так и традиционным способом. Также в ходе апробации разработанного метода тестирования была произведена оценка затрат на внесение изменений в тестовый набор.

Полученные данные позволили оценить ключевые характеристики результатов работы. Сокращение временных затрат на выполнение тестов, полученное за счёт предложенных в

работе методов масштабирования выполнения тестового набора, достигло 85%, нелинейно возрастая при увеличении количества потоков выполнения (Рис. 3). Автоматизация позволила избавиться от необходимости вручную настраивать тестовый набор и управлять запуском при его раздельного выполнения.

Реализация подходов KDT и DDT с поддержкой нелинейных сценариев в рамках предложенного метода создания тестового набора привела к сокращению количества тестовых сценариев, которое необходимо для покрытия требований.

Была расширена область применения системы автоматизации TAT и методов тестирования, основанных на её использовании, благодаря реализованной функциональности. Использование стандарта MPI для метода параллельного запуска тестового набора сделало возможным запуск тестирования на вычислительных системах с кластерной архитектурой. Реализация механизмов кластеризации тестового набора позволяет одновременно выполнять запуск его частей на удалённых рабочих станциях, в том числе и в облачной среде.

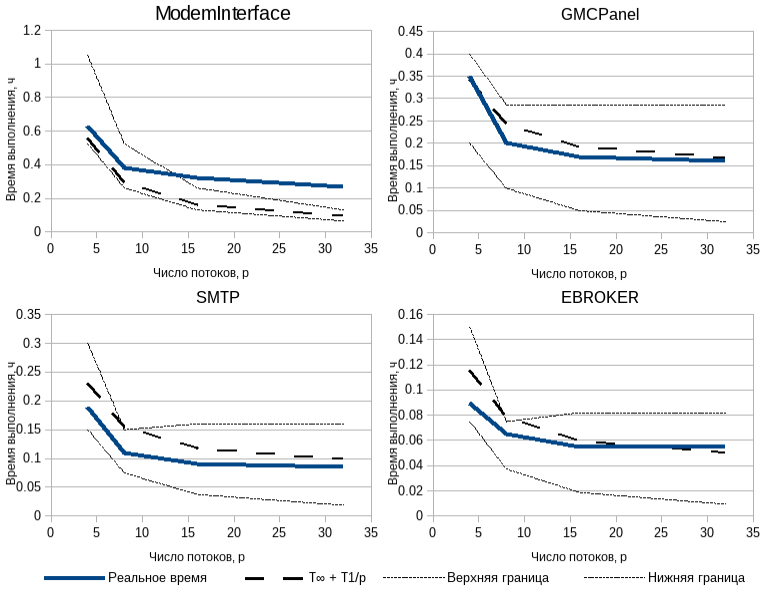

Рис. 3. Время выполнения тестового набора

В **заключении** описаны основные результаты, полученные в ходе выполнения диссертационной работы, и сформулированы возможные направления дальнейшего развития исследования.

В **приложении 1** описаны критерии тестирования, которые применяются при составлении тестового набора и определении задач тестирования в индустриальных проектах. В **приложении 2** приводится классификация видов тестирования по типам объекта тестирования. **Приложение 3** содержит описание актуальных на данный момент технологий параллельного программирования. **Приложение 4** посвящено обзору существующих инструментов и каркасов для их создания в области автоматизации тестирования. В **приложении 5** описаны формальные нотации BP и UCM и возможности их применения для задания спецификации программного

обеспечения. В **приложении 6** представлена грамматика языка GDL. **Приложение 7** посвящено описанию базовых технологий, использованных в данной работе.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1) Разработан подход к интерпретации формальной нотации MSC, реализующий возможность использования нелинейных конструкций и описания параллельных поведений в тестовых сценариях, представленных в данной нотации;
2) Сформулирован метод автоматизации создания тестового набора, основанный на применении формальных нотаций MSC и UCM и принципах DDT и KDT;
3) Разработан метод автоматизации выполнения тестового набора на основе плана тестирования, в рамках которого реализуется создание тестовых сценариев на базе параметризованных поведенческих трасс и их автоматическая конкретизация с помощью тестовых данных;
4) Разработан метод анализа результатов тестирования и коррекции тестов, позволяющий автоматизировать редактирование требований и тестового набора согласно диагностической информации о выполнении тестов;
5) Разработаны методы масштабирования выполнения тестового набора, применяющие критерии разделения тестового набора на независимые части для их параллельного выполнения;
6) Разработанные методы формализованы до уровня алгоритмов и реализованы в виде базирующегося на технологиях TAT и VRS программного обеспечения в объёме 275 KLOC. Объём разработанной сопроводительной документации составил 178 страниц;
7) Разработан метод автоматизированного тестирования, основанный на созданных программных средствах, и реализующего полный набор действий по контролю качества программного продукта в течение всего цикла его создания;
8) Методы и инструменты, созданные в ходе данной работы, апробированы в рамках решения задачи автоматизации тестирования в четырёх проектах. Согласно полученным результатам, методы позволяют не только сократить время выполнения тестового набора, но также его объём и количество операций, которые выполняются вручную при тестировании.

Полученные на этапе апробации результаты применения созданных методов и программного обеспечения позволяют сделать вывод о решении поставленных задач и достижении цели данной работы. По своей значимости её можно отнести к современным технологиям автоматизации тестирования и управления качеством программного продукта.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Основное содержание работы изложено в следующих публикациях (статьи 1–3, 6, 10 опубликованы в изданиях, включенных в перечень ВАК).

1. Средства поддержки интегрированной технологии для анализа и верификации спецификаций телекоммуникационных приложений / И.С. Ануреев, С.Н. Баранов, Д.М. Белоглазов, Е.В. Бодин, П.Д. Дробинцев, А.В. Колчин, В.П. Котляров, А.А. Летичевский, А.А. Летичевский мл., В.А. Непомнящий, И.В. Никифоров, С.В. Потиенко, Л.В. Прийма, Б.В. Тютин // Труды СПИИРАН. 2013. №3 (26). С. 349-383.
2. Тютин Б.В., Веселов А.О., Котляров В.П. Масштабирование выполнения тестового набора при автоматизированном тестировании // Научно-технические ведомости СПбГПУ. 2013. № 3 (174). С. 118-122.
3. Тютин Б.В, Никифоров И.В., Котляров В.П. Построение системы автоматизации статической и динамической проверки требований к программному продукту // Научно-технические ведомости СПбГПУ. 2012. № 4 (152). С. 119-123.
4. B.V. Tyutin, I.V. Nikiforov, V. P. Kotlyarov. Distributed Testing of Multicomponent Systems // Proceedings of the 6th Spring/Summer Young Researchers' Colloquium on Software Engineering. 2012. pp. 75-78.
5. Тютин Б.В., Веселов А.О., Котляров В.П. Разработка метода тестирования телекоммуникационных приложений в распределённой среде // Технологии Microsoft в теории и практике программирования: Материалы межвузовского конкурса-конференции студентов, аспирантов и молодых ученых Северо-Запада. 2012. С. 76-77.
6. Автоматическая настройка тестового окружения телекоммуникационных проектов / А.О. Веселов, А.С. Иванов, Б.В. Тютин, В.П. Котляров // Научно-технические ведомости СПбГПУ. 2011. № 4 (128). С. 149-152.
7. Тютин Б.В., Котляров В.П. Система управления распределённым тестированием // XXXIX Неделя науки СПбГПУ: Материалы международной научно-практической конференции. 2010. С. 118-119.
8. Иванов А.С., Тютин Б.В., Котляров В.П. Использование SDL Value Notation в качестве формата представления данных при организации удалённого взаимодействия с реактивными информационными системами // XXXVIII Неделя науки СПбГПУ: Материалы международной научно-практической конференции. 2009. С. 106-108.
9. Автоматизация нагрузочного тестирования сетевых служб / А.О. Веселов, А.С. Иванов, Б.В. Тютин, В.П. Котляров // Технологии Microsoft в теории и практике программирования: Материалы межвузовского конкурса-конференции студентов, аспирантов и молодых ученых Северо-Запада. 2009. С. 24-25.
10. Автоматизация тестирования телекоммуникационных приложений / А.О. Веселов, А.С. Иванов, Б.В. Тютин, В.П. Котляров // Научно-технические ведомости СПбГПУ. 2009. № 3 (80). С. 208-212.

Подписано в печать ДД.ММ.2013. Формат 60x84/16. Печать цифровая.
Усл. печ. л. 1,0. Тираж XXX. Заказ XXXXX.

Отпечатано с готового оригинал-макета, предоставленного автором,
в типографии Издательства Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 550-40-14
Тел./факс: (812) 297-57-76