ОБЗОР ОСОБЕННОСТЕЙ РАЗРАБОТКИ И ТЕСТИРОВАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Якимов Виктор Игоревич

магистрант, СамГТУ, РФ, г. Самара

Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах.

В настоящее время, используются следующие мобильные операционные системы:

· Android, операционная система для смартфонов, планшетов, электронных книг и других устройств. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android, Inc., которую затем купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы.>

· IOS, мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей и некоторых других устройств, разрабатываемая и выпускаемая американской компанией Apple. Была выпущена в 2007 году; первоначально — для iPhone и iPod touch, позже — для таких устройств, как iPad и Apple TV. В 2014 году появилась поддержка автомобильных мультимедийных систем Apple CarPlay. В отличие от Windows Phone (Microsoft) и Android (Google), выпускается только для устройств, производимых фирмой Apple.

В iOS используется ядро XNU, основанное на микроядре Mach и содержащее программный код, разработанный компанией Apple, а также код из OCNeXTSTEP и FreeBSD. Ядро iOS почти идентично ядру настольной операционной системы Apple macOS (ранее называвшейся OS X).

· Sailfish OS, операционная система, основанная на проектах с открытым исходным кодом и включающая компоненты с закрытым исходным кодом. Sailfish OS развивается с 2012 года финской компанией Jolla. В 2016 году к Jolla присоединилась российская компания «Открытая мобильная платформа». Android-приложения могут устанавливать только владельцы официальных устройств.

· Windows 10 Mobile, версия ОС Windows 10, предназначенная для мобильных устройств с диагональю экрана до девяти дюймов. Призвана обеспечить бóльшую синхронизацию с версией Windows для персональных компьютеров более широкой синхронизацией контента, новыми «универсальными» приложениями, а также возможностью подключения устройств к внешнему дисплею и использовать смартфон в качестве ПК с интерфейсом с поддержкой мыши и клавиатуры.

· BlackBerry OS, операционная система с основным набором приложений для смартфонов и коммуникаторов, выпускаемых компанией Research In MotionLimited (RIM)

· Fire OS, базирующаяся на Linux kernel операционная система, созданная Amazon для собственных смартфонов Fire Phone, электронных книг Kindle Fire, планшетов, иных устройств типа the Fire TV. **Fire OS** является форком Android. Основной акцент в **Fire OS** сделан на потребление контента, кастомизированом пользовательском интерфейсе и привязке к сервисам Amazon.

Якимов В.И. Обзор особенностей разработки и тестирования мобильных приложений // Студенческий форум: электрон. научн. журн. 2018. № 11(32). URL: https://nauchforum.ru/journal/stud/32/37412 (дата обращения: 10.09.2025).

В разных ОС используются разные языки программирования\инструменты для написания приложений. К примеру, для IOS наиболее распространенными на данный момент являются следующие языки (фреймворки):

1) Swift – язык, разработанный компанией Apple и предназначенный для приложений под iOS и OS X. Он заимствовал довольно многоц из C++ и Objective-C;

2) JavaScript — прототипно-ориентированный сценарный язык, который наиболее широко применяется в кроссплатформенных фреймворках (React Native, Ionic, Sencha и т.п.);

3) C# — объектно-ориентированный язык, разработанный в 1998-2001 годах группой инженеров в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework. В области мобильных приложений и используется во фреймворке Xamarin.

4) Objective-C – компилируемый объектно-ориентированный язык корпорации Apple. Его заменяет новый и более простой Swift.

Для разработки и отладки мобильных приложений обычно используются специальные инструменты интегрированной среды разработки (IDE), часто предоставляемые разработчиками ОС. Для наиболее распространенных ОС, это:

· **Android Studio —** это IDE для работы с платформой Android, которая включает в себя: текстовый редактор, компилятор и/или интерпретатор, средства автоматизации сборки, отладчик.

· **Xcode** - IDE для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple. Совместно с ней, поставляется набор симуляторов для запуска и отладки приложения на «реальном устройстве».

Существует ряд требований, предъявляемых к инструментам автоматизированного тестирования, которые рассмотрены ранее:

1) Возможность писать тестовые сценарии на том же языке, что и приложение, например, на JavaScript, чтобы тесты были понятны как и разработчику тестов, так и разработчику тестируемой системы. Это дает возможность обмена опытом в программировании на языке, а также помощь в написании, анализе и отладке тестов как со стороны разработчиков приложения, так и со стороны разработчика тестов;

2) Поддержка инструментом операционной ОС, под которую разрабатывается мобильное приложение;

3) Обеспечение инструментом возможностей react-native для более быстрого тестирования\дебага;

4) Поддержка автосинхронизации (наибольшая проблема при тестировании пользовательских интерфейсов, необходимость постоянно синхронизировать работу приложения, и ожидание тестового сценария. В связи с этим, наличие автосинхронизации серьезно ускоряет и упрощает разработку автотестов);

5) Невысокая цена;

6) Наличие документации (хорошая документация к инструменту ускорит время на его изучение);

7) Возможность запуска тестовых сценариев как на эмуляторе, так и на реальном устройстве.

В настоящее время существуют широкое разнообразие средств для автоматизированного тестирования мобильных приложений, наиболее распространенными из которых является:

1) Calabash - это фреймворк, который служит своего рода драйвером, управляющим работой

приложения на устройстве или симуляторе. Он подходит как для Android-приложений, так и для iOS. Разработкой и поддержкой этого средства занимается компания Xamarin;

2) Appium - это open source фреймворк, который помогает автоматизировать тестирование мобильных приложений.

3) Robotium - предназначен для Android-приложений. С его помощью разработчики могут писать функциональные тесты;

4) Espresso - это инструмент для тестирования пользовательских интерфейсов Android-приложений. Основной API невелик и прост, но поскольку исходный код инструмента открыт, его можно приспособить для любых нужд;

5) XCUITest - это родной инструмент Apple;

6) TESTCOMPLETE - коммерческий продукт компании SmartBear;

7) DETOX - open-source проекта, который начали разработчики из wix.engeneering

Сравнительная характеристика перечисленных средств приведена в таблице 1.1

Таблица 1.1.

Сравнительная характеристика средств автоматизированного тестирования мобильных приложений

	Поддержка JavaScript для написания тестов	Поддержка фреймворком тестирования приложений под IOS	Цена решения	Наличие документации	Возможность запуска тестовых сценариев как на эмуляторе, так и на реальном устройстве	Поддержка автосинхронизации	Поддержка react-native
Calabash	нет	есть	бесплатный (open-source)	есть	отсутвует	отсутвует	отсутвует
Appium	есть	есть	бесплатный (open-source)	есть	присутствует	отсутвует	отсутвует
Robotium	нет	отсутвует	бесплатный (open-source)	есть	присутствует	отсутвует	отсутвует
Espresso	нет	отсутвует	бесплатный (open-source)	есть	присутствует	отсутвует	отсутвует
XCUITEST	нет	есть	бесплатный	есть	присутствует	отсутвует	отсутвует
TESTCOMPLETE	есть	есть	Платный	есть	отсутвует	отсутвует	отсутвует
DETOX	есть	есть	бесплатный (open-source)	есть	присутствует	присутствует	присутствует

Так как приложение для мобильного устройства пишется на стационарном компьютере, а для его запуска и отладки необходимо мобильное устройство, Apple совместно с Xcode поставляет инструменты для выполнения этой операции на симуляторах. При этом можно построить пользовательский интерфейс внутри Xcode для отладки, что обеспечивает следующие возможности:

· просмотр экранных форм пользовательского интерфейса, останавливая анимацию\ работу приложения

· оценка взаимодействия слоев\элементов в приложении

· выявление свойства каждого элемента на пользовательском интерфейсе. К примеру:

а) класс элемента\ набор свойств класса, его цвет;

б) как он отображается относительно других элементов;

в) идентификатор доступа (accessibility Id) элемента, то есть специальный идентификатор, указываемый разработчиком приложений или авто-генерируемый для более удобного нахождения элемента авто-тестами;

г) текст(static text\label) элемента;

Наиболее распространенными операциями пользовательского интерфейса мобильного приложения, которые подвергаются проверкам, являются:

· реакция элементов интерфейса (панели инструментов, меню, изображения, диалоги, поля ввода, списки, кнопки и т.д.) на действия пользователя;

· отсутствие обрабатываемых\необрабатываемых ошибок.

· выполнение ожидаемых в соответствии с требованиями процессов.

При разработке авто-тестов выполняются следующие операции:

· анализируются проставленные заранее идентификаторы доступа на элементах

· если их нет, либо если идентификатор динамический, в режиме отладки просматривает элементы пользовательского интерфейса, и самостоятельно выбирает свойства для программного обнаружения элемента.

Далее, сформировавшийся набор свойств искомого элемента, используется для определения элемента на экране и выполнения действий с ним.

Список литературы:

1. Carl J. Nagle. Test Automation Frameworks. URL: http://safsdev.sourceforge.net/DataDrivenTestAutomationFrameworks.htm

2. http://www.worksoft.com

3. I.B. Bourdonov, A.S. Kossatchev, V.V. Kuliamin, A.K. Petrenko. UniTesK Test Suite Architecture. Proc. of FME 2002. LNCS 2391, pp. 77-88, Springer-Verlag, 2002.

4. В.В. Кулямин, А.К. Петренко, А.С. Косачев, И.Б. Бурдонов. Подход UniTesK к разработке тестов. Программирование, 29(6):25–43, 2003.