На правах рукописи

Хатько Евгений Евгеньевич

ИССЛЕДОВАНИЕ И РАЗРАБОТКА МЕТОДА, МОДЕЛЕЙ И АЛГОРИТМОВ ТЕСТИРОВАНИЯ ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Москва 2013

Работа выполнена в Московском физико-техническом институте (государственном университете) на кафедре «Микропроцессорные технологии».

Научный руководитель: Филиппов Владимир Александрович
кандидат технических наук, старший научный сотрудник, доцент кафедры «Микропроцессорные технологии», МФТИ ГУ

Официальные оппоненты: Саксонов Евгений Александрович
доктор технических наук, профессор кафедры «Вычислительные системы и сети», МИЭМ НИУ ВШЭ
Романчева Нина Ивановна
кандидат технических наук, доцент, декан факультета прикладной математики и вычислительной техники, МГТУ ГА

Ведущая организация: ОАО Научно-исследовательский институт «Аргон»

Защита диссертации состоится «19» июня 2013 г. в 15 час. на заседании диссертационного совета Д 409.009.01 при Институте электронных управляющих машин им. И. С. Брука по адресу: 119334, Москва, Вавилова, 24.

С диссертацией можно ознакомиться в библиотеке Института электронных управляющих машин им. И. С. Брука.

Автореферат разослан __ мая 2013 г.

Ученый секретарь диссертационного совета
кандидат технических наук, профессор Красовский Виктор Евгеньевич

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность исследования. В настоящее время мобильные технологии широко распространены в повседневной жизни. Практически у каждого человека на земле есть мобильный телефон. Мобильные устройства становятся сложнее и сложнее. Появились такие устройства, как смартфоны, коммуникаторы, планшетные компьютеры и др. Эти устройства по характеристикам не уступают вчерашним настольным компьютерам, они представляют собой достаточно сложные программно-аппаратные комплексы, управляемые операционными системами (ОС). До появления смартфонов телефон представлял собой простую систему, со встроенным программным обеспечением (ПО). Тестирование подобного ПО проводилось ручным способом. Дальнейшее ускоренное развитие мобильных технологий приводит к разрыву между сложностью применяемой методологии тестирования ПО для мобильных устройств и сложностью тестируемого ПО. На текущий момент, нельзя обойтись простым ручным тестированием по написанным тестовым сценариям. Требуется применение комплексного подхода, с использованием специальных средств автоматизации тестирования. Задача повышения эффективности методов тестирования приложений для мобильных устройств является важной с исследовательской точки зрения и срочной с практической точки зрения.

В работе рассматривается определенный класс приложений для мобильных устройств. Основные ограничения данного класса заключаются в следующем:

1. Приложения разрабатываются с использованием принципа отделения программного кода логики работы приложения от программного кода пользовательского интерфейса приложения (например, с использованием технологии Модель-Представление-Контроллер или Модель-Представление).
2. Каждое приложение разрабатывается под определенный, узкий круг задач, следовательно, число представлений (видов) приложения – конечное число порядка 100.
3. Для любого приложения существует конечный набор элементов пользовательского интерфейса, который позволяет полностью задать все виды приложения.

Далее термином ПМУ будет обозначаться любое приложение для мобильных устройств, соответствующее вышеописанными ограничениями.

Процесс разработки ПМУ происходит по итеративной схеме, её применение целесообразно в проектах, с продолжительностью до 1 года, при разработке приложений, обладающих пользовательским интерфейсом и ограниченным набором сценариев использования. Особенность данной схемы заключается в ускоренном продвижении к готовому продукту в результате нескольких итераций полного цикла разработки. Данная схема требует оптимизации встроенных процессов тестирования по времени.

Цель работы – повышение эффективности процессов тестирования ПМУ и автоматизация этих процессов на основе разработки соответствующего метода, моделей и алгоритмов.

Задачи исследования. Для достижения поставленной цели в работе были сформулированы следующие задачи:

1. Предложить метрику тестирования ПМУ, учитывающую их особенности и соответствующие критерии оценки эффективности методов тестирования. Проанализировать эффективность существующих методов тестирования ПМУ по времени и полноте тестового покрытия.

2. Предложить эффективный метод тестирования, основанный на генерации автоматизированных тестов из прототипов ПМУ, который позволит сократить общее время разработки и тестирования, обеспечивая при этом необходимое тестовое покрытие.
3. Разработать аналитическую и программную модели генерации автоматизированных тестов из прототипов ПМУ.
4. Предложить имитационно-статистическую модель тестирования, которая позволит оценить эффективность предложенного метода.

На защиту выносятся:

- Метод тестирования ПМУ, основанный на генерации автоматизированных тестов из прототипов ПМУ.
- Система тестирования ПМУ, имеющая в основе программную модель генерации тестов, которая позволит внедрить предложенный метод.
- Имитационно-статистическая модель тестирования ПМУ, имитирующая процесс тестирования с использованием предложенного метода, моделей и алгоритмов.

Научная новизна работы состоит в следующем:

- Предложена метрика тестирования ПМУ и критерий завершенности тестирования, основанные на особенностях разработки и использования ПМУ.
- Получен интегральный критерий оптимизации методов тестирования ПМУ, основанный на анализе и формальном представлении итеративной схемы разработки.
- Получен частный критерий эффективности процесса генерации тестовых сценариев.
- Предложен метод тестирования ПМУ с использованием расширенных конечных автоматов.
- Разработана аналитическая модель процесса генерации тестовых сценариев на основе алгоритма поиска кратчайшего пути на графе A^*.

Практическая значимость результатов диссертации состоит в разработке и внедрении системы тестирования ПМУ, состоящей из программной модели алгоритма генерации и двух модулей конвертации данных. Результаты работы использованы при построении процессов тестирования в компании «AT-Consulting». Предлагаемый метод тестирования применяется для тестирования мобильных и ВЕБ приложений в нескольких отделах компании в России и Казахстане.

Методы исследования. В диссертации проведено исследование процессов тестирования ПМУ с применением теории конечных автоматов, а также разработка алгоритмов генерации тестовых сценариев с применением теории графов. Также были применены принципы объектно-ориентированного программирования и элементы алгоритмической теории при разработке программной модели алгоритма генерации.

Апробация работы. Основные положения диссертационной работы доказывались и обсуждались на научно-практических конференциях и семинарах, в том числе:

- 52 научная конференция МФТИ – «Современные проблемы фундаментальных и прикладных наук», Москва 2009г.
- 53 научная конференция МФТИ – «Современные проблемы фундаментальных и прикладных наук», Москва 2010г.

- 54 научная конференция МФТИ – «Проблемы фундаментальных и прикладных, естественных и технических наук в современном информационном обществе», Москва 2011г.
- Конкурс в ЗАО «Интел А/О» - «Компьютерный континуум: от идеи до воплощения», Москва 2012г.

Публикации. По результатам выполненных исследований опубликовано 11 работ, в том числе 3 статьи в изданиях, рекомендованных ВАК и 2 статьи в зарубежном издании.

Структура диссертации. Диссертация состоит из введения, четырёх глав, заключения, приложений и списка литературы. Работа изложена на 113 стр., содержит иллюстрации.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы диссертации, сформулированы цель и задачи исследования. Область проводимого исследования представлена в табл. 1.

Область проводимого исследования **Таблица 1**

Способ разработки / **Этапы внедрения**	**Ручной**	**Автоматизированный**
Проектирование	Разработка спецификаций **Разработка моделей**	Генерация документов
Разработка	Имплементация приложения	Генерация кода для приложения
Тестирование	Разработка тестов Разработка автоматизированных скриптов Ручное тестирование	**Генерация тестов** **Автоматизированное выполнение тестов**

В **первой главе** проведен анализ итеративной схемы внедрения ПО, по результатам которого предложено 7 основных этапов разработки ПМУ: разработка бизнес требований, разработка высокоуровневой модели, разработка функциональных требований, имплементация, разработка тестов, автоматизация тестов, тестирование с получением результатов.

Схема разбиения процесса показана на рис. 1. Путь прохождения этапов 1-7 задает соответствующий метод разработки и тестирования. На основе представленной схемы методы классифицированы следующим образом:

- Ручная документация, ручное тестирование - РР метод (на рис. 1 выделен точечными стрелками).
- Разработка и тестирование на основе формальной модели - ФМ метод (на рис. 1 выделен штрихпунктирными стрелками).
- Ручная документация, автоматизированное тестирование - РА метод (метод с ручным выполнением всех этапов, кроме тестирования, выделен на рис. 1 аналогично РР методу).
- Генерируемая документация, ручное и автоматизированное тестирование - ПР метод (на рис. 1 выделен пунктирными стрелками).

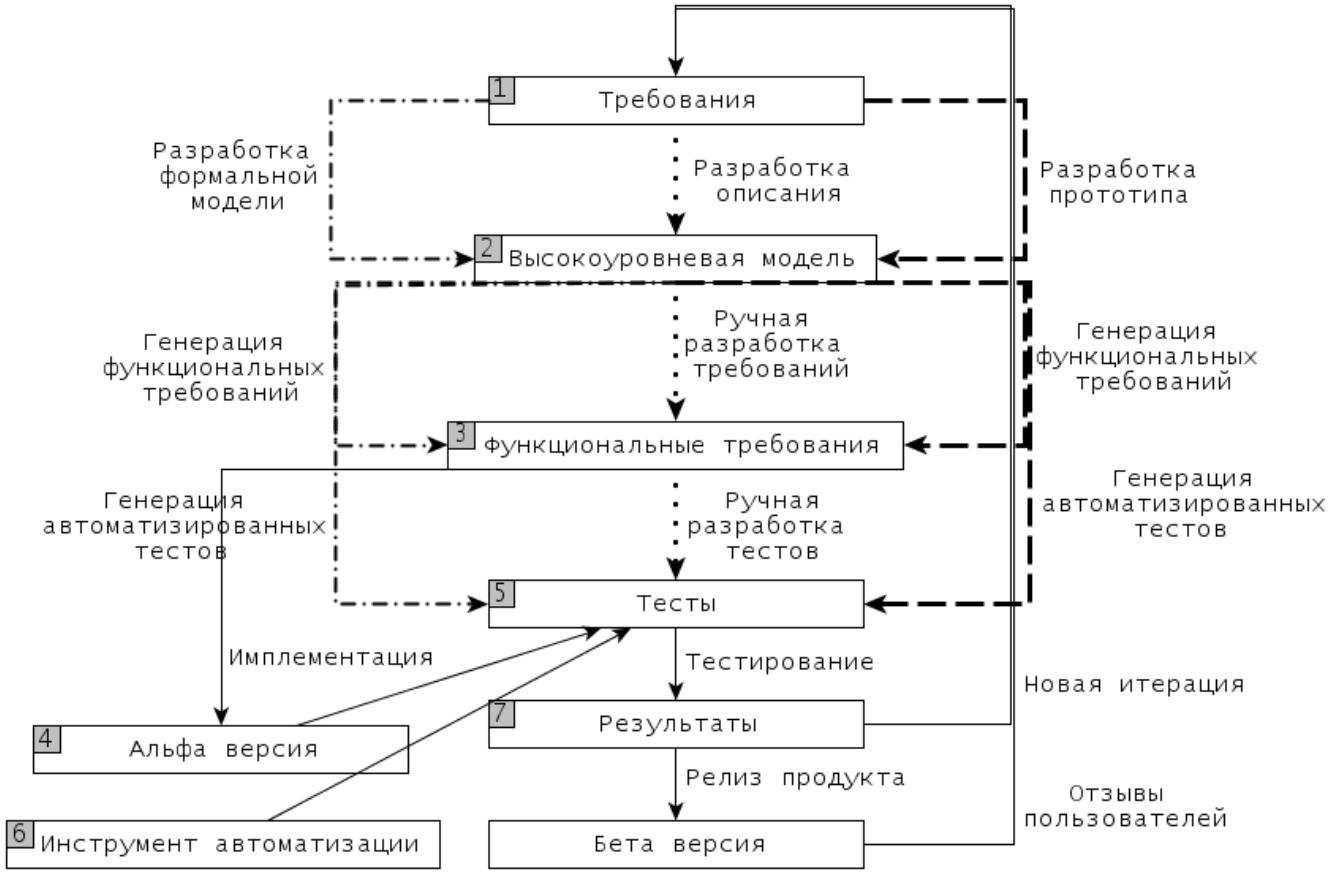

Рис. 1. Этапы разработки ПМУ

Далее предложена классификация средств автоматизации тестирования ПМУ на уровне пользовательского интерфейса на инструменты программной автоматизации, такие как UserEmulator (Symbian), DeviceAnywhere Automation™ for Smartphones, Robotium Framework и «playback» инструменты, например Run-on-Device (Jamo solutions).

Исходя из особенностей разработки и использования ПМУ, предложены метрика тестирования и критерий полноты тестирования.

Метрика тестирования ПМУ – процент проверенных откликов приложения на воздействия пользователя на элементы пользовательского интерфейса с учетом разбиения входных данных на классы эквивалентности, соответствующие пользовательским сценариям использования приложения.

Суть предложенной метрики состоит в том, что в соответствии с ней тестируются приложения на системном уровне, используя пользовательский интерфейс.

Критерий полноты тестирования – для обеспечения полного покрытия функционала достаточно проверить каждый отклик приложения после воздействия на каждый элемент пользовательского интерфейса, вводя данные, соответствующие пользовательским сценариям использования приложения.

Предложен следующий интегральный критерий эффективности методов тестирования в контексте итеративной схемы разработки:

$$T = \frac{\sum_{ij \in [12,23,34,35,45,65,57]} t_{ij}}{N} \rightarrow \min \quad (1),$$ где

- t_{ij} - время, потраченное на одном из этапов 1-2, 2-3, 3-4, 3-5, 4-5, 6-5, 5-7 (см. Рис. 1) в зависимости от выбранного способа прохождения этапа.
- N - покрытие откликов приложения, которое требуется обеспечить.

Исходя из результатов анализа предложенного интегрального критерия, сформулировано предположение об эффективности прототипного метода (ПР метода) разработки и тестирования ПМУ. Под прототипом в работе понимается упрощенная модель приложения, которая определяет его пользовательский интерфейс и описывает процесс взаимодействия пользователя с приложением. На формальном языке прототип ПМУ – это расширенный конечный автомат (РКА), каждое состояние которого соответствует одному виду приложения, а переходы между состояниями соответствуют действиям пользователя на элементы интерфейса. Каждое состояние содержит информацию о наборе элементов интерфейса соответствующего представления ПМУ. Каждый переход содержит информацию о типе соответствующего элемента интерфейса, о вводимых пользователем данных (если элемент интерфейса предполагает ввод данных), о способе проверки результатов выполнения действия. В основе применения предлагаемого метода лежит процесс генерации тестовых сценариев из прототипов ПМУ. Сгенерированные тестовые сценарии могут применяться для ручного или автоматизированного тестирования, с использованием программного подхода автоматизации. Предложенные критерии эффективности модели генерации тестов, представлены в табл. 2.

По результатам 1 главы, сделаны следующие выводы:

- Эффективность процесса тестирования ПМУ напрямую зависит от числа итераций разработки, поэтому оптимизация процессов тестирования по времени сводится к автоматизации процессов создания тестовых сценариев и автоматизации их выполнения.
- Наибольшую эффективность, исходя из аналитической оценки, предоставляет метод тестирования, основанный на применении прототипов ПМУ для генерации тестов.
- Для внедрения ПР метода, необходимо иметь возможность генерировать тестовые сценарии таким образом, чтобы использовать их в программном методе автоматизации тестирования с наименьшими доработками.

Во **второй главе** предложен метод тестирования ПМУ, основанный на использовании прототипа ПМУ для генерации автоматизированных тестов.

В контексте тестирования ПМУ на системном уровне, взаимодействие конечного пользователя с приложением предложено разбить на следующие типы.

Взаимодействие с элементом пользовательского интерфейса, который не предусматривает ввод данных (детерминированный элемент). В дальнейшем будем называть такой тип взаимодействия – «детерминированное действие» (ДД). Множество различных ДД напрямую зависит от разнообразия типов элементов интерфейса ПМУ и способов действия на них и является конечным числом.

Взаимодействие с элементом пользовательского интерфейса, который предусматривает ввод данных (вариационный элемент). В общем случае пользователь может ввести в приложение любой набор данных, это предполагает бесконечное число подобных действий ввода данных. Любое действие, связанное с вводом данных будем обозначать типом «вариационное действие» (ВД).

Критерии эффективности модели генерации тестов **Таблица 2**

Критерий оценки	Способ оценки
Процент покрытия переходов РКА	Усреднение обеспечиваемого алгоритмом покрытия переходов по различным РКА прототипов приложений, подаваемых на вход.
Средняя эффективность алгоритма генерации	$Eff(G) = \frac{N_e}{N \times L} 100\%$ G - граф прототипа ПМУ $N \times L$ - общее количество шагов $N_e = \sum_{s_i \in S} \max(s_i^{in}, s_i^{out})$ - минимальное и достаточное число переходов, в соответствии с теоремой Эйлера, которые нужно совершить для покрытия всех переходов РКА
Средняя скорость работы алгоритма	Скорость работы в секундах в зависимости от подаваемых на вход графов приложений, подвергающихся тестированию.
Автоматическая проверка набора элементов интерфейса текущего состояния	Проверка текущего состояния тестируемого приложение на соответствие текущему состоянию РКА.
Возможность контекстного хранения данных во время генерации	Возможность работы с динамическими данными, которые изменяются в процессе генерации.
Поддержка классов эквивалентности входных данных	Возможность учитывать классы эквивалентности входных данных в процессе генерации.
Поддержка инструментов прототипирования	Возможность работать с различными инструментами прототипирования.

Определив конечный набор детерминированных действий, и ограничив бесконечный набор вариационных действий конечным набором, можно свести взаимодействие пользователя с ПМУ к детерминированному, конечному набору операций. Конечный набор детерминированных действий представлен в табл. 3.

Набор детерминированных действий пользователя **Таблица 3**

идентиф. Действия	тип элемента пользовательского интерфейса	действие	описание
1	кнопка, активная область на экране	press (identificator)	нажатие на кнопку (активную область)
2	кнопка, активная область на экране	long_press (identificator)	долгое нажатие для вызова контекстного меню
3	выпадающий список элементов	select (identificator)	выбор элемента в выпадающем списке
4	флаг	check(identificator, 0|1)	переключение флага в состояния 0|1
5	переключатель	check(identificator, 0|1)	переключение переключателя в состояния 0|1
6	элемент ввода данных	input(identificator, data)	ввести в данный элемент предлагаемые данные
7	вид приложения	swipe_left	перенос экрана влево
8	вид приложения	swipe_right	перенос экрана вправо
9	вид приложения	scroll_up_to (identificator)	перемещение вверх с использованием скролла до элемента – element
10	вид приложения	scroll_down_to (identificator)	перемещение вниз, до элемента – element
11	вид приложения	zoom (value)	изменение масштаба
12	регулятор громкости	volume_up	увеличение громкости
13	регулятор громкости	volume_down	уменьшение громкости
14	кнопка выключения	Power	нажатие на кнопку отключения питания
15	кнопка выключения	long_power	долгое нажатие на кнопку отключения питания
16	разъём данных	cable (in|out)	подключение/отключение кабеля данных или питания
17	вид приложения	element_exists(type, text)	проверка наличия элемента данного типа с данным текстом на экране (включая невидимую область)
18	вид приложения	element_present(type,	проверка наличия

		text)	элемента данного типа с данным текстом в видимой части экрана
19	вид приложения	text_exists(text)	проверка наличия текста на экране (включая невидимую область)
20	вид приложения	text_present(text)	проверка наличия текста в видимой части экрана

Следует отметить, что данный набор может быть расширен дополнительными действиями, характерными для конкретной операционной системы или конкретного проекта внедрения ПМУ.

Проблему ограничения набора вариационных действий предложено решать методом эквивалентного разбиения входных данных. Пример эквивалентного разбиения представлен в табл. 4. Дополнительно введено понятие «контрольного значения» класса эквивалентности, соответствующее одному элементу данного класса.

Пример эквивалентного разбиения данных **Таблица 4**

тип элемента	**предполагаемый тип вводимых данных**	**классы эквивалентности**	**число классов эквивалент-ности**	**контрольные значения**
область ввода	$N \in [1,2,..,10] \subset Z$	$N \in [1,..,10]$ $N \in Z - [1,..,10]$ $N \notin Z$ $N \neq число$	4	5 0 5.5 ‘abcde’
область ввода	s - любая строка	$S - любая\_строка$ $S \neq строка$	2	‘abcde’ 0
список	выпадающий список значений	в зависимости от логики работы списка – каждый элемент списка либо один (любой) элемент списка	1 или n – число элементов списка	один элемент или каждый элемент списка
флаг	отмечен / не отмечен	0|1	2	отмечен неотмечен
Переключатель	отмечен / не отмечен	0|1	2	отмечен неотмечен

Определен способ построения РКА на базе прототипа ПМУ: чтобы построить РКА для данного ПМУ, нужно ввести правила построения прототипа будущего приложения, которые заключаются в декомпозиции пользовательского интерфейса будущего приложения, как показано на Рис. 2:

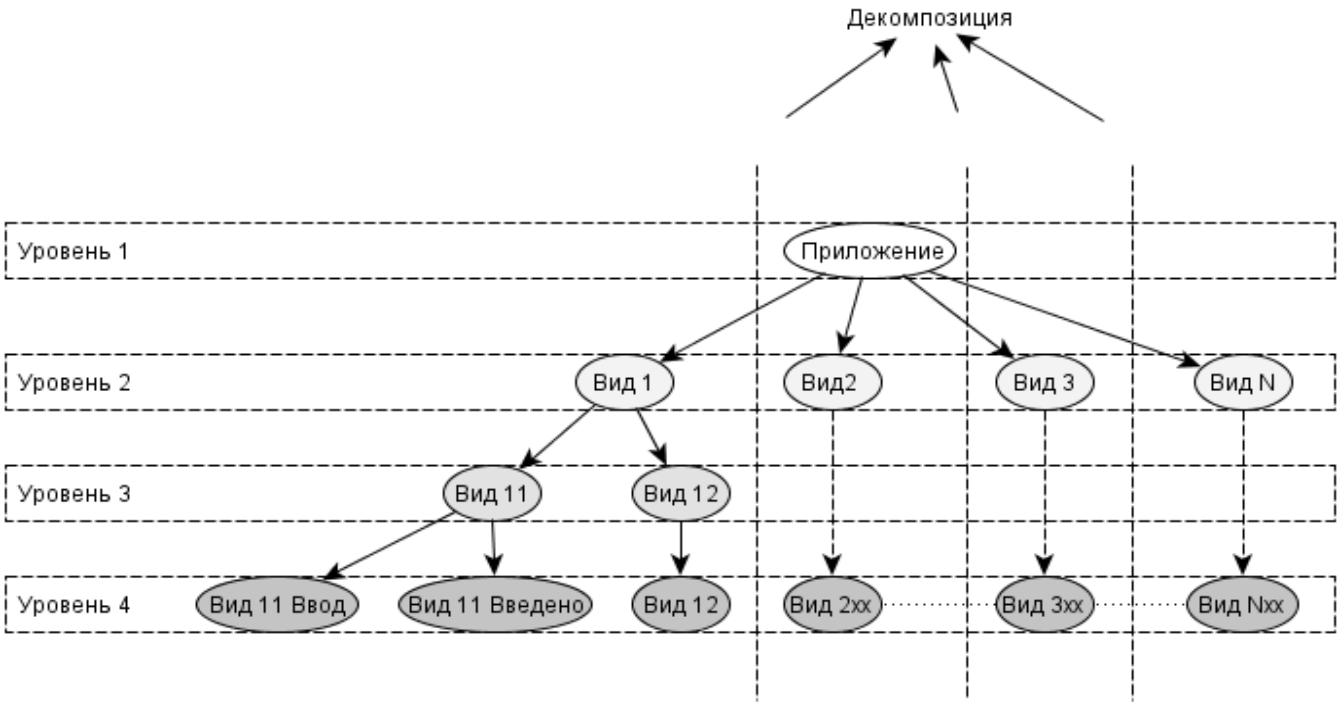

Рис. 2. Декомпозиция пользовательского интерфейса приложения

Пояснения к рис. 2:

1. Первый уровень - уровень приложения. Абстрактный уровень, представляющий приложение как совокупность видов пользовательского интерфейса.
2. Второй уровень - уровень видов. Приложение разбивается на виды, соответствующие различным состояниям его пользовательского интерфейса. Переход между видами осуществляется при помощи детерминированных действий пользователя.
3. Третий уровень – уровень подвидов. У каждого вида могут быть подвиды. Подвид соответствует любому всплывающему диалогу приложения в этом же виде.
4. Четвертый уровень – уровень классов эквивалентности. На этом уровне виды или подвиды предыдущих уровней, в которых возможен ввод данных разбиваются на 2 состояния - ВидВвод и ВидВведено. ВидВвод соответствует состоянию приложения, когда ввод данных ещё не был совершен. ВидВведено соответствует состоянию приложения, в котором пользователь уже ввел некоторый набор данных. Переходы между ВидВвод и ВидВведено соответствуют вариационным действиям пользователя.

Формальное определение РКА прототипа ПМУ задается следующим выражением:

$$PKA = \left\langle \begin{array}{l} S = \bigcup_{i=1..N} V_i^4, W = \bigcup_{i=1..N} \left[\bigcup_{e \in V_i} eq(e) \right], \\ \forall i \in [1..N], \forall e \in V_i \left\langle p_{ij} \xrightarrow{P} eq(e) \right\rangle, \\ S_0, \forall j : p_{1j} = \varnothing, \\ I = \bigcup_{i=1}^{N} \left[\bigcup_{e \in D_i} action(e) \cup \bigcup input(\prod_{e \in V_i} eq(e)) \right], \\ n_I \in [1,2], O = \bigcup_{i=1}^{N} (D_i \cup V_i), n_O = 2 \end{array} \right\rangle$$

Пояснения к формуле:

- $eq(e)$ - отображение, которое определяет набор контрольных значений классов эквивалентности для элемента пользовательского интерфейса e.
- $action(e)$ - отображение элементов интерфейса на действия, которые можно совершить с данными элементами.
- $input(\prod_{i=1}^{l} eq(e_i))$ - действие, связанное с вводом одного набора из классов эквивалентности всех элементов интерфейса текущего вида.
- $S = \bigcup_{i=1..N} V_i^4$ - состояния РКА: набор всех видов четвертого уровня.
- $W = \bigcup_{i=1..N} \left[\bigcup_{e \in V_i} eq(e) \right]$ - множество значений параметров: объединение всех контрольных значений классов эквивалентности входных данных по всем вариационным элементам прототипа.
- $\forall i \in [1..N], \forall e \in V_i \left\langle p_{ij} \xrightarrow{P} eq(e) \right\rangle$ - отображение, задающее параметры РКА, т.о. чтобы каждый параметр соответствовал одному вариационному элементу интерфейса, а значение параметра соответствовало контрольному значению класса эквивалентности этого элемента.
- $S_0, \forall j : p_{1j} = \varnothing$ - начальное состояние РКА и пустые начальные значения параметров.

- $I = \bigcup_{i=1}^{N} \left[\bigcup_{e \in D_i} action(e) \cup \bigcup input(\prod_{e \in V_i} eq(e)) \right]$ - набор стимулов РКА: объединение всех возможных действий пользователя с детерминированными и вариационными элементами интерфейса.
- $n_I = 1$ - число параметров стимулов: либо идентификатор детерминированного элемента, либо вводимые данные вариационного элемента.
- X – множество значений параметров стимулов: значение идентификатора детерминированного элемента, либо значение вводимых данных вариационного элемента.
- $O = \bigcup_{i=1}^{N} (D_i \cup V_i)$ - выходной алфавит РКА: множество всех элементов пользовательского интерфейса.
- $n_O = 2$ - число параметров выходного алфавита: каждый элемент интерфейса определяется по типу и содержанию. Эти параметры позволят определять корректность переходов по набору элементов в результирующем виде. Таким образом задавать ожидаемые результаты теста при генерации.
- Y - множество значений параметров выходного алфавита: множество пар (тип элемента, содержание элемента).
- T - множество переходов РКА: задается структурой пользовательского интерфейса приложения. Переход из одного состояния в другое соответствует некоторому единичному взаимодействию пользователя с приложением.

Предлагаемый в работе метод тестирования ПМУ имеет следующий алгоритм:

1. Построить прототип приложения на этапе проектирования по следующим правилам:
   a. Провести декомпозицию пользовательского интерфейса будущего приложения.
   b. Внести в прототип информацию о детерминированных и вариационных действиях пользователя, используя функции инструмента прототипирования.
2. Автоматически получить РКА прототипа, используя инструмент конвертирования прототипа.
3. Реализовать входной и выходной алфавиты РКА на программном уровне, используя программные средства инструмента автоматизированного тестирования.
4. Сгенерировать тестовые сценарии, с использованием предложенной модели генерации путем обхода графа полученного РКА.
5. Провести тестирование с использованием программной реализации входного и выходного алфавитов РКА на тестируемом ПМУ с помощью сгенерированных тестов.

Следует отметить, что прототип используется в качестве основного описательного источника в ходе разработки приложения, поэтому сгенерированные тестовые сценарии позволяют проверять функционал разрабатываемого приложения, который разрабатывается также как и сценарии, на основе построенного прототипа.

Предложенный метод позволяет свести процесс написания тестов для ПМУ к их генерации, и минимизировать процесс ручного тестирования, поскольку сгенерированные тесты являются автоматизированными. Акцент работы отдела тестирования, таким образом, смещается в область поддержки и оптимизации инструмента генерации тестов и программного представления входного и выходного алфавитов РКА. В 4 главе показано, что подобный подход позволяет оптимизировать процесс разработки и тестирования ПМУ по времени. Разработанный же в 3 главе алгоритм генерации позволяет достичь полного тестового покрытия ПМУ в соответствии с определенной метрикой тестирования.

На Рис. 3 приведена общая схема предложенного метода тестирования. На схеме более жирно выделены блоки, составляющие предлагаемую в работе систему тестирования, которая позволяет внедрить ПР метод.

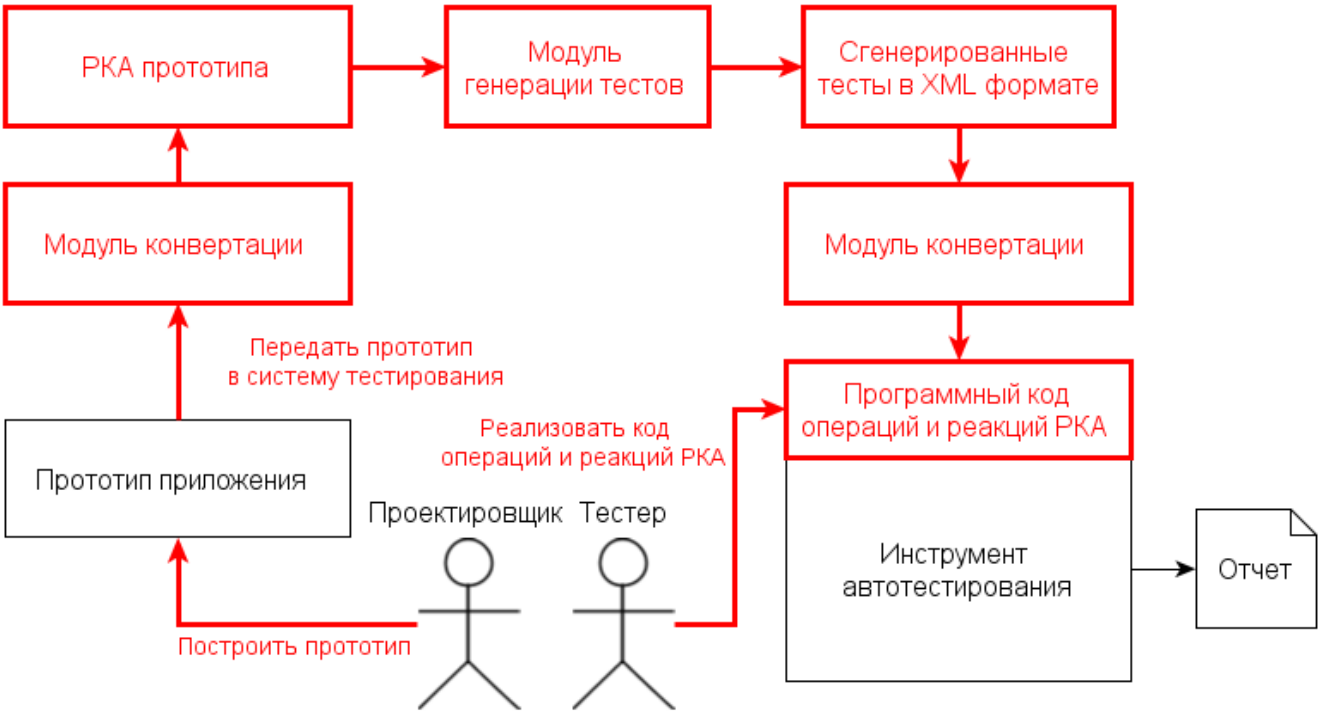

Рис. 3. Прототипный метод тестирования ПМУ

В **третьей главе** разработаны аналитическая и программная модели генерации тестовых сценариев.

В соответствии с разработанным критерием полноты тестирования ПМУ, требуется рассмотреть все возможные переходы РКА прототипа ПМУ. Известно, что конечные автоматы удобно представлять в виде графов. В таком представлении состояния РКА – это узлы графа, а переходы между состояниями – это ветви графа. Задача генерации тестового набора сводится к задаче обхода графа. При этом, в соответствии с предложенной метрикой тестирования, необходимо пройти по всем переходам графа, выбрав наикратчайший путь. Таким образом, сгенерируется минимальный набор тестовых сценариев, обеспечивающий полное тестовое покрытие. В случае существования Эйлерова цикла, рассматриваемая задача сводится к «задаче Китайского почтальона». Согласно теореме Эйлера в графе существует Эйлеров цикл тогда и только тогда, когда он сильно связан и для каждой вершины графа её полустепень захода равна

её полустепени исхода. Поэтому решить «задачу Китайского почтальона» для произвольного графа невозможно. Необходимо достичь максимального результата в её решении с точки зрения тестирования ПМУ: обойти все ветви графа с минимальным количеством повторений каждой ветви. Задача генерации тестовых сценариев хорошо исследована, но предлагаемые подходы во многом не учитывают специфики ПМУ. Рассмотрим основные алгоритмы обхода графов:

Алгоритм Дейкстры позволяет находить кратчайшие расстояния от одной из вершин графа до всех остальных. Данный алгоритм не подходит для решения поставленной задачи, поскольку в случае расширенного конечного автомата, веса ветвей могут динамически изменяться в зависимости от текущего контекста переменных КА. Аналогичная ситуация возникает с *алгоритмом Беллмана-Форда*.

Алгоритм Флойда-Уоршелла находит кратчайшие расстояния между всеми вершинами графа, его применение нецелесообразно, поскольку граф РКА в общем случае динамически изменяется после каждого следующего перехода, и пересчет путей нужно будет делать после каждого перехода.

Алгоритм $A*$ является алгоритмом поиска пути с наименьшей стоимостью. Алгоритм является «жадным» алгоритмом, работающим по первому совпадению. Помимо этого он использует эвристическую функцию для определения пути. В данной работе предлагается модификация $A*$, которая содержит информацию о параметрах графа и может «выбирать» маршрут с учетом условий переходов расширенного конечного автомата.

Функция стоимости F является основополагающей для работы алгоритма. Находясь в некотором узле графа, эта функция определяет какой переход выбрать следующим. Выбирается тот переход, для которого значение F наименьшее. В работе предлагается представление $F = G + H$, где:

G – суммарный вес всех покрытых на данный момент переходов (иными словами стоимость уже пройденного пути). На каждом шаге значение функции либо увеличивается на величину веса пройденного перехода, если этот переход ещё не покрыт (т.е. на данном шаге переход был пройден в первый раз), либо остается без изменения, если пройденный переход уже покрыт (т.е. был пройден ранее, в ходе работы алгоритма).

H – суммарный вес пути до ближайшего ещё непокрытого перехода (этот вес дает нижнюю оценку стоимости пути, который ещё предстоит пройти алгоритму). Путь до ближайшего непокрытого перехода ищется при помощи поиска в ширину (алгоритм BFS).

Во время работы алгоритма, перед каждым расчетом G и H веса всех переходов пересчитываются с учетом условий графа. Переходы, которые становятся недоступными из-за определенного условия, получают бесконечный вес, остальные переходы получают единичный вес.

Формально, пусть:

$T = [t_1, t_2, ..., t_n]$ - массив всех переходов РКА.

$C = [t_i, | t_i - \text{покрыт}]$ - массив всех покрытых на данный момент переходов. Массив изменяется в процессе работы алгоритма.

$W = [w_1, w_2, ..., w_n]$ - массив динамических весов, изменяющихся в процессе работы алгоритма. Веса меняются в соответствии с параметрами РКА.

$c_i = \begin{cases} 1, t_i \in C \\ 0, t_i \notin C \end{cases}$ - флаг, соответствующий покрытию i-го перехода.

P_i - путь из i-го состояния до ближайшего непокрытого перехода.

N - коэффициент, учитывающий приоритет условий: вначале выбираются непокрытые переходы, затем переходы, ближайшие к непокрытым ещё переходам.

Тогда функция стоимости алгоритма $A*$ задается следующей формулой:

$$F_i = (\|C\| + c_i) \times N + \sum_{P_i} w_i \quad (3)$$

Для внедрения аналитической модели, предложена программная модель процесса генерации со схемой, изображенной на рис. 4. Также в работе приведены схемы и алгоритмы работы основных модулей программной модели, обоснование выбора языка программирования для реализации модели. Рассмотрены основные входные и выходные данные. Предложена схема сервера визуализации работы модели.

Четвертая глава посвящена процессу построения имитационно-статистической модели тестирования по предложенному методу. Проведен анализ существующих мобильных операционных систем (ОС). Для проведения имитационного моделирования выбрана ОС Android как наиболее распространенная на данный момент. Предложен инструмент прототипирования ПМУ (Axure) на основе ранжирования по выбранным критериям.

Приведена схема модуля конвертации прототипа в формат входных данных программной модели алгоритма генерации. Данный модуль предоставляет техническую возможность перевести Axure прототип ПМУ в программное представление соответствующего РКА. Предложен инструмент автоматизации тестов Robotium – инструмент программной автоматизации тестирования приложений для операционной системы Anndroid, с помощью которого предложено реализовать операции и реакции РКА. Предложено описание модуля конвертации сгенерированных тестов в формат Robotium. Данный модуль переводит тесты, сгенерированные в xml формате в программный код автоматизированных тестов. Предложена конфигурация среды тестирования, со следующими основными требованиями:

- ПК с операционной системой Windows7, с установленными инструментами: IronRuby, AxureRP, Android SDK, средствами конвертации rp2xml, xml2test, Eclipse, Robotium.
- Тестируемое приложение установлено на мобильное устройство с операционной системой Android.
- Мобильное устройство подключено к ПК с помощью USB кабеля данных в режиме Android debugging (ADB).
- Известен адрес сервиса генерации тестов.

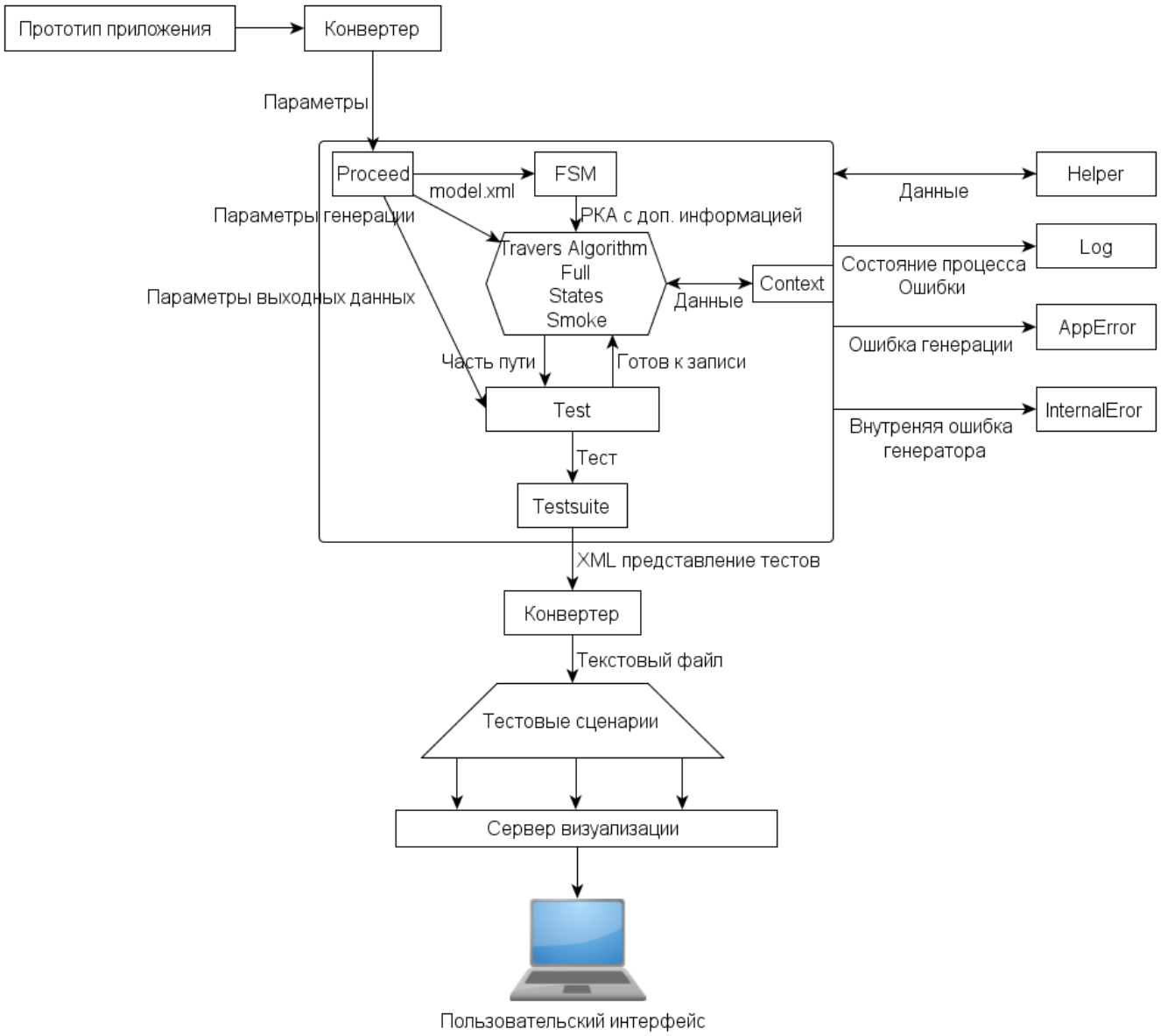

Рис. 4. Схема программной модели процесса генерации

В результате предложенной конфигурации, построена имитационно-статистическая модель тестирования, схема которой изображена на рис. 5.

Далее рассмотрена эффективность предложенного метода тестирования ПМУ и алгоритма генерации тестовых сценариев.

Основываясь на предложенных критериях оценки эффективности моделей генерации тестовых сценариев, рассмотрена эффективность предложенной модели. Для проведения оценки выбран генератор-оппонент (http://mbt.tigris.org/). Выбор именно этого генератора обусловлен следующими его характеристиками:

- Принимает на вход данные в формате graphml инструмента yEd, который позволяет задавать описания пользовательских интерфейсов приложений.
- Использует различные алгоритмы генерации, в том числе алгоритм $A*$.

Также были соответствующим образом подобраны тестовые данные в виде различных РКА. Все тестовые РКА были построены на основе одного РКА, который описывает

простейший вариант взаимодействия пользователя с приложением. Подобный РКА в работе назван «ядро тестового РКА», а построенные на его основе сложные РКА отличаются друг от друга только числом ядер. Для оценки эффективности процесса генерации были выбраны тестовые РКА с различными значениями числа ядер. В табл. 5 приведены результаты эффективности предложенной модели генерации в сравнении с генератором-оппонентом.

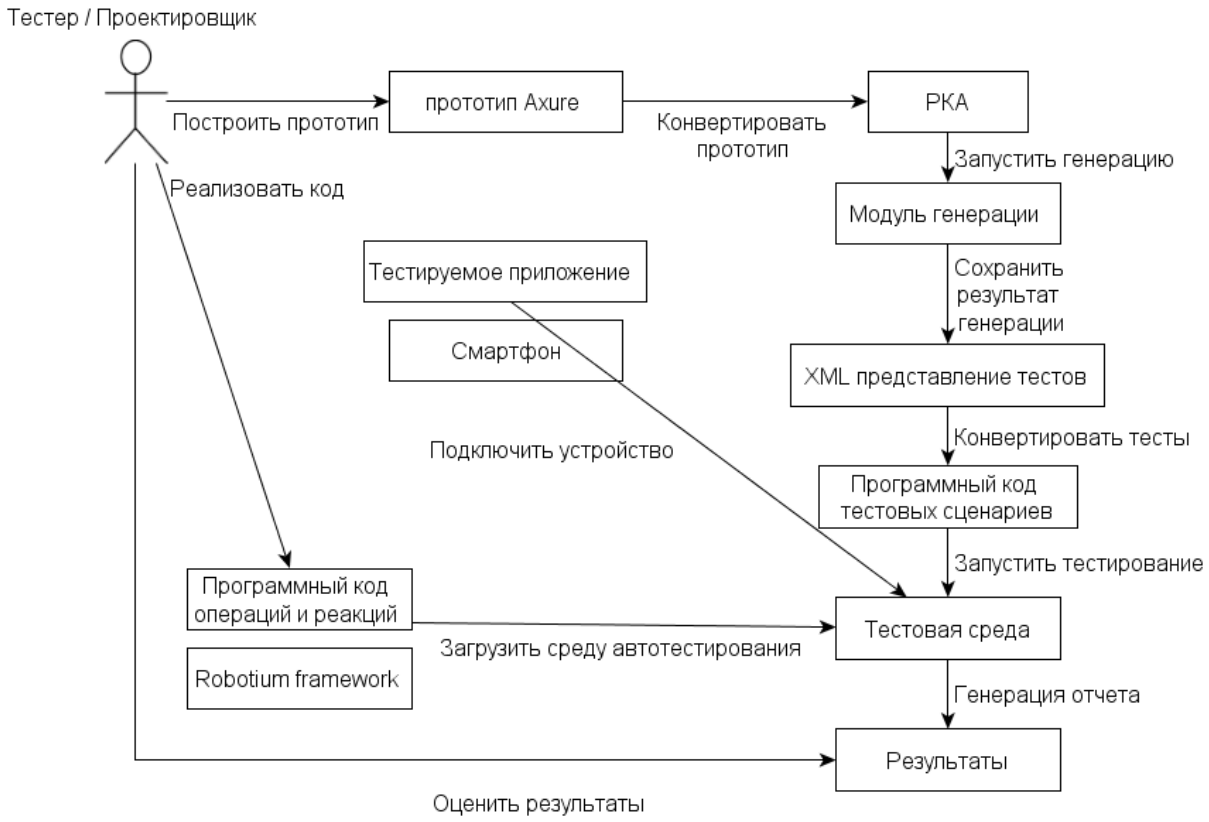

Рис. 5. Схема имитационно-статистической модели процесса тестирования ПМУ

Эффективность предложенной модели генерации **Таблица 5**

Критерий оценки	**Результаты оценки**	
	Генератор-оппонент	**Предложенная модель генерации**
Процент покрытия переходов РКА	95%	100%
Средняя эффективность алгоритма генерации	79%	77%
Средняя скорость работы алгоритма	294 сек/приложение	0.18 сек/приложение
Автоматическая проверка набора элементов интерфейса текущего состояния	нет	да
Возможность контекстного хранения данных во время генерации	да	да
Поддержка классов эквивалентности входных данных	нет	да
Поддержка инструментов прототипирования	yEd	yEd, AxurRP

Из представленной таблицы следует, что предложенная модель генерации превосходит генератор оппонент по совокупности выбранных критериев оценки. Однако основная положительная особенность предложенной модели состоит в возможности её применения в рамках предложенного метода тестирования. В частности в поддержке инструмента прототипирования AxureRP и возможности задавать классы эквивалентности входных данных.

Далее проведена оценка эффективности предложенного метода тестирования. Обозначим методы тестирования, следующим образом:

- $PA(\text{программный})$ – метод программного подхода автоматизации (РР метод с использованием программного подхода к автоматизации).
- $PA(\text{playback})$ – метод playback подхода автоматизации (РР метод с использованием «playback» подхода к автоматизации).
- $ПР$ – прототипный подход, аналитическая оценка (ПР метод).
- $ФМ$ – подход с использованием формальной модели (ФМ метод).
- $ПР(\text{фактич.})$ – прототипный подход, фактический результат (ПР метод).

Тогда, основываясь на различных допущениях относительно времени выполнения каждого шага итерационной схемы (рис. 2), не критичных для проведения оценки, интегральный критерий эффективности тестирования (1) сводится к следующим выражениям:

$$\begin{cases} Eff_{PA(\text{программный})}(K) = \dfrac{2.3K^2 + 5K}{0.8K^2 + 4.5K + 10} \\ Eff_{PA(\text{playback})}(K) = \dfrac{2.3K + 5}{K + 5.5} \\ Eff_{ПР}(K) = \dfrac{2.3K + 5}{0.8K + 3.5} \\ Eff_{ФМ}(K) = \dfrac{2.3K + 5}{1.6K + 11.5} \\ Eff_{ПР(\text{фактич.})}(K) = \dfrac{2.3K^3 + 5K^2}{0.8K^3 + 3.5K^2 + 0.3K + 3} \end{cases}$$

, где K – число итераций разработки

На рис. 6 представлены графики зависимости эффективности от числа итераций полного цикла разработки.

Следует отметить, что на целевых значениях $K \in [5..20]$ фактический результат эффективности выше теоретических результатов для других методов, но не достигает теоретической оценки из-за увеличенных затрат, связанных с реализацией программного кода детерминированных и вариационных действий. Увеличение эффективности относительно метода РР составляет в среднем 100% в зависимости от числа итераций полного цикла разработки.

Увеличение эффективности относительно метода РА (программный подход) составляет в среднем 30%.

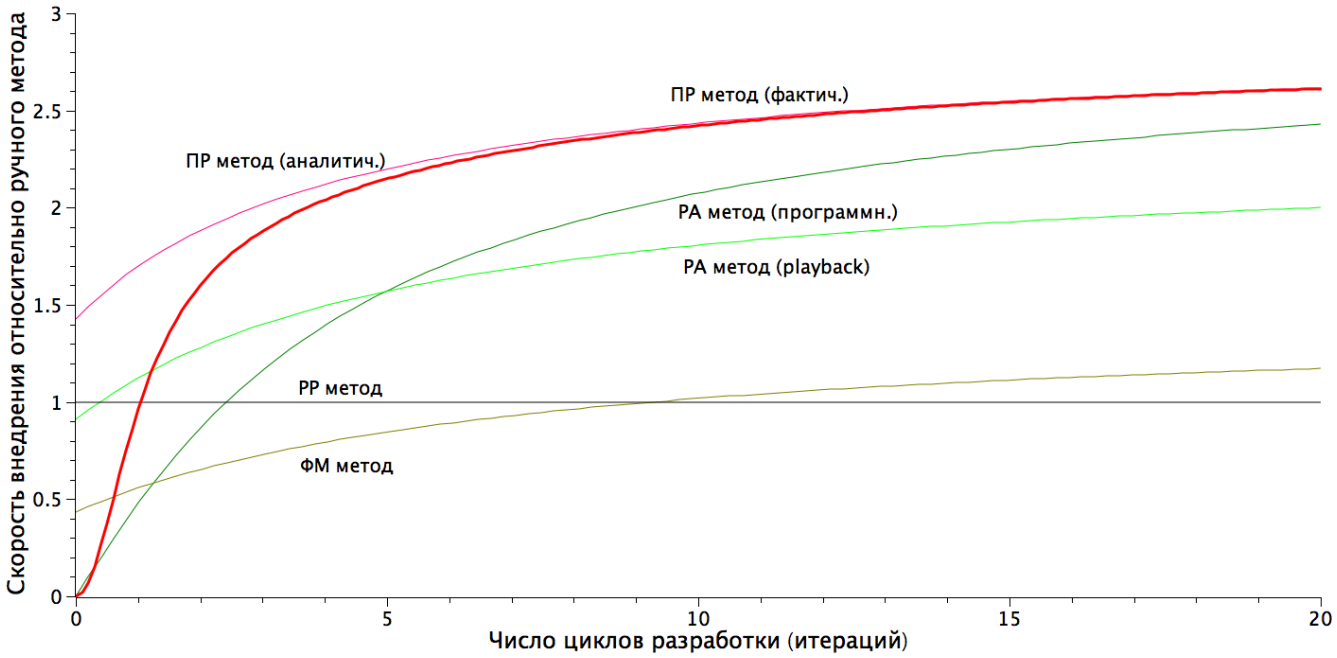

Рис. 6. Эффективность методов тестирования при фиксированном покрытии

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

Итоги проведенной работы приведены ниже:

1. Предложен новый, эффективный метод тестирования ПМУ, в рамках которого
   - Предложена метрика тестирования ПМУ и сформулирован соответствующий критерий полноты тестирования.
   - Предложена модель описания взаимодействия пользователя с ПМУ.
   - Разработан алгоритм построения прототипов ПМУ, который позволяет представлять мобильные приложения в виде расширенных конечных автоматов. Предложенный алгоритм позволяет учитывать классы эквивалентности входных тестовых данных.
2. Разработана система тестирования ПМУ, которая позволила внедрить предложенный метод. Система состоит из модуля конвертации прототипа приложения в программное представление РКА, модуля генерации тестовых сценариев и модуля конвертации сгенерированных тестов, представленных в XML формате в автоматизированные тесты, представленные при помощи операций и реакций РКА.
3. Разработана имитационно-статистическая модель тестирования ПМУ для ОС Android, с использованием среды автоматизации тестирования Robotium framework и инструмента прототипирования AxureRP, которая позволяет увеличить эффективность тестирования на 30%.
4. Результаты работы использованы при построении процессов тестирования в компании «AT-Consulting». Предлагаемый метод тестирования применяется для тестирования мобильных и ВЕБ приложений в нескольких отделах компании в России и Казахстане.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. **Хатько Е. Е., Филиппов В. А.** Проблемы качества тестирования программного обеспечения для мультизадачных пользовательских комплексов. // *Качество. Инновации. Образование.* 2011 г., Т. 3, стр. 32-35.
2. **Хатько Е. Е.** Об одном методе тестирования «мобильных» приложений. // *Труды МФТИ.* 2012 г., Т. 4, стр. 132-140.
3. **Хатько Е. Е., Филиппов В. А.** Алгоритмы генерации тестовых сценариев для повышения качества программного обеспечения многозадачных пользовательских комплексов. // *Качество. Инновации. Образование.* 2011 г., Т. 10, стр. 47-52.
4. **Fillipov V., Khatko E.** An analytical model of mobile applications' tests generation process. // *Software Engineering, USA.* 2012.Vol. 4, pp. 165-173.
5. **Khatko E., Fillipov V.** Mobile applications testing processes metrics and optimization criteria. // *Software Engineering, USA.* 2012. Vol. 4, pp. 174-179.
6. **Филиппов В. А., Хатько Е. Е.** Проблемные вопросы автоматизации тестирования для мультизадачных пользовательских комплексов. // *Сборник научных трудов МИЭМ.* 2012 г., стр. 81-85.
7. **Филиппов В. А., Хатько Е. Е.** Модели для мультизадачных пользовательских комплексов. // *Сборник научных трудов МИЭМ.* 2012 г., стр. 86-98.
8. **Филиппов В. А., Хатько Е. Е.** Генерация тестовых сценариев для ПМУ. // *Сборник научных трудов МИЭМ.* 2012 г., стр. 99-106.
9. **Хатько Е. Е.** Один из подходов к анализу системы тестирования сложных программных комплексов. // *52 Научная конференция МФТИ,* 2009 г., Т. 1, стр. 104-107.
10. **Хатько Е. Е.** Один способ реализации алгоритма генерации тестов в тестировании на основе моделей. // *53 Научная конференция МФТИ*, 2010 г., Т. 1, стр. 92-95.
11. **Хатько Е. Е.** Оптимизация процессов разработки пользовательских приложений с точки зрения тестирования. // *54 Научная конференция МФТИ*, 2011 г., Т. 1, стр. 84-85.

Хатько Евгений Евгеньевич
Исследование и разработка метода, моделей и алгоритмов тестирования приложений для мобильных устройств

Подписано в печать 06.05.2013г. Формат 60 × 84 1/16.
Усл. печ. л. 2, 2. Тираж 100 экз. Заказ №

Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Московский физико-технический институт (государственный университет)»

Отдел оперативной полиграфии «Физтех-полиграф»
141707, Моск. обл., г. Долгопрудный, Институтский пер., 9
E-mail: polygraph@mipt.ru