федеральное государственное автономное образовательное учреждение
ВЫСШЕГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет *Информационных Технологий и Программирования*
Направление (специальность) *Прикладная математика и информатика*
Квалификация (степень) *Бакалавр прикладной математики и информатики*
Кафедра *Компьютерных технологий* Группа *4539*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Разработка методики автоматического тестирования взаимодействий мобильных приложений с системой Android

Автор квалификационной работы *Баглай Богдан* (подпись)

Руководитель *Макаров Никита Сергеевич* (подпись)

Консультанты:

а) По экономике и организации производства _____ (подпись)

б) По безопасности жизнедеятельности и экологии _____ (подпись)

в) _____ (подпись)

К защите допустить

Зав. кафедрой *Васильев В.Н.* (подпись)

«___» _____ 2015 г.

Санкт-Петербург, 2015 г.

Дата защиты «____» _____ 2015 г.

Секретарь ГАК _____

Листов хранения _____

Чертежей хранения _____

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Основной целью данной работы является разработка подхода автоматического тестирования взаимодействий мобильных приложений с операционной системой Android. Под взаимодействием приложения с системой, имеются в виду любые действия пользователя в операционной системе, имеющие какое-либо влияние на работу приложения и наоборот. Любые действия пользователя в приложении, влияющие на работу операционной системы. Зачем такое тестирование необходимо?

В настоящее время широкое распространение получили приложения, взаимодействующие с системой. Так например, системные уведомления, без которых уже трудно представить современное мобильное приложение. «Список дел» напоминает нам выполнить запланированное важное дело, приложение «Холодильник», проскандировав ближайшие магазины, предлагает закупиться свежим молоком, а «Здоровье» рекомендует подышать. Кликнув по одному из таких уведомлений, мы перейдем в приложение уведомившее нас. В какую часть приложения мы попадем зависит от самого уведомления. Уведомления могут вести в разные части приложения, которых может быть не мало. А сами уведомления, создаются при определенных условиях: вышло обновление, вам написали сообщение, вам отправили файл. Логику работы таких уведомлений хочется полноценно тестировать перед релизом приложения. Но, большое количество всевозможных комбинаций уведомлений, и не малое время создания необходимых условий, очень сильно замедляет ручное тестирование данной функциональности приложения. На решение данной проблемы и нацелена эта работа.

Целью данной работы является обеспечить возможность покрытия автоматическими тестами взаимодействий приложения с системой Android.

Задачей данной работы является разработать новый подход к автоматизации тестирования взаимодействий приложения на Android в условиях реалий автоматического тестирования на предприятии с активной разработкой приложений для мобильных устройств.

На момент реализации, поставленной в данной работе, цели, в качестве традиционного автоматического тестирования мобильных приложений на Android использовалось два подхода. Эти подходы отличны друг от друга и они не взаимозаменяемы по функциональность. По каждому из направлений существуют инструменты автоматического тестированию и последователи развивающие их. Но, во многих случаях, в мобильном автоматическом тестировании есть необходимость в применении обоих способов. В виду такой

необходимости, существует инструмент - «обертка» над существующими проектами обоих подходов. По своей сути, это два независимых инструмента. А при запуске автоматического теста, в зависимости от необходимой функциональности, запускается один из инструментов. Большой минус такой «обертки» это отсутствие связи двух подходов на программном уровне. Кроме того, проекты входящие в состав данного решения разрабатываются разными командами, поддержание работоспособности автоматических тестов для двух инструментов несет дополнительные временные расходы и ведет к понижению общей стабильности, и надежности системы автотестирования.

В данной работе, в рамках реализации новой методики был сознан программный комплекс объединяющий оба подхода автоматического тестирования на программном уровне, и, как следствие, решающий проблемы традиционных подходов. Разработанный комплекс успешно прошел проверку в системе автоматизированного тестирования мобильных приложений под Android на предприятии. В настоящее время данная разработка обеспечивает функционирование автоматических тестов взаимодействующих с системой Android, а так же дала возможность покрыть автотестами более тридцати тесовых ситуаций основанных на взаимодействии с системой.

Первая глава работы посвящена обзору известных систем автоматического тестирования мобильных приложений на Android. Там же обсуждаются недостатки существующих систем автоматического тестирования мобильных приложений.

Во второй главе, на основании анализа различных систем автоматического тестирования мобильных приложений, а также опыта автоматического тестирования системных взаимодействий на предприятии, сформулированы основные принципы и требования к системе. Далее приведено общее описание и модель автоматической системы тестирования системных взаимодействий.

В третьей главе приведена архитектура созданного решения. Описано взаимодействия тестовой программы с сервером тестирующего инструмента и других частей системы автоматического тестирования.

В заключении дано описание текущего состояния разработки и перспективы её развития.

ГЛАВА 1. Обзор

1.1. Описание предметной области

В данном разделе подробно опишем цели работы, а также выведем термины, используемые в других частях представленной работы.

1.1.1. Термины и понятия

Введем термины и понятия касающиеся данной работы. В дальнейшем приведенные термины будут использоваться в указанных значениях, если не оговорено обратное.

Системное взаимодействие - любые действия пользователя в операционной системе, имеющие какое-либо влияние на работу приложения и наоборот, любые действия пользователя в приложении влияющие на работу операционной системы. Речь идет о обычном пользователе, который, в обоих случаях, взаимодействует с системой или приложением посредством пользовательского интерфейса (далее UI).

Шторка - была была

1.1.2. Цели работы

Как уже было упомянуто в ведении, целью данной работы является разработка методики позволяющей осуществлять тестирование взаимодействий мобильного приложения на Android. Для простоты понимания, приведем несколько примеров таких взаимодействий.

Самым простым взаимодействием пользователя с ОС, влияющим на работу приложения, может быть изменение текущего времени в настройках телефона. Приложение, состояние которого зависит от времени («Будильник», «Фазы сна» и т.д.) , должно будет корректно отреагировать на данное системное взаимодействие. Подобным образом, изменения языка в системе, может повлечет некоторые изменения в приложении.

Особый интерес представляют собой Push уведомления. В Android нет никакой встроенной системы для прямого отображения пользователю push-уведомлений. Все данные передаются («пушатся») исключительно в само приложение и в произвольной форме. После получения данных, приложение может, например, выдать стандартное для Android уведомление. Такое уведомление отобразится в верхней части экрана и в “шторке”. А может появится баннер, как в iOS. Учитывая открытость и гибкость ОС Android, после получения push-уведомления, оно может выводиться, практически в любой форме.

В виду гибкости и экономичности к ресурсам, данная технология получила широкое распространение в современных мобильных приложениях. Функциональность приложений все чаще задействует технологию push-уведомлений. А тестирование такой функциональности, связанной с push-уведомлениями, занимает много времени ручного тестировщика. В основном, из-за необходимости создавать определенные условия, при которых трестируемое push-уведомление генерируется. Но, также, из-за необходимости производить определенные манипуляции в UI, вне трестируемого приложения, что увеличивает вероятность ошибки тестировщика. В случае ошибки, все действия, проделанные для создания определенных условий, необходимо повторить заново. Учитывая что таких уведомлений может быть десятки в одном приложении, процесс ручного тестирования данной функциональности будет очень сильно тормозить цикл разработки приложения.

Далее в работе, из системных взаимодействий, по большей части, мы будем говорить о push-уведомления, ввиду их большего использования. Но рассмотренные методы и разработанное решение актуальны и применимы ко всем видам системных взаимодействий.

Уточним, что наша цель, не посто иметь возможность тестировать системные взаимодействия, но делать это в условиях процесса разработки реального мобильного приложения, нацеленного на широкий круг

пользователей. Это означает, что приложение должно корректно работать на давольно большом диапазоне версий Android. Что накладывает дополнительные требования к функциональности инструмента используемого в целях автотестирования. Данное ограничение сильно повлияло обзор существующих подходов автоматического тестирования системных взаимодействий.

Есть еще одно немаловажное требование к системе автоматического тестирования. Система должно базироваться на использовании одного инструмента автоматического тестирования. Данное ограничение необходимо для обеспечения стабильность системы автоматического тестирования в целом. Использование нескольких инструментов влечет за собой следующие сложности. Дополнительные трудозатраты на поддержание в работоспособном состоянии уже написаных автоматических тестов. Трудность связанные с переходами на новые версии используемых инструментов(может частично пропасть совместимость инструментов), так так разработка разных инструментов ведется разными командами. А как же, общая стабильность снижается, так как в нескольких инструментах потенциально больше багов чем в одном.

1.2. Анализ

В данном разделе описаны требования к инструменту автоматического тестирования приложений написанных под операционную систему Android, описаны существующие, на момент написания данной работы, инструменты, а так же, рассмотрены недостатки готовых решений.

1.2.1. Требования

Учитывая требования выдвинутые в части 1.1.2., составим список из наиболее важных требований к инструменту автоматического тестирования мобильных приложений. Стоит отметить, что все рассматриваемые инструменты удовлетворяют некоторому подмножеству требований, и потому, такие требования не будут включены в список требований. Перечислим наиболее показательные требования:

- Тестовое приложение не должно изменяться.
- Поддержка старых(< 4.2) версий Android.
- Поддержка тестирования системных взаимодействий.
- Система автотестирования должна базироваться на одном инструменте.
- Максимальное «переиспользование» уже существующей инфраструктуры автоматического тестирования web

1.2.2. Существующие решения

В виду большого количества инструментов автоматического тестирования для платформы Android, в обзор попали инструменты, максимально удовлетворяющие основным требованиям, описанным в части 1.2.1.

Все существующие решения, после отсеивания по первому требованию, делятся на два вида. Первый вид, использующие подход Instrumentation. Второй, использующие подход UIAutomator.

UIAutomator разрабатывается корпорацией Google и поставляется вместе с Android SDK. Android SDK предоставляет следующие инструменты для поддержки автоматизированного функционального тестирования пользовательского интерфейса: UIAutomatorviewer - графический инструмент для распознавания компонентов пользовательского интерфейса в Android приложении, UIAutomator - библиотеки Java API, содержащие методы для создания тестов пользовательского интерфейса.

Для использования UIAutomator-а, необходима версия Android SDK Tools 21 или выше, API 16 или выше. Из чего следует, что UIAutomator-ом можно тестировать только приложения, версия которых выше 4.1.

Instrumentation разрабатывается корпорацией Google и поставляется вместе с Android SDK. Данный способ был изначально, API level 1. Недостатком данного подхода является отсутствие возможности тестирования объектов вне трестируемого приложения. Это означает, что с помощью инструментов, использующих данный подход, нет возможности тестировать системные взаимодействия.

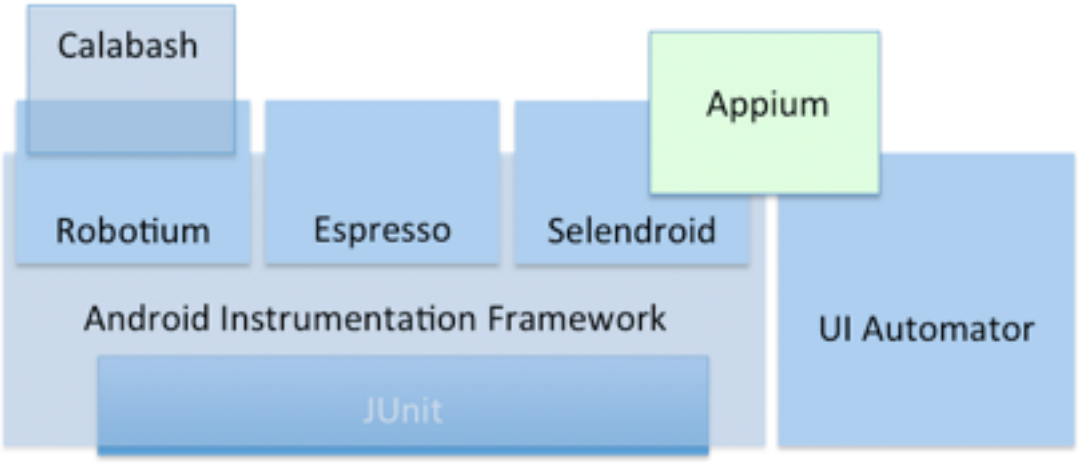

Рис. 2. Схема готовых решений и методов которые они используют.

1.2.2.1 Selendroid

Самым ярким представителем Instrumentation подхода является проект Selendroid. Данный проект активно разрабатывается компанией ebay inc. Поддерживает тестирование приложений начиная с версии API 10.

Selendroid работает с наивными приложениями и гибридными. Тесты пишутся с использованием клиентского API WebDriver. Работает с эмуляторами, реальными устройств и может быть интегрирован в качестве узла в Selenium Grid для масштабирования и параллельного тестирования.

На схеме (рис.1.) представлена архитектура проекта Selendroid.

Рис. 1. Схема архитектуры Selendroid

1.2.2.1 Appium

Проект Appium является наиболее интересным для нас инструментов. Само его существование подтверждает важность выдвинутых нами требований к инструменту автоматического тестирования.

Appium объединяет в себе оба подхода автоматического тестирования мобильных приложений под Android, что дает возможность тестировать как старые (<4.2) версии Android посредством Instrumentation подхода, так и системные взаимодействия с помощью UIAutomator-а. Такая функциональность нам и нужна.

Но не все так гладко с Appium. Как можно увидеть из схемы (Рис.2.), Appium использует оба подхода, Instrumentation и UIautomator. Но, «честно», они это делаю только в случае с UIAutomator походом. А вот по части Instrumentation подхода, Appium является оберткой над Selendroid-ом, которая переводит команды Selendroid WebDriver в Selendroid команды. В итоге, получаем, что для версий Android 4.2+ используется Google UiAutomator, а для версий Android 2.3+ используется Selendroid, работающий посредством Google Instrumentation.

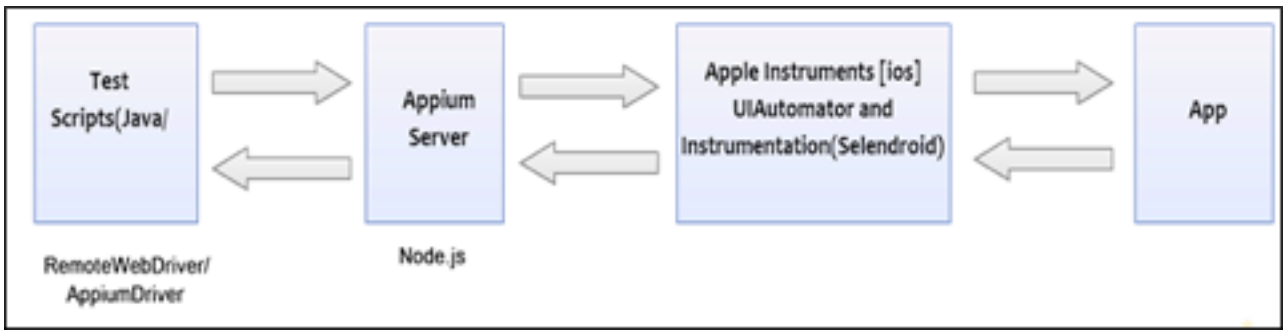

Рис.3. Схема составных частей Appium.

По сравнению с Selendroid-ом, Appium находится ближе к нашей цели и удовлетворяет большему количеству требований.

1.2.3. Недостатки готовых решений

Недостатком всех инструментов, использующих Instrumentation подход, в том числе и Selenium-a, является полное отсутствие возможности тестирования системных взаимодействий. Ароматическому тестирования подлежит только трестируемое приложение, доступ вне преложения отсутствует.

Недостатком

1.3. Постановка задачи

ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ РЕЗУЛЬТАТЫ

В данной главе рассматриваются теоретические аспекты создания автоматической системы тестирования. Приведена общая схема тестирующего инструмента.

ГЛАВА 3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

В предыдущих главах были подробно описаны концепции положенные в основу системы автоматического тестирования мобильных приложений. В этой главе описаны технические подробности реализации данной системы.

Заключение

СПИСОК ЛИТЕРАТУРЫ

1. Software Test Automation Paperback use pre formatted date that complies with legal requirement from media matrix, by Mark Fewster, Dorothy Graham – September 4, 1999.
2. Selenium 2 Testing Tools: Beginner's Guide Paperback use pre formatted date that complies with legal requirement from media matrix, by Burns David – October 19, 2012.

Ресурсы глобальной сети Интернет

4. http://automated-testing.info/ — информационный сайт посвященный автоматизации тестированя
5. http://selendroid.io — сайт инструмента автоматизации тестирования для Android — selendroid
6. http://selenium2.ru — сайт инструмента автоматизации тестирования — selenium