Содержание:

ВВЕДЕНИЕ

Компьютеры тесно вошли в наш привычный мир. Компьютер не может существовать без операционной системы. ОС обеспечивает абсолютно всю работу компьютера, будь то мощный сервер или маленький телефон в кармане. Мобильная операционная система (мобильная ОС) -- операционная система для смартфонов, планшетов, КПК или других мобильных устройств. Мобильные операционные системы сочетают в себе функциональность ОС для ПК с функциями для мобильных и карманных устройств: сенсорный экран, сотовая связь, Bluetooth, Wi-Fi, GPS-навигация, камера, видеокамера, распознавание речи, диктофон, музыкальный плеер, NFC и инфракрасное дистанционное управление.

Именно ОС является визитной карточкой всех устройств. Операционные системы для мобильных устройств многообразны, но более 95% на рынке занимают всего несколько из них, а именно Android от Google и iOS - операционная система Apple. Все остальные вместе взятые занимают менее 5% рынка, сюда относятся Windows Phone и Blackberry OS.

В течение последних лет стало возможным перераспределение состава аппаратных платформ для конечных пользователей. Доля рынка мобильных устройств в форм-факторе планшетных компьютеров превысило число персональных компьютеров. Сейчас три операционные системы, iOS, Android и Windows Mobile конкурируют на рынке. Колличество приложений, разрабатываемых для мобильных платформ, достигает количества приложений для настольных операционных систем.

Количество мобильных приложений в каждой эко-системе для мобильных и планшетных компьютеров нелинейно возрастает ежегодно. В 74% смартфонов, проданных в 3-ем квартале 2017 года, была установлена операционная система Android.

Цель работы заключается в анализе специфики разработки, тестирования и внедрения мобильных приложений (МП) на существующих операционных системах.

Предметом исследования является специфика практических подходов к разработке, тестированию и внедрению мобильных приложений.

Задачами работы являются:

1. Анализ особенностей мобильных операционных систем.
2. Анализ особенностей и этапов разработки мобильных приложений.
3. Обзор ключевых понятий и типов тестирования мобильных приложений.
4. Анализ средств автоматизации тестирования.
5. Анализ основных этапов интеграции релиза мобильного приложения.
6. Анализ особенностей публикации мобильного приложения в специализированных электронных магазина.

В рамках данной работы было рассмотрено 22 литературных источника, большая часть из которых являются достаточно новыми и насыщенными полезной технической информацией в сфере исследования.

ГЛАВА 1 АНАЛИЗ СПЕЦИФИКИ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

1.1 Анализ особенностей мобильный операционных систем

1.1.1 Специфка ОС Android

Android — операционная система для коммуникаторов, планшетных компьютеров, цифровых проигрывателей, нетбуков и смартбуков, основанная на ядре Linux. Изначально разрабатывалась компанией Android Inc., которую затем купила Google. Впоследствии Google инициировала создание Open Handset Alliance (OHA), которая сейчас и занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit создаёт приложения, описанные на Си и других языках [1].

Приложения для Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik. Google предлагает для свободного скачивания инструментарий для разработки (Software Development Kit), который предназначен для x86-машин под операционными системами Windows XP, Windows Vista, Mac OS X (10.4.8 или выше) и Linux. Для разработки требуется JDK 5 либо JDK 6 [2].

Достоинства OS Android.

1. Разнообразие приложений и игр. Их число в официальном магазине давно уже преодолело миллионную отметку. Плееры, браузеры, текстовые редакторы и многое другое – всего этого в избытке и на любой вкус. Каждое решение имеет свои платные и бесплатные варианты, различающиеся функционалом и ограничениями [3].

2. Android – операционная система с открытым исходным кодом. Производство приложений, игр, всяческих поправок и обновлений упрощено до невозможности. Стремительно набирает обороты новая профессия — программист приложений для Андроид.

3. В системе хорошо реализована многозадачность, когда без проблем работает одновременно несколько приложений. Даже на не слишком мощном устройстве может эффективно работать сразу несколько вкладок браузера, музыкальный плеер и какое-нибудь ресурсоемкое приложение. Переключение между задачами происходит быстро.

4. Оперативные обновления. Компанией Google ведется непрерывная работа над улучшением функционала операционной системы, исправляются баги, вносятся изменения в интерфейс. Кроме того, независимые разработчики тоже прилагают немало усилий по совершенствованию своих приложений, быстро адаптируя их под новые версии ОС [1].

Недостатки [2]:

1.Вне зависимости от того, насколько хорошее в устройстве железо, смартфон или другой девайс под управлением OS Android приходится довольно часто подзаряжать. Связано это с тем, что сама система не экономит на предоставляемых ей аппаратных ресурсах. Частично недостаток можно исправить настройкой энергосбережения, отключением ненужных функций, но лишь частично.

2. Проблемы совместимости. Новые версии операционной системы часто конфликтуют со снятыми с продажи устаревшими устройствами или устройствами, которые выпущены неизвестными “китайскими” производителями. Таким образом мобильное устройство после обновления начинает вести себя не так, как хотелось того – батарея быстро садится, начинаются зависания, перезагрузки. Приходится принудительно откатываться на старую версию и отключать автоматическое обновление [3].

3. Обилие настроек. Любителям часами копаться в телефоне, превращая его в многофункционального помощника, разнообразие настроек придется по душе, а вот обычные пользователи, у которых на первое место выходит практичность и скорость работы, могут оказаться недовольны. В этом плане система от Apple выглядит более привлекательно.

Несмотря на все свои недостатки, операционная система Android была и будет востребована на рынке мобильных устройств. Основной причиной тому является лояльное отношение к пользователям.

Широкий ассортимент устройств под управлением Андроид охватывает все ценовые сегменты – как бюджетные модели, так и премиум класс, позволяя обзавестись устройством с операционной системой от Google практически каждому [4].

1.1.2 Специфика ОС iOS

iOS (до 24 июня 2010 года - iPhone OS) - мобильная операционная система, разрабатываемая и выпускаемая американской компанией Apple. В отличие от Windows Phone и Google Android, выпускается только для устройств, производимых фирмой Apple. Была выпущена в 2007 году. Первоначально - для iPhone и iPod touch, позже - для таких устройств, как iPad и Apple TV [6].

Все пространство экрана состоит из четырех составляющих [5]:

1. Рабочий экран (или Home Screen) - вмещает в себя 16 иконок различного пользовательского назначения: почта, календарь, фотографии, контакты, заметки, часы, калькулятор, камера, настройки, App Store и т.д.

2. Строка Dock. Находится в нижней части рабочего экрана, состоит из четырех элементов.

3. Панель навигации Spotlight с рабочими экранами и поиском - нижняя часть экрана

4. Строка состояния Status Bar - правый верхний угол экрана - отображает уровень сигнала сети, EDGE, 3G, Wi-Fi, Bluetooth, индикатор заряда батареи, состояние будильника, воспроизведение музыки и TTY.

Платформа iOS отличается удачно реализованной многозадачностью. Без каких-либо сложностей можно свернуть и развернуть утилиту. Самое главное заключается в том, что свернутые программы не оказывают воздействия на операционную систему и не снижают заряд батареи. Еще неоспоримым преимуществом является тот факт, что каждый активный процесс легко закрывается за несколько движений Удобная работа с инструментами беспроводной связи.

К положительным качествам ОС можно отнести [6]:

- интуитивно понятный интерфейс (уменьшается время, затрачиваемое на запуск программ);
- экономность (Apple гарантирует длительное время автономной работы даже при высоком уровне загруженности аппарата);
- отсутствие программных сбоев (нет зависаний и странностей в поведении);
- высокая скорость работы (неважно, запущена игра, работает браузер google или «тяжёлое» приложение);
- практичность (сравнение применяемого софта на iOS с софтом на Android показывает высокое его качество и продуманность) [5];
- защищённость (гаджет на платформе Ай ОС сложно заразить вирусом или вывести из строя по незнанию);
- облачное хранение данных, автоматическое сохранение резервных копий (первый вариант допускает совместное использование файлов на всех устройствах Apple, второй обезопасит от потери всех пользовательских данных в случае поломки или кражи гаджета);
- качественная работа в беспроводных сетях (аппарат автоматически переключается с мобильной передачи данных на Wi-Fi, и наоборот);
- многозадачность (свёрнутые приложения не потребляют ресурсы).

Недостатками являются [7]:

- закрытость файловой системы (невозможность прямой переброски файлов в Apple iPhone, iPod и iPad, отсутствие возможностей для полного обзора

содержимого устройства);
- отсутствие пользовательских настроек;
- ограничение памяти (дальнейшее её наращивание невозможно по причине отсутствия разъёмов);
- ограниченность использования софта (формула один почтовый клиент — один браузер – один магазин приложений);
- высокие цены на приложения.

1.1.3 Специфика ОС Windows Phone

Windows Phone — мобильная операционная система, являющаяся преемником Windows Mobile, но несовместима с ней. С выходом Windows Mobile версии 6.5 компания Microsoft начала создавать бренд под названием Windows Phone [5].

Так стали именоваться все телефоны с этой операционной системой, но первая операционная система Windows Phone (сразу под номером 7 — как продолжение линейки мобильных систем от Microsoft) вышла 11 октября 2010 года. Интерфейс пользователя основан на дизайнерской системе Windows Phone под названием Metro, принципы которой были ранее использованы в дизайне интерфейса Windows Media Center, Zune и Xbox.

Начальный экран составляют так называемые «живые плитки» которые отображают информацию в режиме реального времени без участия пользователя [7].

Плитки также являются ссылками на приложения, различные функции и индивидуальные объекты (контакты, веб-страницы и т. д.). Пользователь может добавлять, перемещать или удалять плитки. Начиная с Windows Phone 8 размер плиток можно регулировать, выбирая между тремя размерами отображения [2].

Установка приложений и игр на Windows Phone возможна только из официального интернет-магазина Windows Phone Marketplace. Для энтузиастов возможна разблокировка телефона при помощи утверждённого Microsoft сервиса ChevronWP Labs, в результате чего на смартфон можно устанавливать самодельные приложения или использовать его для тестирования в обход Marketplace и официальной разблокировки телефона в качестве разработчика [8].

Достоинства Windows Phone [9]:

1. Центр уведомлений. Обеспечивает доступ к расширенному списку переключателей, удаление отдельных уведомлений, возможность отвечать в уведомлениях.

2. Наличие удобного и информативного стартового экрана.

3. Поддержка джойстика на клавиатуре. Удобная альтернатива перемещению курсора с помощью увеличительного стекла у iOS девайсов, которая является одной из самых удобных.

4. Многопоточная загрузка приложений в магазине. Функция которой очень не хватает в других ОС во время восстановления из резервной копии. И в остальном это очень грамотный подход к реализации функции.

5. Новые системные приложения. Сюда входят новые офисные приложения, почта и календарь, музыка, калькулятор, будильники и часы и в особенности новый браузер.

Недостатки Windows Phone [10]:

1. Ошибки масштабирования. В первую очередь, это непонятные переносы текста в магазине, очень крупные или наоборот очень мелкие элементы интерфейса и др.

2. Некорректные сообщения в шторке уведомлений. Часть дополнительных иконок при опускании шторки внезапно могут исчезнуть.

3. Баги клавиатуры. Когда происходит набор текста вручную вместо некоторых букв могут вводиться соседние, причем только при определенной последовательности, т.е. в остальных случаях буква активна. Автозамена срабатывает часто некорректно и зачастую предлагает слова с большой буквы в середине предложения.

4. Неправильная фоновая работа приложений. Если правильно не настроить фоновую работу приложений, то при включённой экономии заряда, они могут не корректно функционировать. И даже отключив фоновую работу приложений, некоторые из них могут присылать push-уведомления.

5. Медленная анимация закрытия приложений. Также при открытии меню управления приложениями в горизонтальной ориентации позволят закрывать приложения только в ней и вертикально не поворачивается.

6. Баг экрана блокировки. Экран после разблокировки может не загореться или загореться не сразу, при этом дата и время появляются с задержкой.

1.2 Особенности разработки мобильных приложений

Процесс разработки программного обеспечения для мобильных устройств требует придерживаться определенных ограничений, что связано со спецификой работы приложений в мобильных ОС [2]. Особенности работы мобильных приложений касаются, прежде всего [11]:

- учета расхода ресурса аккумуляторной батареи мобильного устройства;

- ограничения на количество данных, передаваемых через Интернет;

- уменьшенной скорости передачи пакетов в мобильном беспроводном Интернет-соединении, в сравнении со стационарной кабельной связью;

- возможности потери пакетов при передаче в мобильном Интернет-соединении;

- количество обменов данными с внешними устройствами на Bluetooth;

- безопасностью данных пользователей [1];

- значительную фрагментацию версий операционных систем и фреймворков;

- большое разнообразие размеров экранов и разрешений мобильных устройств;

- ограничение запросов к системе определения местоположения абонента;

- ограничение на объем файла приложения;

- сравнительно небольшой лимит оперативной памяти мобильного устройства.

Основными этапами процесса разработки МП являются [12]:

1. Анализ основных трендов современного рынка информационных технологий для выявления целевой ниши, потенциальной аудитории пользователей и актуальных сфер применения МП. Исполнителем может являться системный аналитик [10].

2. Формализация цели использования МП, основных задач, которые оно позволит решить и возможного экономического эффекта. Исполнителем может являться системный аналитик.
3. Разработка технического задания в виде перечня функциональных и нефункциональных требований к будущему МП. Исполнителем может являться лидер команд разработчиков.
4. Разработка плана реализации МП, включающего перечень сроков, задач, исполнителей, методологий управления проектом. Исполнителем может являться проектный менеджер [11].
5. Разработка проектной документации для формализации функционирования создаваемого МП. Для этого используются различные методологии и нотации формализации и проектирования бизнес-процессов (BPMN, ARIS, IDEF3, IDEF0, DFD и др.). Исполнителем может являться бизнес-аналитик.
6. Разработка основных абстракций и логики взаимосвязей между компонентами МП. Это реализуется с помощью различных UML диаграмм (классов, последовательности действий, развертывания, компонентов и т.д.). Исполнителем может являться архитектор или старший разработки программного продукта.
7. Разработка прототипа интерфейса пользователя МП в виде форм с визуальными компонентами, карт маршрутов переходов с активностей. Исполнителями могут выступать дизайнеры интерфейсов и UI-UX специалисты [5].
8. Разработка программного кода реализации описанных функциональных возможностей МП. Исполнителями являются программисты.
9. Написание модульных тестов для проверки работоспособности всех методов отдельных классов. Исполнителями являются программисты.

Для организации процесса разработки МП часто используется методология создания модульных тестов перед написанием кода TestDrivenDesign (TDD) [6].

Методика разработки МП с помощью модульных тестов заключается, в первую очередь, в создании теста, а затем реализацию соответствующего кода программы для выполнения функций, которые требует созданный тест.

Методология создания программного обеспечения с помощью тестов достаточно широко используется для разработки мобильных приложений в Android.

Стандартные среды разработки Android приложений имеют в своем составе мощную систему создания модульных тестов и инфраструктуру для их запуска и

анализа результатов выполнения [10].

Диаграмма состояний для методологии разработки программного обеспечения с помощью юнит-тестов показано на рис. 1.

Например, среда разработки Eclipse предоставляет пользователям возможность выполнять юнит тесты на мобильных устройствах и на эмуляторе. Для выполнения тестов создается отдельный проект, что позволяет оставлять основной проект без существенных изменений [11].

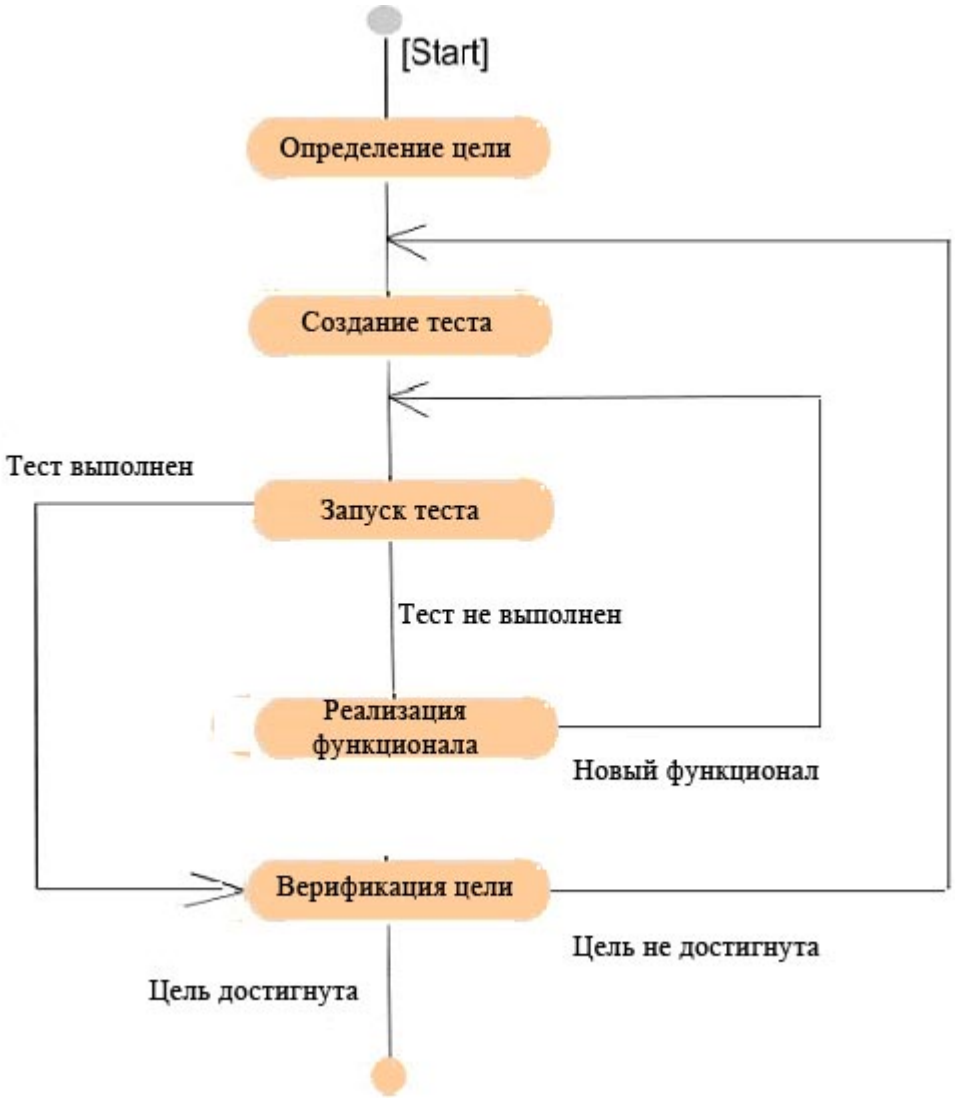

Рисунок 1 - Диаграмма состояний для методологии разработки МП с помощью юнит-тестов

ГЛАВА 2 АНАЛИЗ СПЕЦИФИКИ ТЕСТИРОВАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

2.1 Ключевые понятия и типы тестирования

Тестирование мобильных приложений (ТПМ) – производственная деятельность, направленная на определение оценки и обеспечение повышения качества МП. Необходимость подобной деятельности базируется на важности своевременного обнаружения различных проблем и дефектов в разрабатываемых программных системах [13].

Задача тестирования - выявление дефектов в программном обеспечении до того, как эти дефекты будут обнаружены заказчиком или конечным пользователем.

Тест **–** выполняемая тестовая процедура с заданными входными данными, набором исходных условий и вариантами ожидаемых результатов, которая должна быть выполнена для заданной цели. В качестве такой цели, как правило, выступает проверка нужного МП или проведение верификации работы программы по конкретным требованиям.

Компонент **–** набор функций, который будет использоваться многократно в различных тестах [15].

Создание тестов или компонентов происходит путем записи сеанса работы с приложением или web-сайтом. Также существует возможность ручного создания тестового скрипта, используя в поле ключевых слов ключевые слова из предварительно созданного хранилища объектов (object repository).

Цель ТПМ - это выявление и устранение ошибок в тестируемом объекте. Однако полностью протестировать даже несложную программу невозможно. Обычная практика тестирования такова: программное средство считается пригодным к выпуску, если в коде устранены все обнаруженные критические ошибки и 85 % некритических ошибок [14].

При тестировании МП, можно параметризовать тест или компонент, чтобы проверить, как приложение выполняет те же операции с другими данными. В этом случае можно воспользоваться таблицей данных или генератором случайных чисел. Каждый запуск сеанса, который использует параметризацию, называется

итерация.

Также можно использовать и выходные величины, чтобы извлекать данные из теста или компонента. Это позволит использовать извлеченные данные в течение исполнения сеанса в других частях теста или компонента [13].

Тестовые сценарии (ТС) применяют для проверки различных функциональных требований и оценки нефункциональных требований. В ряде случаев применяются тесты, в которых количественные параметры проведенных результатов только в общем виде удовлетворяют поставленным целям тестирования.

Тестированию подлежит [15]:

1. Программа при ее непосредственном запуске и исполнении (software).

2. Код программы без запуска и исполнения (code).

3. Прототип МП.

4. Проектная документация, содержащая:

- технические требования к работе программного продукта;
- функциональные спецификации и требования к МП;
- документы, описывающие архитектуру дизайна МП;
- план проекта и план теста;
- тестовые сценарии.

5. Сопроводительная документация и документация для пользователей:

- интерактивная помощь (on-line help);
- руководства по установке (installation guide) и использованию программного продукта (user manual).

ТС позволяет описать проверку работы МП, которую способен выполнить любой человек из производственной команды, однако чаще всего это является работой штатных или удаленных инженеров по тестированию МП.

Набор тестовых сценариев часто именуют тестовым набором (ТН).

ТС должен помочь провести проверку продукта без ознакомления со всей документацией. Написанный один раз, он удобен в поддержке, т.к. экономит много времени и сил тестировщикам [16].

Для написания тестовых сценариев, используются, как правило, следующие элементы:

- Идентификатор - уникальный идентификатор тест-кейса. Он может быть применен для единообразного понимания, о которой проверке говорится.
- Название - краткое описание сути проверки. Должно быть компактным и быть понятным.
- Описание - описание элемента тестового сценария.
- Предпосылка - действия, которые нужно выполнить перед запуском теста.
- Инструкция - это шаги теста, которые необходимо выполнить для получения ожидаемого результата.
- Test Data - данные, которые используют тестовые сценарии.
- Ожидаемый результат - сама проверка: что мы ожидаем получить после выполнения всех шагов тестирования [17].

ТМП можно классифицировать по следующим признакам [18]:

- по степени использования тестируемого МП;
- по знанию системы;
- по объекту тестирования;
- по уровню изолированности составных компонентов;
- по признаку позитивности выполняемых сценариев;
- по времени осуществления тестирования;
- по степени подготовленности к тестированию;
- по уровню тестирования.

Виды тестирования МП приведены на рис.2.

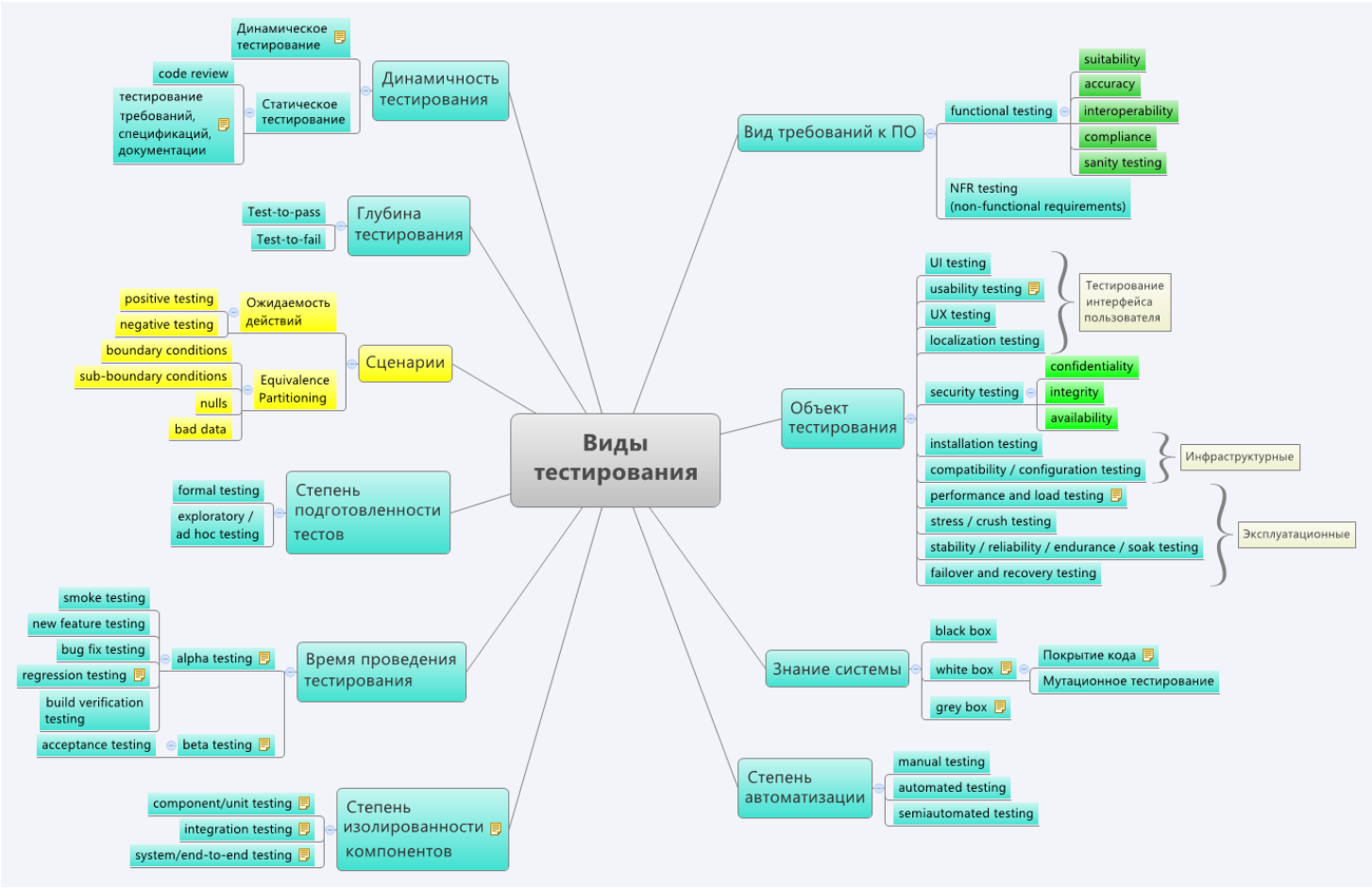

Рисунок 2 – Виды тестирования МП

По степени использования тестируемого МП выделяют:

- статическое тестирование (static testing);
- динамическое тестирование (dynamic testing).

Статическое тестирование - это процесс анализа самой разработки программного обеспечения, т. е. тестирование без запуска программы. Статическое тестирование предусматривает проверку программного кода, требований к программному продукту, функциональной спецификации, архитектуры, дизайна и т. д [14].

Динамическое тестирование - это тестовая деятельность, предусматривающая эксплуатацию (запуск) программного продукта. Динамическое тестирование предполагает запуск программы, выполнение всех ее функциональных модулей и сравнение фактического ее поведения с ожидаемым.

По знанию системы выделяют [19]:

- тестирование без учета внутреннего строения (метод «черного ящика»);

- тестирование с учетом внутреннего строения и доступом к коду (метод «белого ящика»);
- тестирование с частичным доступом (метод «серого ящика»).

При тестировании по методу «черного ящика» идеи для тестирования формируются путем предположений о сценариях, которые будут реализовываться и применяться пользователями. Поэтому метод «черного ящика» также называют «поведенческим тестированием». Примером такого тестирования является функциональное тестирование. Тестировщик запускает приложение и тестирует его функциональность, используя пользовательский интерфейс для ввода входных и получения выходных данных [17].

«Белый ящик» также известен под именами «стеклянный ящик» (glass/clear box), открытый ящик (open box) и даже никакой ящик (no box). При тестировании по методу «белого ящика» тестировщик основывает идеи для тестирования на знании устройства и внутренней логики тестируемой программы, а не на образцах поведения пользователей. В реальной жизни тестирование по методу «белый ящик» проводится либо самими программистами, написавшими код, либо их коллегами с программистской квалификацией [15]. Юнит- тестирование (unit-testing) - это часть тестирования по методу «белого ящика».

«Серый ящик» - метод, сочетающий элементы «белого ящика» и «черного ящика». С одной стороны, тестирование, ориентированное на пользователя, а значит, используются сценарии поведения пользователя («черный ящик»), с другой, информированное тестирование, т. е. тестирование со знанием, как устроена хотя бы часть анализируемой программы («белый ящик»).

Таким образом, тестирование МП проводиться тестировщиками и проходит в ряд следующих этапов [16]:

1. Тестирование работы МП с помощью модульных тестов.

2. Использование серверов непрерывной интеграции для автоматизации процесса проверки тестов по работе функциональностей.

3. Проведение black-box, white-box и grey-box тестирования.

4. Составление Test-кейсов и отправка их разработчикам МП с целью устранения возможных дефектов и ошибок в работе программного приложения.

5. Написание отчетной документации.

2.2 Анализ средств автоматизации тестирования

Для эффективной функциональной проверки работы мобильного приложения можно использовать скрипты автоматического тестирования сценариев пользователя, что позволяет проводить серию тестов по влиянию на мобильное приложение действий со стороны графического интерфейса пользователя [15]. Такой подход дает возможность более гибко строить набор регрессионных тестов для верификации функциональной части мобильных приложений и выбирать наиболее эффективный метод создания тестов: через юнит-тестирование и автоматизированное тестирование сценариев пользователя (рис. 3) [17]. Для тестирования сценариев пользователя удобно использовать системы, подобные Robotium [20]. Основными преимуществами разработки с помощью тестов являются:

- четкая и системная архитектура проекта;

- высокий процент покрытия тестами основного функционала мобильного приложения;

- поддержка инкрементального процесса создания регрессионных тестов;

- качественный анализ сценариев пользователя и их покрытия автоматизированными тестами.

Рисунок 3 – Схема одновременного использования автоматизированного тестирования по сценариям пользователя и модульного тестирования МП

2.2.1 Специфика использования SoapUI

SoapUI — это кроссплатформенное клиентское оконное приложение с открытым исходным кодом, лицензией GNU и реализацией на языке Java. Поддерживаются версии для Linux, Windows и MacOS [19].

Поддерживаются технологии SOAP/WDSL, REST, Web и HTTP, AMF, JDBC, JMS и системы автоматической сборки Maven, Hudson, Bamboo, Ant, JUnit и другие. Возможности интеграции с IDE IntellijIdeam, NetBeans, Eclipse.

SoapUI обладает гибкими возможностями для тестирования МП и веб-сервисов путем отправки им сообщений и получения ответов [20].

SoapUI является одним из ведущих функциональных инструментов для тестирования SOA. С помощью понятного в использовании графическому

пользовательскому интерфейсу, функциям корпоративного сегмента, данное приложение обеспечивает простоту и мобильность создания и выполнения функциональных, регрессионных и нагрузочные тестов разрабатываемого ПО. Таким образом в рамках единой среды SoapUI обеспечивает комплексное покрытие тестов – от SOAP и REST, на базе применения различных-служб, до JMS сообщающихся слоев предприятий, баз данных, Rich Internet Applications, и др [21].

При необходимости, возможно напрямую добавить WSDL, разработав образец запроса для всех операций в службе, и создать макет импортированного WSDL. После создания проекта предоставляются функциональные возможности создания и запуска любое количество функциональных / нагрузочных тестов и MockServices. Наличие окна Navigator, отображающего структуру дерева в левой части главного окна, прогресс испытания тестов постоянно находится в поле зрения. Из главного окна приложения можно управлять и контролировать все, что связано с проектом.

2.2.2 Специфика использования Selenium

Selenium 2 - новый набор инструментов автоматизации тестирования, предоставляющий большой набор возможностей по обеспечению управления веб-браузером, благодаря наличию целостного, объектно-ориентированного интерфейса, посредством чего исключаются различные ограничения, которым часто встречались в ранних версиях [17].

Selenium Remote Control – первая версия данного инструмента. Selenium 1 широко применяется на практике тестирования в режиме сопровождения, в силу того, что позволяет обеспечить ряд возможностей, не до конца реализованных в рамках Selenium 2. В частности, это поддержка ряда устаревших языков программирования (Perl, Fortran) и поддержку ряда браузеров [20].

Selenium IDE — это среда разработки, которая применяется ждя создания и выполнения тестовых сценариев. Является удобным в использовании дополнением (плагином) к распространенному бесплатному браузеру Firefox. Плагни поддерживает возможность использования контекстного меню, обеспечивающего пользователю возможность выбора различных элементов интерфейса на открытой странице в веб-браузере, после чего есть возможность выбора команды из списка Selenium с необходимыми параметрами.

Selenium Grid. Данный инструмент позволяет обеспечить масштабирование значительных тестовых наборов, поддерживая при этом запуск тестов, выполняемых в нескольких наборах окружений [21].

Selenium Grid обеспечивает возможность запуска тестов в параллельном режиме, что позволяет различным тестам проходить проверку одновременно на нескольких удаленных рабочих станциях или серверах. В связи с этим появляется два значительных преимущества [22].

Во-первых, в случае большого количества тестовых операций или временные затраты на имплементацию и проверку тестов слишком велико, поддержка параллельного режима способствует увеличению производительности путем разделения тестов на несколько отдельных потоков.

Во-вторых, в случае, когда разработанные тесты ПО необходимо запускать в различных рабочих окружениях, браузерах или операционных системах, появляется возможность настройки удаленных серверов развертывания с независимом запуском одного набора тестов во всех необходимых средах. Это значительно ускоряет процесс тестирования. Это является значительным преимуществом Selenium в сравнении с существующими аналогами [20].

2.2.3 Специфика использования Ranorex и IBM Rational Functional Tester

Ranorex является средством автоматизации тестирования GUI для тестирования настольных, web и мобильных приложений. Ranorex не имеет собственного языка сценариев, и использует в этом качестве стандартные языки программирования C# и VB.NET в качестве базы [21].

Ranorex поддерживает возможности фиксации действий на базе применения интегрированного рекордера, идентификации различных элементов интерфейса пользователя при помощи инструмента Ranorex Spy. Все обнаруженные элементы хранятся в формате XML в соответствующих репозиториях. Отдельный элемент в них записан с помощью нотации XPath [20].

Исполнение тестов происходит путем последовательного запуска .exe файлов test-suite. После их выполнения формируется по одному файлу формата zip на один test-suite, каждый из которых включает один файл XML с полученными результатами.

Затем программный скрипт осуществляет конвертацию XML формата в xUnit. За счет этого достигается возможность получить отчетов по каждому клиенту в Ranorex и в графическом формате представления тестов.

На базе записи данных действий происходит автоматическое формирование программного кода. Каждый шаг, при этом, можно написать вручную [22].

IBM Rational Functional Tester – это комплексный инструмент проведения автоматизированного регрессионного и функционального тестирования.

Данное программное средство предоставляет тестировщикам набор универсальных и удобных механизмов проведения автоматизированного тестирования, применяемых на практике для осуществления функционального, регрессионного тестирование, а также для тестирования пользовательского интерфейса.

Rational Functional Tester входит в состав IBM Rational Quality Manager, что представляет собой стек средств управления процессом тестирования, идентификации дефектов, управления различными версиями сценариев осуществления тестирования и менеджмента требований. Инструментальные средства, которые интегрированы в данную платформу, позволяют обеспечить процесс ускорения разработки приложений, что способствует облегчению координации и коммуникации внутри разработчиков [21].

Преимущества автоматизации ТМП:

- Повторяемость. Разработанные тестовые сценарии выполняться единообразно, «человеческий фактор» не оказывает на процесс неожиданных негативных влияний. Таким образом, тестировщик не сможет пропустить тест.

- Высокая скорость выполнения. Отсутствует необходимость в сверке этапов тестирования с инструкциями и регламентной документацией, что существенно уменьшает затраты времени на выполнение.

- Снижение затрат на поддержку. Поддержка готовых тестов и анализ результатов их работы не требуют таких затрат времени, как проведение этого объема работы вручную.

- Наличие гибких отчетов. Отчеты о результатах тестирования генерируются и рассылаются автоматически.

Недостатки [21]:

- Повторяемость. Это является и недостатком, что связано с тем, что тестировщик, который выполняет тест в ручном режиме, может акцентировать свое внимание на ряде различных деталей, анализ которых позволит найти скрытый или не очевидный программный дефект. Автоматический скрипт не позволяет этого.

- Рост расходов на поддержку. С увеличением количества релизов и развертываний в различных инфраструктурах повышаются временные и материальные затраты на поддержку [18].

- Значительные затраты на разработку скриптов автоматизации. Это является трудоемким процессом, т.к. разрабатывается приложение, тестирующее другую программу. В сложных и иерархических автоматизированных тестах как правило присутствуют свои прикладные утилиты, библиотеки и фреймворки, подключение и адаптация которых требует значительных затрат времени.

- Стоимость средства автоматизации. Лицензионное ПО, его поддержка и установка не являются дешевыми и оплачиваются не единожды. Свободные средства не имеют, как правило, достаточной степени гибкости и удобства в использовании [19].

- Пропуск мелких ошибок. Автоматический скрипт может не проверить различные мелкие ошибки, проверка которых не предусмотрена разработчиком. К таким ошибкам относят различные неточности в позиционировании окон, имеющиеся лингвистические ошибки в надписях, ошибки в работе форм, с которыми не производится непосредственное взаимодействие при выполнении скрипта.

ГЛАВА 3 ОСОБЕННОСТИ ВВОДА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ В ЭКСПЛУТАЦИЮ

3.1 Основные этапы интеграции релиза мобильного приложения

Ключевыми этапами внедрения МП в деятельность в компаниях или на мобильных устройствах отдельных пользователей являются [23]:

1. Проведении маркетинговых исследований и подготовка маркетинговых материалов в виде демонстрации преимуществ и возможностей созданного МП .
2. Публикация бета-версии МП в специализированных магазинах (Play Market, Apple App Store и др.) с целью их свободного скачивания пользователям.
3. Проведение бета-тестирования пользователями и получение обратной связи по работе МП.
4. Выпуск стабильной версии МП, техническая поддержка, сбор отзывов и обратная связь с пользователями [24].
5. Внедрение механизмов монетизации МП (предоставление пользователю бесплатной версии приложении с урезанным функционалом, расширение которого возможно путем оплаты лицензии или проведения регулярных платежей) [25].
6. Внедрение новых функциональных возможностей, обновление интерфейса, развитие МП.
7. Анализ популярности и рентабельности разработанного МП.
8. Прекращение поддержки и отказ от сопровождения с возможной выгрузкой из специализированных магазинов.

Одним из частых вопросов, возникающих в голове разработчиков мобильных приложений, является вопрос о том, как же монетизировать разработанное приложение. Существуют следующие популярные модели монетизации МП:

1. Freemium. Суть этой модели монетизации заключается в том, что пользователь скачивает не все МП, а лишь его демонстрационную бесплатную (free) версию. Пользователь обычно сталкивается с какими-либо искусственно созданными ограничениями, а разработчик в свою очередь за несколько долларов предлагает снять эти ограничения, т.е. предлагает купить пользователю премиум-версию. Другой вариант Freemium-модели позволяет скачивать полную версию МП бесплатно, но при определенных ограничениях. В таких играх ограничение снимается при покупке премиум аккаунтов. К слову, 34% из ТОП-100 кассовых приложений в App Store используют модель Freemium [23].

2. Рекламная модель. МП, разработанные с использованием этой модели, показывают своим пользователям рекламу. Зачастую такую модель используют с какой-либо другой моделью монетизации, например, с моделью Freemium. Большой прибыли с такого варианта монетизации получить практически нереально, только если МП не имеет миллионную аудиторию.

Преимущества размещения рекламы в мобильных приложениях СМИ очевидны. Средства массовой информации, будь то печатные издания, телеканалы, радиостанции, информационные агентства или онлайн-порталы, как правило, имеют состоявшуюся репутацию, часто – долгую историю существования, а самое главное – лояльную аудиторию, о которой известно намного больше, чем об аудитории иных мобильных рекламных площадок. Более глубокое представление о посетителе ресурса позволяет рекламодателям не скупиться на дорогие имиджевые кампании на данных площадках [24].

Рекламное сообщение, распространяемое в мобильном канале, должно быть выполнено на высшем уровне по креативу, оформлению и интерактивным возможностям. При этом средства массовой информации достаточно консервативны при разработке мобильных приложений. Некоторые рекламные форматы – видео, игровая реклама и т.д. – не всегда одобряются редакциями СМИ, так как могут мешать восприятию основной информации [25].

Сегодня медиакомпании продолжают активно вкладывать деньги в разработку мобильных приложений. Рекламодателям это интересно, а СМИ, привыкшие оперативно реагировать на любой спрос, держат руку на пульсе.

3.2 Особенности публикации мобильного приложения в специализированных электронных магазинах

Опубликовать МП в сторах Google Play и App Store можно двумя способами [25]:

- самостоятельно завести в магазинах аккаунты,
- опубликовать приложение на аккаунте разработчика.

Если необходимо не просто опубликовать МП, но и монетизировать его, то потребуется завести аккаунт продавца в Google Payments Merchant Center. Порядок действий такой [23]:

-
  - войти в консоль разработчика Google Play;
  - нажать «Отчеты»;
  - выбрать в меню «Финансовые отчеты»;
  - кликнуть на «Настроить аккаунт продавца»;
  - ввести все необходимые сведения о компании.

В Google Play аккаунт разработчика можно прикрепить к аккаунту продавца. Делается только один раз, т.е. отменить или изменить данную связь не получится.

Прежде всего необходимо создать иконку МП. Пользователь увидит ее в первую очередь и заинтересуется, прочитает название приложения, перейдет к странице. При подготовке иконки необходимо проявить индивидуальность. Не следует создавать иконку трендового в нише цвета — приложение смажется и потеряется на фоне своих конкурентов.

Технические требования к иконке: 512*512 px, формат 32-битный PNG, с альфа-каналом. Допустимый объем файла до 1024 КБ [24].

Размер скриншотов — от 320 px до 3840 px при соотношении сторон не более 2:1. Формат JPG или 24-битный PNG без альфа-канала. Цель скриншотов — познакомить пользователя с интерфейсом приложения и его основными фичами. В идеале после просмотра скриншотов пользователь должен убедиться, что приложение очень полезное и удобное в работе.

К скриншотам стоит написать сопроводительный текст — небольшую инструкцию по МП, которая расскажет о том, чем приложение полезно и уникально. Это поможет выделиться на фоне конкурентов [24].

Название МП может состоять из 25-55 символов. Главное, чтобы оно не содержало чужих товарных знаков и названий других приложений.

Объем краткого описания — до 80 символов, полного — до 4000 символов.

Для публикации МП в App Store необходимо соблюдать ряд следующих технических требований.

Разрешение иконки: 1024*1024 px, без закругленных углов. Файл не должен содержать слои. Разрешение скриншота: 1334*750 px (для iPhone) и 1536*2048 px (для iPad). Скрины должны быть в формате JPG и PNG, без альфа-канала [25].

Требования к названию приложения следующие. Объем до 50 символов, для поиска через iPhone — желательно до 23 символов. Без упоминания других компаний или платформ. Описание мобильного приложения — до 4000 символов, включая пробелы. В отличие от Google Play, App Store вообще не учитывает ключевики в описании.

Общие рекомендации к МП такие. Если говорить о UI, то МП должно быть интуитивно понятным, плавным и воздушным (плавность — это вообще особенность данной платформы).

Обновления у iOS выходят очень часто, поэтому нужно помнить об актуализации функционала до последних версий платформы [23].

Несмотря на тщательную подготовку приложения к публикации, App Store может отклонить его. Причин тому много: приложение падает, содержит ошибки или скрытые функции, провоцирует употребление алкоголя или наркотиков, повторяет приложения (миллионный фонарик или калькулятор калорий), ограничивает пользователей географически и многое другое.

ЗАКЛЮЧЕНИЕ

Оригинальные и удобные функции «умных» смартфонов и коммуникаторов во многом стали возможными благодаря разработке для них, по аналогии со стационарными и карманными компьютерами, собственных операционных систем. У каждой модели своя установленная в ПЗУ операционная система — сменить ее нельзя, можно только обновить. Следовательно, выбирая смартфон, стоит заранее определиться с предпочтениями и пожеланиями к ОС, взвесить достоинства и недостатки каждой из них. Современные мобильные устройства в основном базируются на одной из четырех операционных систем: iOS, Windows Mobile, Android.

Операционная система смартфона / коммуникатора становится связующим звеном между собственно аппаратом и его программным обеспечением, а также позволяет устанавливать на устройство нужные дополнительные приложения от других разработчиков - мультимедийные, офисные, коммуникационные. Для самых популярных мобильных ОС созданы тысячи (а для некоторых десятки тысяч) прикладных программ, существенно расширяющих и улучшающих функциональность смартфона, делающих его уникальным и максимально полезным для владельца. Чтобы превратить компактное устройство в GPS-навигатор или счетчик калорий, достаточно найти и установить в него нужную программу.

Можно заметить, что доля устройств, имеющих операционную систему Android, велика, правильней сказать, занимает наибольшую часть и лидирует на рынке.

Таким образом, поставленная цель данной работы достигнута. Для этого были выполнены следующие задачи:

1. Проведен анализ особенностей мобильных операционных систем.
2. Проведен анализ особенностей и этапов разработки мобильных приложений.
3. Проведен обзор ключевых понятий и типов тестирования мобильных приложений.
4. Проведен анализ средств автоматизации тестирования.
5. Проведен анализ основных этапов интеграции релиза мобильного приложения.
6. Проведен анализ особенностей публикации мобильного приложения в специализированных электронных магазинах

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коматинени С. Android для профессионалов / С. Коматинени, Д. Маклин. — М.: «Вильямс», 2011. – 231 с.
2. Голощапов А. Google Android: программирование для мобильных устройств / А. Голощапов. — СПб.: БХВ-Петербург, 2010. — С. 448.
3. Роджерс Р., Ломбардо Д. Android. Разработка приложений / Р. Роджерс, Д. Ломбардо. — Москва: ЭКОМ Паблишерз, 2010. — С. 400.
4. Голощапов А. Google Android: программирование для мобильных устройств / А. Голощапов. – СПб.: БХВ-Петербург, 2010. – 448 с.
5. Конвэй Дж. Программирование под iOS. Для профессионалов / Дж. Конвэй, А. Хиллегасс. – СПб.: Питер, 2013. – 608 с.
6. Марк Д. Разработка приложений для iPhone, iPad и iPod Touch с использованием iOS SDK / Д. Марк, Д. Натинг, Д. Ламарш. – М.: ООО Вильямс, 2012. – 624 с.
7. Махер А. Программирование для iPhone / А. Махер. – М.: Эксмо, 2013. – 368 с.
8. Коматинэни С. Google Android: программирование для мобильных устройств / С. Коматинэни, Д. Маклин, С. Хэшими. – СПб.: Питер, 2011. – 736 с.
9. Коматинени С. Android для профессионалов. Создание приложений для планшетных компьютеров и смартфонов / С. Коматинэни, Д. Маклин. – М.: Вильямс. — 880 с.
10. Роджерс Р., Android. Разработка приложений / Р. Роджерс, Д. Ломбардо. — М.: ЭКОМ Паблишерз, 2010. — 400 с.
11. Донн Ф. Android: разработка приложений для чайников / Ф. Донн, Д. Маклин. — М.: Диалектика, 2011. — 336 с.
12. Коматинэни С. Google Android: программирование для мобильных устройств / С. Коматинэни. — СПб.: Питер, 2011. — 736 с.
13. Бахтизин В.В. Автоматизация тестирования программного обеспечения / В.В. Бахтизин. — Минск: БГУИР, 2012. — 72 с.

14. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. – СПб.: Издательский дом "ПИТЕР", 2014. – 426 с.
15. Блек Р. Ключевые процессы тестирования / Р. Блек. – М.: Академия, 2013. - 544 с.
16. Винниченко И. Автоматизация процессов тестирования / И. Винниченко. – СПб.: Питер, 2015. – 451 с.
17. Дворянкин А.М. Основные методы тестирования программного обеспечения / А.М. Дворянкин, А.А. Ерофеев, А.В. Аникин. — Волгоград: ВолгГТУ, 2015. — 120 с.
18. Канер С. Тестирование программного обеспечения / С. Канер. – К.: ДиаСофт, 2014. — 612 с.
19. Котляров В. Основы тестирования программного обеспечения / В. Котляров. – СПб.: Бином. Лаборатория знаний, 2016. – 421 с.
20. Криспин Л. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд / Л. Криспин, Дж. Грегори. — М.: Вильямс, 2012. — 514 с.
21. Куликов С.С. Тестирование программного обеспечения / С.С. Куликов. — Минск: Четыре четверти, 2015. — 294 с.
22. Липаев В.В. Тестирование программ / В.В. Липаев. — М.: Радио и связь, 2014. — 296 с.
23. Рамануджам М. Монетизация инноваций. Как успешные компании создают продукт вокруг цены / М. Рамануджам, Г. Таке. - М.: Библос, 2017. — 320 с.
24. Шмидт Э. Мобильная реклама и аналитика для начинающих / Э. Шмидт. – М.: Фабрика бизнеса, 2012. -144 с.
25. Меркулов А. Монетизация бизнеса / А.Меркулов. – М.: Фабрика бизнеса, 2014. – 213 с.