На правах рукописи

Голубев Алексей Андреевич

Методики создания и внедрения агентов в прикладное и системное программное обеспечение для автоматизации тестирования и мониторинга встроенных вычислительных 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 - 2007

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 физико-математических наук, доцент
Кознов Дмитрий Владимирович

Ведущая организация - Федеральное государственное унитарное предприятие «Научно-производственное объединение «Импульс»

Защита состоится « 1 » ноября 2007 г. в 16 часов на заседании диссертационного совета Д 212.229.18 при ГОУ 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 ГОУ ВПО «Санкт-Петербургский государственный политехнический университет».

Автореферат разослан « 28 » сентября 2007 г.

Ученый секретарь
диссертационного совета Шашихин В.Н

1. ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

1.1 Актуальность работы. В процессе разработки программного обеспечения (ПО) важную роль играет автоматизация тестирования совместной работы разных типов приложений вычислительных систем. Трудозатраты по интеграции указанных типов тестирования определяют длительность цикла разработки и качество ПО.

С развитием Java технологии для встроенных вычислительных систем существенно возросли требования к сложности, качеству и трудоемкости разработки приложений, объединяющих платформенные и Java компоненты. Основные свойства технологий автоматизации тестирования, поддерживающих такое применение, следующие:

– использование формальных спецификаций на ранних этапах разработки;
– поддержка автоматической генерации кода тестов в соответствии с особенностями целевой платформы;
– возможность наблюдения внутреннего поведения тестируемой системы посредством генерации воздействий и обработки откликов;
– автоматизированный прогон тестовых циклов, сбор и анализ результатов;
– гибкая настройка и адаптация к новым версиям программных платформ.

Перечисленные свойства подтверждают необходимость создания методик и инструментальных средств технологии автоматизации тестирования встроенных систем, поддерживающих тестирование свойств совместного функционирования платформенных и Java компонентов. Анализ предметной области проведен по двум основным направлениям: анализ существующего инструментария для автоматической генерации кода тестов из формальных описаний; анализ существующих методов и средств, обеспечивающих возможность автоматизации тестирования встроенных Java систем на этапах прогона тестов, сбора и анализа результатов.

Анализ существующего инструментария для автоматической генерации кода тестов из формальных описаний: TVG, MulSaw, GOTCHA-TCBEAN, AsmL Test Tool, UniTesK, VRS/TAT – показал актуальность использования для этих целей инструмента VRS/TAT, удовлетворяющего требованиям современного процесса разработки ПО встроенных вычислительных систем, включающих в себя платформенные и Java компоненты. Преимущества выбранного инструмента: возможность гибкой настройки целевого языка кода тестов, автоматическая генерация оптимального набора тестовых сценариев по формальным описаниям требований и генерация кода тестов для соответствующей платформы Java ME, что не поддерживается в других рассмотренных промышленных средствах автоматизации тестирования.

Анализ существующих методов и средств автоматизации тестирования встроенных Java систем на этапах выполнения тестов, сбора и анализа результатов показал, что в настоящее время в исследуемой предметной области используются технологии внедрения агентов, позволяющих наблюдать поведение тестируемой системы (TCK, Java Device Test Harness, Communology MTE, Mobile Complete Live Test, PTF, FlexAuto). Выявленные недостатки не позволили использовать их для решения поставленной в работе задачи по следующим причинам: узкая специализированная направленность рассмотренных средств не обеспечивает гибкой настройки на платформу; отсутствие средств, поддерживающих тестирование платформенных и Java приложений, и их взаимодействий; избыточная трудоемкость расширения тестовых наборов, включенных в рассмотренные средства; невозможность свободной интеграции с системами генерации кода тестов для искомой платформы.

На основе анализа работ В.И. Городецкого, В.Б. Тарасова, Р. Брукса, Н. Дженнингса и других авторов по проектированию агентов сделаны выводы о перспективности использования агентов для обеспечения возможности наблюдения и управления на программном уровне, и о необходимости адаптации существующих подходов к разработке агентов, исходя из специфики исследуемых встроенных систем.

В диссертационной работе автором разработаны методики создания и внедрения программных агентов в прикладное и системное ПО встроенных Java систем. Применение предлагаемых методик для автоматизации этапов выполнения тестов, сбора и анализа результатов совместно с системой VRS/TAT для генерации кода тестов по формальным спецификациям позволяет создать технологию автоматизации тестирования платформенных и Java приложений. Технология, созданная на базе предложенных методик, позволяет в несколько раз сократить длительность и трудоемкость процесса тестирования при отсутствии регрессии качества, что и определяет актуальность диссертационной работы. Результаты анализа предметной области позволили сформулировать цели и задачи исследования и определить его структуру.

1.2 Цели и задачи диссертационной работы.

Целью диссертационной работы является разработка комплекса методик встраивания адаптивных программных агентов, позволяющих осуществлять наблюдение и управление внутренним поведением тестируемой системы для обеспечения автоматизации этапов прогона тестов, сбора и анализа результатов, сокращающей трудозатраты и время фазы тестирования. Агенты должны обеспечивать наблюдаемость поведения прикладного и системного ПО, а также их взаимодействий; поддерживать возможность гибкой адаптации к целевой платформе. Методики должны обеспечивать

возможность создания технологии автоматизации тестирования с использованием средств автоматической кодогенерации по формальным описаниям. В поддержку технологии должны быть разработаны настраиваемые на платформу Java ME инструментальные средства. В рамках достижения цели в работе решены следующие задачи:

- разработка требований к встраиваемым программным агентам на основе проведенного анализа существующих методов и средств автоматизации тестирования;
- создание поведенческих моделей основных архитектурных компонентов встроенных Java систем и модели универсального агента, инвариантных относительно платформы;
- разработка обобщенной методики встраивания агентов на основе предложенных моделей;
- доказательство корректности разработанных моделей, их функционирования и взаимодействия с использованием инструментов автоматической верификации;
- разработка концепции программной реализации и адаптации к целевой платформе методик встраивания агентов в прикладное и системное ПО на уровне основных архитектурных компонентов вычислительных систем, обеспечивающих управление и наблюдаемость поведения платформенных и Java приложений, и их взаимодействий;
- создание программного инструментария, обеспечивающего настройку и интеграцию агентов с тестируемой системой;
- проверка работоспособности и эффективности предложенных методик и инструментальных средств в 6 проектах для двух различных архитектур встроенных мобильных платформ.

1.3 Предметом исследования являются методы, средства и программный инструментарий автоматизации тестирования и мониторинга поведения встроенных мобильных платформ.

1.4 Методы исследования. В диссертации использованы результаты теории реактивных и транзитивных систем, конечных автоматов и базовых протоколов, аппарат формальных спецификаций, концепция объектно-ориентированных моделей. Основными критериями являлись универсальность, технологичность, простота использования и адаптации разработанных методик, а также возможность интеграции с существующими средствами автоматической генерации кода тестов. Применялись стандарты UML, MSC, ANSI C и Java. В основу исследований положен системный подход.

1.5 Обоснованность и достоверность полученных результатов обеспечена корректным использованием теорий реактивных и транзитивных систем, конечных автоматов и базовых протоколов; использованием строгого аппарата формальных спецификаций; положительными итогами использования разработанных методик и программных средств в реальных проектах; совпадением результатов по достижению качества ПО, полученных различными методами.

1.6 Научные результаты и их новизна. На защиту выносятся следующие научные результаты работы:

1. Модели архитектурных компонентов встроенных Java систем и модель универсального агента в виде реактивных систем переходов.
2. Обобщенная методика встраивания агентов, разработанная на основе моделей.
3. Доказательство корректности функционирования и безопасности встраивания агентов с точки зрения сохранения логики работы межкомпонентных интерфейсов на основе верификации моделей и их взаимодействий.
4. Специализированные методики встраивания агентов, позволяющие обеспечить наблюдение и управление в различных задачах тестирования и мониторинга: методика встраивания агентов на уровне Java ME приложений; методика встраивания агентов на уровне kJava виртуальной машины (KVM); методика встраивания агентов на уровне платформенного окружения.
5. Алгоритм анализа и модификации байт-кода Java ME приложения, обеспечивающий возможность наблюдения за внутренним поведением тестируемого приложения и генерации воздействий встроенным агентом.
6. Методика для ускорения выбора путей ветвления тестовых сценариев на основе выбора альтернатив с помощью сигнатур.

1.7 Практическая значимость работы. На базе полученных научных результатов разработан комплекс программных средств, использующий методики встраивания агентов для автоматизации тестирования прикладного и системного ПО, и их взаимодействий. Программный комплекс использован в компании Motorola в 6 программных проектах для двух различных платформенных архитектур в таких областях как: разработка Java ME приложений (MIDlet, i-appli, CORElet), сертификационное и компонентное тестирование встроенных Java платформ. Созданные методики и программные средства являются универсальными и могут быть использованы для автоматизации тестирования и мониторинга прикладного и системного ПО любых современных платформенных архитектур встроенных вычислительных систем. Применение технологии автоматизации тестирования на базе методик встраивания агентов позволяет в среднем сократить время

фазы прогона тестов, сбора и анализа результатов в 6 раз по сравнению с ручным подходом и в 1,7 раза по сравнению с существующими подходами автоматизации.

1.8 Апробация работы. Основные результаты и выводы диссертации докладывались на следующих международных научных конференциях: «IEEE Russia Northwest Section, 110 Anniversary of Radio Invention conference» (СПб., 2005 г.); «2006 IEEE Tenth International Symposium on Consumer Electronics» (СПб., 2006 г); Motorola Technology Day (Spb 2005, 2006); конференциях «Технологии Microsoft в теории и практике программирования» 2004, 2005 и 2006 гг.; конференциях XXXII–2004 г, XXXIII–2005 г, XXXIV–2006 г недели науки СПбГПУ. По материалам диссертации опубликовано 6 печатных работ, в том числе одна статья в издании, входящем в список рекомендованных изданий ВАК.

1.9 Внедрение. Методики встраивания агентов для исследуемых вычислительных систем и технология автоматизации тестирования, разработанная на их базе, внедрены в ЗАО «Северо-Западная Лаборатория Лтд.», ОАО «Интелтех», ЗАО «Моторола ЗАО» и использовались при разработке учебно-методического комплекса СПбГПУ по курсу «Технология индустриального программирования» на кафедре ИУС. Практическое использование представляемых на защиту результатов подтверждено соответствующими актами о внедрении.

1.10 Структура и объём работы. Работа содержит введение, 5 глав, заключение и 5 приложений. Объём работы 150 страниц, количество иллюстраций 69, список использованной литературы состоит из 112 наименований.

2. СОДЕРЖАНИЕ РАБОТЫ

В **первой главе** диссертационной работы проанализированы особенности предметной области и направления совершенствования современных методов и средств автоматизации тестирования и мониторинга встроенных вычислительных систем:

1. Необходимость совершенствования существующих методов процесса тестирования для увеличения степени автоматизации, обеспечивающей повышение эффективности и уменьшение трудоемкости.
2. Возможность автоматизации тестирования встроенных мобильных платформ на этапах прогона тестов, сбора и анализа результатов, основанная на применении программных агентов для обеспечения наблюдаемости состояний и управления внутренним поведением тестируемой системы.
3. Проведенный анализ позволил выявить следующие недостатки существующих методов и средств встраивания агентов с позиций автоматизации тестирования в рассматриваемой предметной области:

- узкая специализация всех методов и средства, исключающая возможность гибкой настройки для решения различных задач автоматизации тестирования;
- отсутствует возможность расширения тестовых наборов, поставляемых вместе со средствами встраивания агентов;
- затруднен процесс сопряжения интерфейсов агента с автоматически сгенерированными из формальных описаний тестовыми наборами;
- ограничены возможности контроля и анализа графических образов.

4. Необходимость адаптации существующих подходов к разработке агентов вследствие ресурсных ограничений и специфики решаемых задач в исследуемой предметной области встроенных Java систем.

5. Проведенный анализ методов и средств автоматизации тестирования на этапе генерации кода тестов по формальным описаниям, выявил возможность применения системы VRS/TAT, поддерживающей адаптацию автоматической генерации кода на целевой язык Java ME тестируемой системы.

Полученные результаты анализа позволили установить отсутствие и необходимость создания универсальных и эффективных методик встраивания агентов для автоматизации этапов прогона тестов, сбора и анализа результатов, обеспечивающих возможность создания наиболее полной технологии автоматизации тестирования исследуемых мобильных платформ.

Во **второй главе** на основе теории реактивных систем разработаны модели основных архитектурных компонентов встроенных Java систем в виде систем переходов. Обосновано введение понятия системы переходов с наблюдаемым временем – промежуточный вид между простой системой переходов и системой временных переходов. В зависимости от целей и задач можно перейти к системе переходов (путем отбрасывания времени) или к системе временных переходов (путем введения временных ограничений). Системой переходов с наблюдаемым временем называется пара $S^t =< S, t >$, где S – система переходов, определяемая как $S = \{X, \Sigma, \mathrm{T}, \Theta\}$ (где X – множество переменных над областью определения D, $\Sigma : \{X \to D\}$ – множество состояний системы, $\Theta \in \Sigma$ – множество начальных состояний, T – конечное множество переходов), а t – множество моментов времени срабатывания переходов. Вычислением системы переходов с наблюдаемым временем будет цепочка $\sigma \in \Sigma^*, \sigma = s_0 s_1 \ldots s_n$, где $s_0 \in \Theta$, $(\forall i)(\exists ! \tau \in \mathrm{T}) : \tau(s_i) = s_{i+1}$, и дополнительное условие $(\forall i)(\forall t_i \in t) : t_i < t_{i+1}$, которое как раз и показывает, что функционирование системы происходит во времени.

Основными архитектурными компонентами встроенных Java систем являются: Java ME приложение (S^t_{App}), виртуальная Java машина (KVM) или системное приложение (S^t_{KVM}), низкоуровневое окружение или платформа (S^t_{Env}). Разработаны реактивные модели в виде систем переходов с наблюдаемым временем всех трех компонентов (рис. 1) и описано их функционирование.

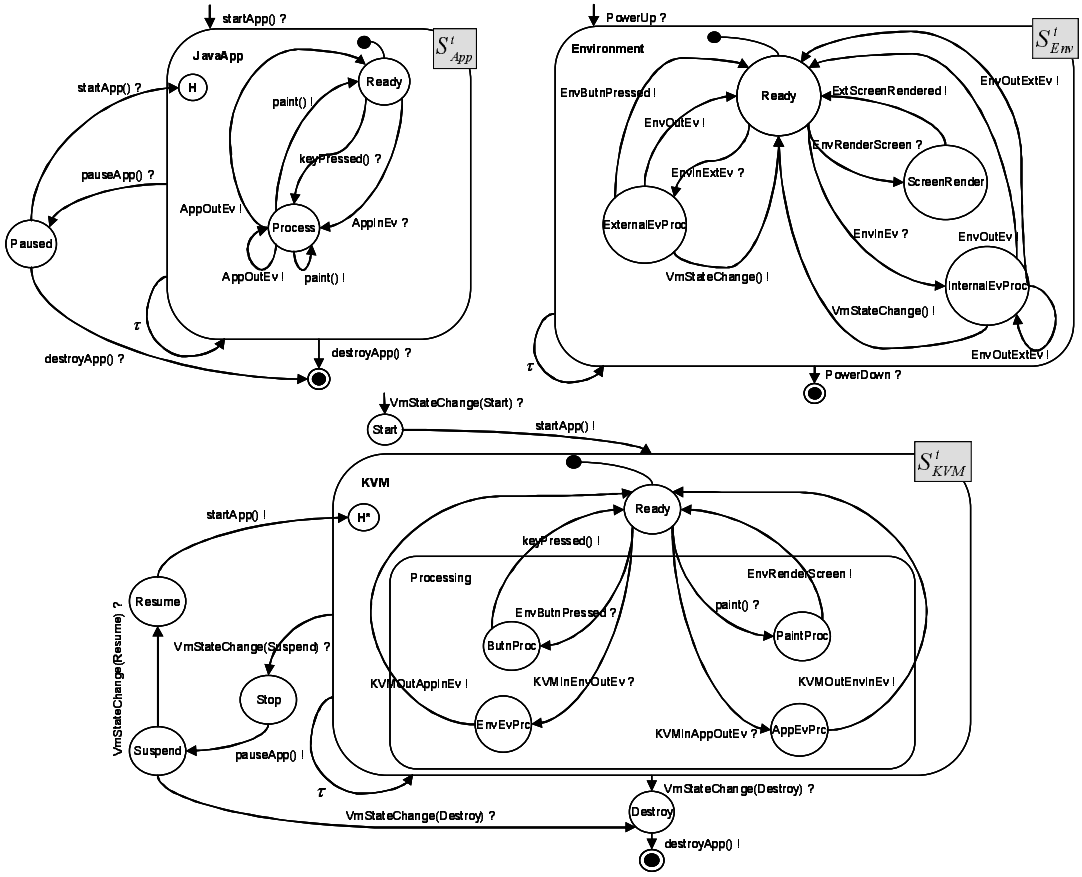

Рис. 1 Модели архитектурных компонентов встроенных Java систем

На основе моделей компонентов разработана интегрированная модель их взаимодействия (рис. 2), которая является дискретно-параллельной системой переходов с наблюдаемым временем, представленной в виде параллельной композиции систем переходов отдельных компонентов: $S^t_{Int} = S^t_{App} \times S^t_{KVM} \times S^t_{Env}$.

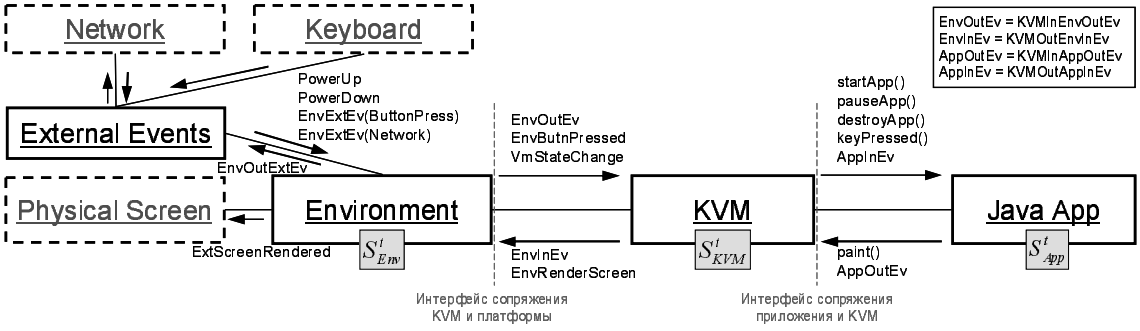

Рис. 2 Интегрированная модель взаимодействия компонентов встроенной Java системы

В местах сопряжения интерфейсов (пунктирные линии на рис. 2) необходимо обеспечить возможность программного управления и наблюдения для решения задач автоматизации тестирования и мониторинга. Для обеспечения возможности программного управления и наблюдения за интерфейсами внутри закрытых вычислительных систем разработана модель универсального встраиваемого агента в виде системы переходов с наблюдаемым временем (рис. 3) – S_A^t.

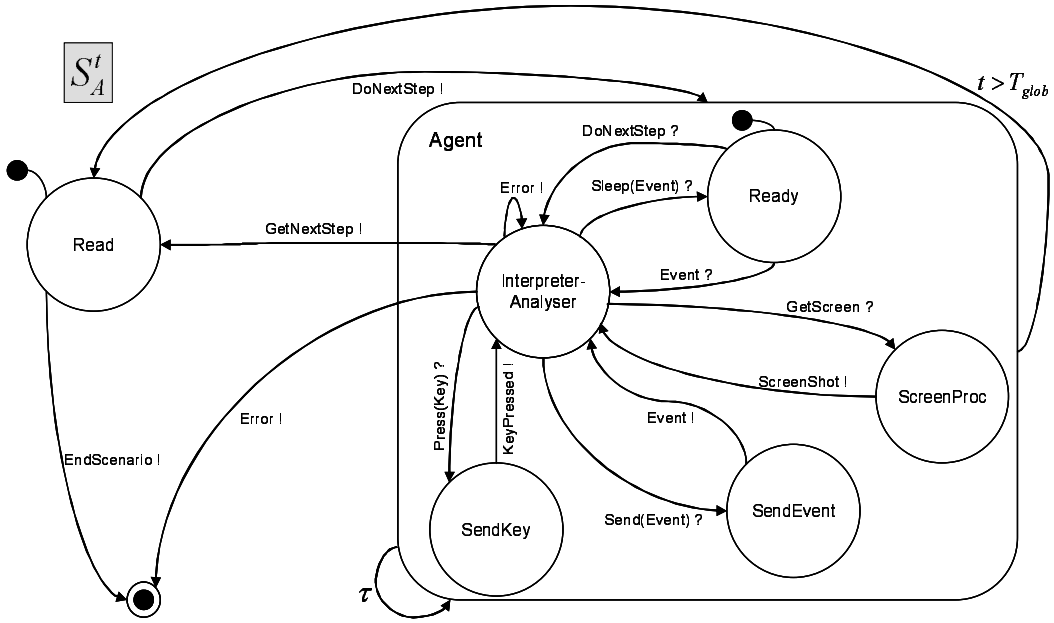

Рис. 3 Система переходов универсального встраиваемого агента

Модель агента может функционировать в двух режимах: активном и пассивном, первый режим обеспечивает поддержку автоматизации тестирования, второй – мониторинга. Возможности универсальной модели агента продемонстрированы с помощью двух теоретических методик встраивания:

1. **Методика встраивания агентов для Java ME приложения**. В данном случае тестируемой системой является приложение – S_{App}^t, а окружением для него – KVM – S_{KVM}^t. Тогда встраивание агента возможно между этими компонентами после приведения множества генерируемых агентом событий и множества наблюдаемых событий в соответствие с множествами соответствующих событий тестируемой системы и окружения: $S_{Im+jA}^t = (S_{App}^t \times S_A^t \times S_{KVM}^t) \times S_{Env}^t$.

2. **Методика встраивания агентов на уровне окружения**. Тестируемой системой является KVM – S_{KVM}^t (или иные системные платформенные (native) приложения), а тестовым окружением является уже платформенное окружение – S_{Env}^t. Встраивание агента

(после согласования событий и сопряжения интерфейсов) осуществляется путем композиции: $S^t_{Int+nA} = S^t_{App} \times (S^t_{KVM} \times S^t_A \times S^t_{Env})$.

Для всех описанных моделей компонентов (KVM, приложения, платформы, агента), а также для интегрированных моделей с разными типами агентов была проверена корректность функционирования с помощью автоматической верификации с использованием системы VRS. Для верификации производится генерация трасс посредством последовательного применения базовых протоколов, результатом чего является начальное множество трасс базовых протоколов: $s_1 \xrightarrow{B_1} s_2 \xrightarrow{B_2} \ldots \xrightarrow{B_{n-1}} s_n$, где $B_1, B_2, ..., B_{n-1}$ – применяемые базовые протоколы, $s_1, s_2, ..., s_n$ – промежуточные состояния. Далее происходит перебор базовых протоколов, и выявляются классы функциональной эквивалентности, путем проверки возможности перестановок, в результате строится дерево трасс базовых протоколов, по которому происходит генерация трасс с учетом верифицируемых условий, которые проверяют и доказывают все исследуемые свойства моделей (отсутствие тупиков, достижимость состояний, наличие или отсутствие недетерминизмов). В процессе верификации корректность моделей была доказана автоматически. Последовательность обмена событиями между компонентами после внедрения агентов в систему сохраняется. Это подтверждает безопасность встраивания агентов для логики взаимодействия компонентов в системе.

На основе разработанных моделей и проведенного анализа возможностей модели универсального встраиваемого агента, сформулированы основные этапы **обобщенной методики встраивания агентов**:

1. Выделить у тестируемой системы и окружения необходимые для управления и наблюдения входные и выходные интерфейсы.
2. Разделить выделенные интерфейсы по трем основным классам событий агента (клавиатура, экран, прочие события).
3. Отобразить интерфейсы тестируемой системы и окружения в область событий агента.
4. Детализировать универсальную модель системы переходов агента на формальном языке, с указанием всех необходимых событий (как входных, так и выходных).
5. По формальной спецификации требований к агенту разработать код агента на целевом языке (возможно использование методов кодогенерации).
6. Специфицировать входные и выходные интерфейсы агента для взаимодействия с тестовыми наборами.

Разработанная универсальная модель агента учитывает специфику функционирования основных архитектурных компонентов исследуемых встроенных вычислительных систем, что позволяет решать задачу создания специализированных агентов, сопряженных с разрабатываемым программным обеспечением, при сохранении универсальности общего подхода. Таким образом, вторая глава содержит теоретическую и обобщенную методическую базу для реализации методик встраивания агентов и создания на их базе технологии автоматизации тестирования и мониторинга.

В **третьей главе** разработаны концепции реализации обобщенной методики встраивания агентов для автоматизации этапов прогона тестов, сбора и анализа результатов, применимые к разным компонентам вычислительных систем.

1. **Концепция реализации методики для тестирования встроенных Java ME приложений**. Предложенная реализация учитывает специфические свойства языка Java и основывается на использовании библиотеки классов-оболочек и перехвате управления потока исполнения приложения. Классы-оболочки обеспечивают возможность наблюдения внутренних программных интерфейсов, а перехват управления потока исполнения приложения – возможность генерировать управляющие воздействия. Для тестирования системных Java ME приложений (CORElet) и приложений, имеющих строгие ограничения на размер (i-appli приложения) разработана дополнительная разновидность методики – **методика встраивания агентов в KVM**. Параллельная композиция моделей агента, тестируемой системы и окружения выглядит следующим образом: $S^t_{Im+jnA} = S^t_{App} \times (S^t_A \times S^t_{KVM}) \times S^t_{Env}$. Для реализации методики встраивания агентов в KVM также предложено использовать перехват управления потока исполнения приложения, но вместо библиотеки классов-оболочек – низкоуровневую библиотеку классов, встроенную в KVM и обеспечивающую возможность генерации входных воздействий (нажатий клавиш), а также сохранения содержимого графического буфера экрана для последующего анализа. Основными этапами методики встраивания агентов на уровне Java ME приложений являются:

– разработка библиотеки классов-оболочек используемого приложением профайла (MIDP, DoJa) для стандартных API;
– разработка платформенного API, в случае, если тестируемые приложения используют не стандартизированные закрытые API или существуют жесткие ограничения на размер приложения;
– определение сравнительной эффективности использования обоих подходов для тестирования конкретного Java ME приложения – классов-оболочек или платформенного API, также возможна комбинация этих подходов;

– инструментация приложения дополнительными библиотеками, открытие внутренних интерфейсов, передача управления агенту;
– интеграция инструментированного приложения с тестовым набором.

2. **Концепция интеграции тестового набора, агента и тестируемого Java ME приложения**. Проведенный в работе анализ показал неэффективность модификации Java ME приложения на уровне исходного кода для интеграции с агентом и тестовым набором. Для инструментирования тестируемого приложения и интеграции с тестовым набором и агентом разработан **алгоритм анализа и модификации байт-кода**, основными этапами которого являются:

– изменение атрибутов доступа к полям и методам классов приложения (для получения доступа к ним и возможности их вызова);
– замена наследования классов приложения (для передачи управления);
– замена имен библиотечных классов на имена классов-оболочек.

Разработанный подход модификации байт-кода удовлетворяет следующим требованиям: новое приложение совместимо на уровне байт-кода с оригинальным в области своей работы; логика работы приложения не изменяется.

3. **Концепция реализации методики для тестирования системных платформенных приложений**. Разработано две версии реализации методики – для встраивания пассивного и активного агентов. Предложен подход к программной реализации методик, а также методика адаптации к целевой платформе.

Методика встраивания пассивного агента на уровне окружения включает в себя следующие этапы:

– идентификация необходимых для наблюдения событий системных приложений и платформенного окружения;
– разделение наблюдаемых интерфейсов по трем основным классам событий агента: клавиатура, экран, прочие события;
– детализация универсальной модели агента на формальном языке, с указанием всех необходимых для наблюдения событий (входных событий агента);
– разработка кода агента;
– интеграция агента в окружение;
– разработка конфигурационного файла со списком наблюдаемых событий.

Основные этапы **методики встраивания активного агента на уровне окружения** представлены на рис. 4. Для реализации встраиваемого агента предложено использовать структуру фонового платформенного приложения, а также разработанный универсальный интерпретатор скрипта тестовых сценариев. Для платформы с DoJa описаны этапы

программной реализации кода агента. Для адаптации кода агента к целевой платформе необходимо: определить структуру фонового платформенного приложения; реализовать функции регистрации приложения и обработки событий в соответствии с платформенной структурой; осуществить отображение множества имен входных и выходных событий агента на множество имен событий платформы, а также определить функции считывания экрана из драйверного буфера и генерации нажатий клавиш через драйвер клавиатуры; портировать универсальный интерпретатор скрипта тестовых сценариев.

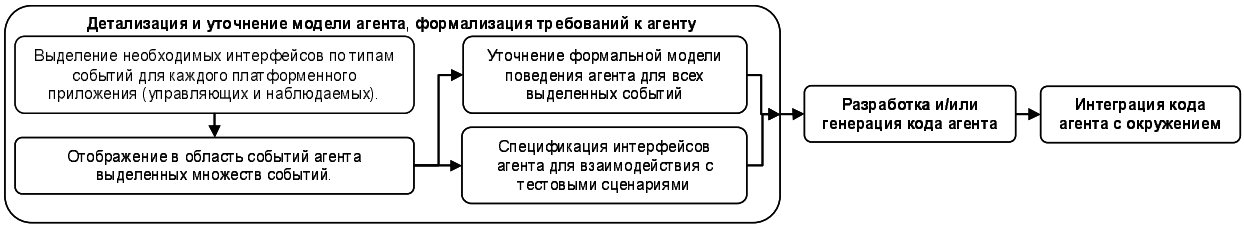

Рис. 4 Основные этапы методики встраивания активного агента на уровне окружения

4. **Методика выбора альтернатив на основе контрольных сумм** обеспечивает эффективную поддержку недетерминированных тестовых сценариев, когда выбор осуществляется по графическому содержимому экрана. Проведенный анализ показал возможность эффективного использования сигнатур, посчитанных по алгоритму CRC-32, при сравнении данных экрана с множеством эталонов. Разработанный подход позволяет существенно сократить использование памяти и уменьшить время сравнения. Основные этапы методики выбора альтернатив на основе контрольных сумм:

– определение возможности использования различных схем сравнения: сравнение на основе сигнатур полных экранов, сравнение на основе сигнатур отдельных областей («масок»), поэлементное сравнение содержимого экрана с эталонами;
– подготовка сигнатур эталонов для тестового набора на компьютере с помощью специального инструментария;
– подготовка тестовых сценариев с использованием необходимой функции выбора – на основе сигнатур или поэлементной.

В четвёртой главе на основе предложенных моделей и методик построена технология автоматизации тестирования встроенных Java систем (рис. 5). Технология реализована с использованием системы VRS/TAT для поддержки верификации требований и автоматической генерации тестовых наборов по формальным описаниям и расширена модулями встраивания агентов для различных платформ. В главе 4 описан программный комплекс автоматизации тестирования встроенных Java систем, разработанный на основе предложенной технологической цепочки, и включающий в себя: систему VRS/TAT, программно реализованные методики встраивания агентов, модуль байт-код анализатора, инструменты для настройки системы автоматизации тестирования.

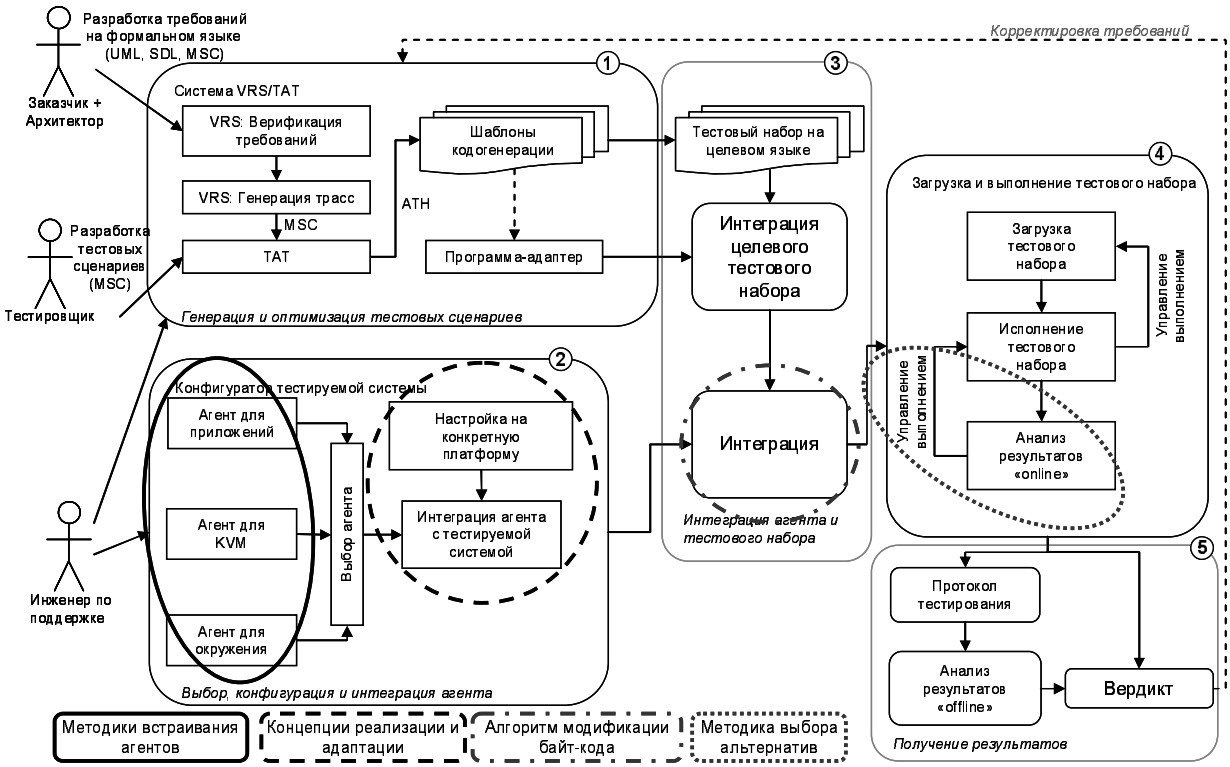

Рис. 5 Технологическая цепочка использования системы автоматизации тестирования

В пятой главе представлены результаты применения разработанных методик и технологии автоматизации тестирования в реальных проектах по разработке программного обеспечения для встроенных Java систем. При этом для каждого проекта применялись методики, соответствующие постановке задачи.

Пилотирование и сбор статистики по автоматизации тестирования Java ME приложений осуществлялись в трех реальных проектах: MIDlet «DAP» (применялась методика встраивания агентов на уровне приложений), CORElet «SynerJ ImageEditor» (применялась методика встраивания агентов на уровне KVM), i-appli приложение «DoJa ImageEditor» (применялась комбинация методик). На рис. 6 (1) представлены временные и человеческие затраты на один тестовый цикл ручного и автоматизированного подхода к тестированию для каждого проекта. При автоматизированном подходе получается усредненный выигрыш в 6 раз по затратам человеко-часов.

Для автоматизации тестирования системного ПО применялась методика встраивания агентов на уровне платформенного окружения. Пилотирование технологии для платформы с DoJa проходило в рамках внедрения в процесс сертификационного и компонентного тестирования. Для сертификационного тестирования удалось увеличить степень автоматизации с 75% до 98%, что обеспечило сокращение временных человеческих трудозатрат на фазе прогона тестов, сбора и анализа результатов более чем в 20 раз по сравнению с ручным подходом. Для компонентного тестирования удалось добиться 99%-ной степени автоматизации. При большом количестве тестовых циклов

(например, при приемочном sanity-тестировании) суммарные временные затраты в автоматизированном подходе уже при 7-8 циклах тестирования сравнимы с однократным ручным прогоном. Использование предложенной технологии автоматизации дает выигрыш по человеческим трудозатратам, в среднем, более чем в 30 раз, выигрыш только по времени – в 7,6 раз. При этом 75% времени составляет время работы мобильного устройства без участия тестировщика.

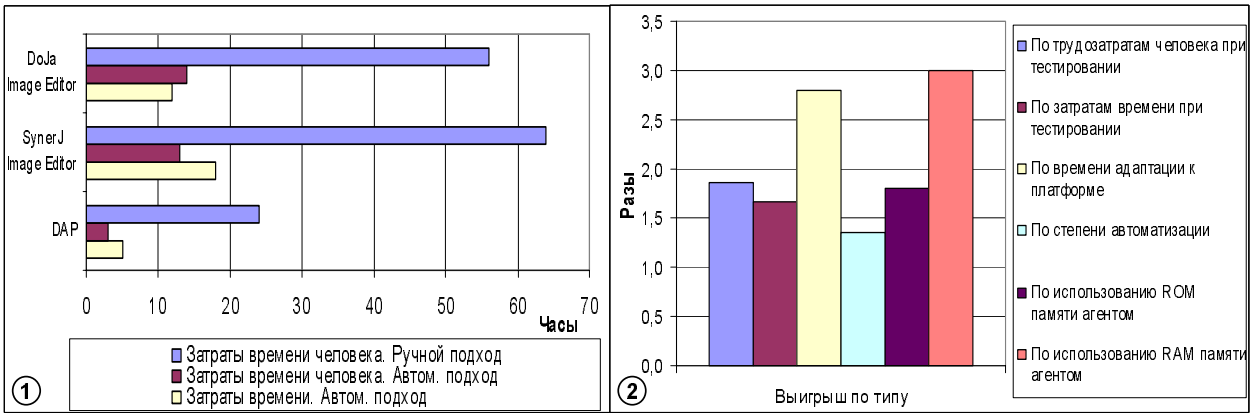

Рис. 6 Соотношение временных и трудозатрат при автоматическом и ручном подходе тестирования приложений (1) и выигрыш по сравнению с технологией автоматизации на основе PTF-агента (2)

Стоит отметить, что применения методик встраивания агентов в рамках технологии автоматизации тестирования улучшает качество ПО, в среднем, на 9-12%, за счет увеличения количества выявленных дефектов при автоматизированном подходе в рамках одного тестового цикла, проводимого параллельно с ручным подходом.

Затраты на встраивание агентов для различных применений приведены в таблице 1.

Таблица 1 Затраты ресурсов при применении методик встраивания агентов

	Методика встраивания агентов		
	На уровне приложения	На уровне KVM	На уровне окружения
Затраты на адаптацию к новому Java ME профайлу (человеко-дни)	3	-	-
Затраты на разработку агента под новую платформу (человеко-дни)	-	1	20
Затраты на адаптацию к новой платформе (человеко-дни)	-	5	5
Размер библиотеки классов-оболочек (Кб)	14	0	-
Использование ROM-памяти агентом (Кб)	-	5,5	35,0
Использование RAM-памяти агентом (Кб)	-	0,5	0,5
Время на подготовку тест.набора (человеко-часы)	0,5	1	1
Средний размер одного тестового сценария (Кб)	10	1	300
Необходимый объем дополнительного свободного места в файловой системе (усредненный, Кб)	-	300	7300

Для всех методик прослеживаются общие тенденции: время на адаптацию обычно меньше времени прогона одного ручного цикла тестирования; размеры агентов малы в сравнении с размерами компонентов, в которые они встраиваются; время на подготовку

тестового набора мало; при увеличении размера тестового сценария существует возможность использования файловой системы платформы и внешних карт памяти.

В пятой главе проводится сравнение разработанных методик с существующим подходом автоматизации тестирования мобильных платформ на базе PTF-агента. Собранные статистические данные в проектах, поддерживаемых PTF (DAP, компонентное тестирования платформ с MIDP), показывают, что использование разработанных методик встраивания агентов, в среднем, в 2 раза эффективнее существующего подхода на основе PTF-агента при том же уровне качества. На рис. 6 (2) показаны выигрыши от применения универсального агента по сравнению с PTF-агентом по трудозатратам и времени тестирования, времени на адаптацию агентов, используемым ресурсам. Также наблюдается увеличение степени автоматизации (с 72% при использовании PTF-агента до 97,5% при использовании универсального агента) за счет методики выбора альтернатив и поддержки автономного параллельного тестирования.

3. ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

В диссертационной работе решена научная задача разработки методик создания и внедрения агентов для обеспечения возможности автоматизации тестирования и мониторинга встроенных мобильных платформ, что позволяет существенно сократить трудозатраты и время фазы тестирования. Основными результатами диссертационной работы являются:

1. Разработка и доказательство корректности поведенческих моделей основных архитектурных компонентов встроенных Java систем и модели универсального агента, на основе которых предложена обобщенная методика встраивания агентов, обеспечивающая возможность программного управления и наблюдения за внутренними интерфейсами вычислительных систем.
2. Создание концептуальных основ реализации методик встраивания агентов на уровне основных архитектурных компонентов мобильных платформ для решения различных задач автоматизации тестирования и мониторинга. В том числе: алгоритм анализа и модификации байт-кода для поддержки встраивания агентов в Java ME приложения; методика выбора альтернатив на основе контрольных сумм.
3. Создание технологии автоматизации тестирования встроенных мобильных платформ на основе использования разработанных методик и системы VRS/TAT.
4. Разработка программного комплекса поддержки технологии автоматизации тестирования встроенных мобильных платформ.
5. Оценка эффективности разработанных методик и ПО на базе использования в программных проектах различной сложности в ЗАО «Motorola ЗАО», ЗАО «Северо-

Западная Лаборатория Лтд.», ОАО «Интелтех» позволяет установить, что при сравнительно небольших трудозатратах на встраивание и потребляемых агентом ресурсов, применение методик обеспечивает сокращение длительности фазы прогона тестов, сбора и анализа результатов, в среднем, в 6 раз по сравнению с ручным подходом и в 1,7 раза по сравнению с существующими подходами автоматизации.

Результаты, полученные в процессе выполнения проектов с использованием предложенной технологии на базе разработанных методик встраивания агентов, позволяют сделать выводы о работоспособности и эффективности методов и средств создания и внедрения агентов для автоматизации тестирования и мониторинга прикладного и системного ПО встроенных вычислительных систем.

4. СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

1. Голубев А.А. Методы разработки расширений (API) и ускорение KJAVA виртуальной машины для мобильных устройств // Технологии Microsoft в теории и практике программирования: Материалы межвузовского конкурса-конференции студентов и молодых ученых Северо-Запада. – СПб.: Изд-во СПбГПУ, 2004. – С.19-20.
2. Поддубный В.Е., Полубенцева И.К., Голубев А.А., Котляров В.П. Система автоматизации тестирования J2ME-приложений // XXXIII неделя науки СПбГПУ: Материалы Всероссийской межвузовской научно-технической конференции студентов и аспирантов, Ч.V. – СПб.: Изд-во Политехн. ун-та, 2005. – С.32-34.
3. Голубев А.А., Карпов А.Н., Котляров В.П. Автоматизация тестирования системных J2ME приложений // Технологии Microsoft в теории и практике программирования: Материалы межвузовского конкурса-конференции студентов, аспирантов и молодых ученых Северо-Запада. – СПб.: Изд-во Политехн. ун-та, 2006. – С.28-30.
4. Котляров В.П., Голубев А.А., Карпов А.Н. Автоматизация тестирования встроенных Java приложений с адаптацией к целевой платформе // Научно-Технические Ведомости СПбГТУ. – СПб.: Изд-во Политехнического Ун-та, 2006. – N5. – С.117-124.
5. Vsevolod P. Kotlyarov, Alexey A. Golubev, Andrey N. Karpov, “Testing Automation For J2ME Applications and API” // International Conference “Radio – That Connects Time. 110 Anniversary of Radio Invention”, 2005 / Proceedings of St.Petersburg IEEE Chapters, Volume II, Year 2005, p.98-103.
6. Vsevolod P. Kotlyarov, Alexey A. Golubev, Andrey N. Karpov, “Testing Automation for system core kJava applications” // 2006 IEEE Tenth International Symposium on Consumer Electronics (ISCE 2006) / Proceedings, p.596-599.