МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., док. физмат. наук, зав. каф |  |  |  | Смирнов А.О. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

|  |  |
| --- | --- |
| на тему | Применение средств автоматизации для повышения эффективности |
| тестирования мобильных приложений | |
|  | |

|  |  |
| --- | --- |
| выполнена | Барановым Семеном Алексеевичем |
| фамилия, имя, отчество студента в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 01.04.02 |  | Прикладная математика и информатика |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности |  |  |  |
|  | код |  | наименование направленности |
|  | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы № | М210М |  |  |  | С.А.Баранов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., док. физмат. наук, зав. каф |  |  |  | Смирнов А.О. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ

Заведующий кафедрой №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., док. физмат. наук, зав. каф |  |  |  | Смирнов А.О. |
| должность, уч. Степень, звание |  | подпись, дата |  | инициалы, фамилия |

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

|  |  |  |  |
| --- | --- | --- | --- |
| студенту группы № | М210М |  | Баранов Семен Алексеевич |
|  |  |  | (фамилия, имя, отчество) |

|  |  |
| --- | --- |
| на тему | Применение средств автоматизации для повышения эффективности |
| тестирования мобильных приложений | |

|  |  |  |  |
| --- | --- | --- | --- |
| утвержденную приказом ГУАП от |  | № |  |

|  |  |
| --- | --- |
| Цель исследования: | Определить, насколько эффективнее тестирование с применением |
| средств автоматизации по сравнению с ручным тестированием | |
|  | |

|  |  |
| --- | --- |
| Задачи исследования: | 1. Изучить теоретические основы автоматизации тестирования |
| 1. Проанализировать доступные средства по автоматизации тестирования 3. Создать мобиль- | |
| ное приложение 4. Протестировать полученное приложение с помощью средств автоматизации | |

|  |  |
| --- | --- |
| Содержание диссертации (основные разделы): | 1. Теоретические основы автомати- |
| зации тестирования 2. Анализ доступных средств автоматизации тестирования мобильных | |
| приложений. 3. Создание мобильного приложения 4. Создание и прогон автоматизирован- | |
| ных тестов | |
|  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Срок сдачи диссертации « |  | » |  | 20 |  |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доц., док. физмат. наук, зав. каф |  |  |  | Смирнов А.О. |
| должность, уч. Степень, звание |  | подпись, дата |  | инициалы, фамилия |

Задание принял к исполнению

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| студент группы № | М210М |  |  |  | Смирнов А.О. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

**ОГЛАВЛЕНИЕ**

**ВВЕДЕНИЕ1**

**1** **Теоретические основы автоматизации тестирования2**

1.1 Роль тестирования в жизненном цикле разработки программного обеспечения3

1.2 Виды тестирования программного обеспечения4

1.3 Ручное тестирование программного обеспечения 5

1.4 Автоматизированное тестирование программного обеспечения 6

**2 Анализ доступных средств автоматизации тестирования мобильных приложений7**

2.1 UI Automator8

2.2 Espresso9

2.3 Earl Grey10

2.4 Detox11

2.5 Appium12

2.6 Robotium13

2.7 Test Complete14

2.8 Нативные14

2.9 Ненативные14

2.9 Смешанные14

**3 Создание мобильного приложения**1**5**

3.1 Определение требований16

3.2 Разработка дизайна приложения17

3.3 Программирование логики18

3.4 Отладка приложения19

**4 Создание автоматизированных тестов20**

4.1 Создание тест-кейсов для тестирования21

4.2 Автоматизация тест-кейсов22

4.3 Прогон полученных тест-кейсов и анализ полученных результатов23

# ВВЕДЕНИЕ

**Актуальность исследования** обусловлена широким распространением высоконагруженных мобильных приложений. Современное мобильное приложение имеет обширный функционал, сложную структуру интерфейса, множество интеграций со сторонними сервисами.

Для тестирования таких сложных систем может потребоваться большое количество ресурсов. Недостаточное тестирование может привести к большому количеству ошибок, сбоев, уязвимостям, что в свою очередь может привести к потерям прибыли, а в некоторых специфических сферах к причинению вреда здоровью и даже смертям.

Ручное тестирование подобных систем может занимать значительное время. Внедрение средств автоматизации в процесс тестирования поможет сократить время на тестирование, увеличить его эффективность. Благодаря этому продукты станут более качественными, надежными и безопасными.

**Целью работы** является определить, насколько эффективнее тестирование с применением средств автоматизации по сравнению с ручным тестированием (с учетом времени на разработку и поддержку автоматизированных тестов, стоимости трудозатрат ручного и автоматизированного тестирования).

**Задачи** необходимые для достижения целей данной работы:

1. Изучить теоретические основы автоматизации тестирования;
2. Проанализировать доступные средства по автоматизации тестирования;
3. Создать мобильное приложение, которое необходимо протестировать;
4. Протестировать полученное приложение с помощью средств автоматизации. Оценить эффективность тестирования.

**Научная новизна** обусловлена отсутствием научных исследований в области, которые бы ответили бы на вопросы:

* насколько тестирование с использованием средств автоматизации быстрее, чем тестирование без применения средств автоматизации;
* насколько тестирование с использованием средств автоматизации дешевле, чем тестирование без применения средств автоматизации.

Большинство научных работ в данной предметной области рассказывают о плюсах и минусах, приведенных выше видов тестирования, о проблемах внедрения автоматизированного тестирования в проект, о принципах раннего тестирования, о рекомендациях что именно стоит автоматизировать, но в них не приведены какие-то конкретные числа и примеры.

**Практическая значимость** работы состоит в сокращении времени тестирования за счет принятия решения об внедрения средств автоматизации в процесс разработки мобильного приложения.

**Методы исследования**:

* Теоретический анализ технической документации;
* Сравнение;
* Индукция.

# 1 Теоретические основы автоматизации тестирования

На данный момент автоматизация тестирования в различных его формах внедряется и развивается в широком множестве сложных систем. Его применение в одних случаях помогает увеличить эффективность тестирования, в других случаях просто необходимо ввиду либо невозможности провести ручное тестирование, либо высокой стоимости, сложности его проведения.

Автоматизированные тесты используют такие известные компании как Сбербанк, Альфа-Банк, Google, Amazon, Microsoft и т.д. Для того, чтобы понять нужно ли использовать автоматизацию для конкретного продукта, компании проводят внутренние исследования. На основании этих исследований принимается решение о внедрении автоматизации в процесс тестирования.

Для того, чтобы понять основы автоматизации тестирования, необходимо понять суть мануального (ручного) тестирования. И в целом определить роль тестирования в жизненном цикле разработки программного обеспечения. Любой специалист по UI автотестам или Unit-тестам понимает принципы ручного тестирования.

## 1.1 Роль тестирования в жизненном цикле разработки программного обеспечения

Тестирование программного обеспечения имеет большое множество определений, раскрывающих понятие с разных сторон. Тестирование программного обеспечения рассматривают как отдельный этап жизненного цикла программного обеспечения, как процесс, происходящий на всех этапах жизненного цикла ПО, как отдельную дисциплину и т. д.

Ввиду того, что необходимо оценить эффективность автоматизации тестирования по сравнению с ручным тестированием, то наилучшим образом подходит следующее определение:

**Тестирование программного обеспечения** — проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, для получения информации о ее качестве для заинтересованных лиц.[[1]](#footnote-1)

Суть автоматизации тестирования в конечном итоге сравнить ожидаемое поведение программы с ее реальным и, если они не совпали, сообщить об этом.

Сами же понятия ручного и автоматизированного тестирование обычно включают в себя гораздо больше. Здесь же все зависит от обозначенного круга ответственности специалистов по тестированию конкретного продукта. Поэтому, чтобы сравнить эти два вида было необходимо рассмотреть их с одной общей для этих видов стороны.

Любой специалист по тестированию тем или иным образом ищет несоответствия между реальным поведением программы и ее ожидаемым поведением, а затем оформляет их как баг-репорт. Эти баг-репорты являются формой информации о состоянии программного обеспечения.

В данном случае тестирование проявляется себя наибольшим образом как один из этапов жизненного цикла разработки программного обеспечения. Далее будет дано наиболее подходящее определение жизненного цикла программного обеспечения.

**Жизненный цикл разработки программного обеспечения** представляет собой процесс создания программного обеспечения от начала до конца. SDLC включает в себя последовательность фаз, этапов и действий, которые разработчики и инженеры программного обеспечения выполняют для создания и поддержки программного продукта. Общепринятая модель SDLC включает следующие основные этапы: [[2]](#footnote-2)

1. **Планирование. На этой стадии определяются цели проекта, ресурсы, риски, бюджет.**
2. **Анализ требований**. В этой фазе происходит сбор и анализ требований пользователей и бизнеса. Определяются функциональные и нефункциональные требования к программному продукту.
3. **Проектирование. На этом этапе происходит разработка архитектуры программного продукта, определяются технологии, инструменты и платформы разработки.**
4. **Реализация (разработка). В этой фазе программисты начинают писать код, реализуя функциональность и логику программного продукта на выбранном языке программирования. Стоит отметить, что на данном этапе пишутся автотесты, такие как UI-тесты, Unit-тесты.**
5. **Тестирование. После завершения разработки проводятся тесты для проверки работоспособности, безопасности и соответствия требованиям программного продукта.**
6. **Внедрение (развертывание). После успешного завершения тестирования программное обеспечение разворачивается на целевой среде и становится доступным для конечных пользователей. (Это может быть общедоступный домен сайта, либо публикация приложения на любой общедоступной площадке такой как AppStore, Google Play, RuStore, App Gallery и другие, так же встречается распространение любым другим образом)**
7. **Сопровождение (поддержка). После внедрения происходит постоянная поддержка и обновление программного продукта, включая исправление ошибок, добавление новых функций и адаптацию к изменениям в окружающей среде.**[[3]](#footnote-3)

Важно отметить, что тестирование должно быть «сквозным», то есть проводиться на всех этапах жизненного цикла программного обеспечения тем или иным образом. Чем раньше будет внедряться тестирование в проект, тем лучше.

Например:

* На этапе **анализа требований** документация проверяется на завершенность, недвусмысленность и согласованность. Главная задача — убедиться в том, что требования были правильно интерпретированы и остаются корректными, понятными и последовательными.
* На этапе **разработки важно** проводить юнит-тестирование, интеграционное тестирование и системное тестирование. В самом начале этого этапа разработки проводится юнит тестирование. Этот процесс представляет собой проверку отдельного модуля системы или функционала. Интеграционное тестирование проводится после того, как несколько модулей объединены вместе как отдельная часть приложения. В дальнейшем в процессе разработки все больше и больше модулей объединяются воедино. После того, как разработка закончена, наступает время подготовки к системному тестированию.
* На этапе **тестирования** тесты проводятся независимо от того, проводились ли они на предыдущих этапах. Должно быть проведено полное функциональное тестирование и тестирование пользовательских интерфейсов, а все обнаруженные дефекты должны быть задокументированы в системе баг-трекинга. Помимо этого, применяется регрессионное тестирование. После завершения отладки предоставляется оценка общего качества продукта. После завершения последнего теста процесс тестирования программного обеспечения считается законченным. *На этом этапе так же пишут и исполняют авто-тесты.*
* На этапе **сопровождения** остается необходимость в тестировании, проводимом на этапе эксплуатации и поддержки. Разные пользователи могут работать в абсолютно разных окружениях. Поэтому всегда возможно, что новые ошибки, которые не были выявлены ранее дадут о себе знать. Более того, пользователи могут использовать программное обеспечение изначально непредвиденным способом. Это, в свою очередь, может вызвать некоторые непредвиденные проблемы. В таком случае потребуется вмешательство отдела обеспечения качества.

Выводы:

Таким образом, тестирование, с одной стороны, является одним из этапов жизненного цикла разработки программного обеспечения, включающего планирование, анализ требований, проектирование, реализацию, тестирование, внедрение, сопровождение и поддержку.

С другой стороны, тестирование пронизывает каждый из вышеперечисленных этапов. Для повышения качества и соответствия требованиям продукта тестирование в той или иной форме должно проводиться на каждом из этапов жизненного цикла программного обеспечения.

Для удобства далее тестирование будет рассматриваться как один из этапов жизненного цикла. При чем, тестирование будет рассмотрено как сравнение реального поведения программы с ее ожидаемым поведением на ограниченном наборе тестов. Только таким образом, возможно сравнить два его вида – автоматизацию тестирования и ручное тестирование.

## Виды тестирования программного обеспечения

В контексте тестирования программного обеспечения существует большое множество видов тестирования. Для некоторых видов тестирования существуют отдельные профессии. Например, тестирование безопасности, нагрузочное тестирование, автоматизация тестирования, модульное тестирование (unit-тесты).

Далее будут рассмотрены лишь небольшое количество видов, достаточных для достижения целей работы и выполнения поставленных задач.

**По подходам к исполнению тестирования выделяют:**

* Скриптовое (сценарное) тестирование – это формализованный подход к тестированию, основанный на использовании заранее подготовленных тест-кейсов и наборов тест-кейсов. Это самый распространённый способ тестирования, позволяющий достичь максимальной полноты исследования продукта благодаря строгой систематизации процесса, удобству применения метрик и широкому набору рекомендаций.
* Исследовательское тестирование - частично формализованный подход, в рамках которого тестировщик работает с продуктом по выбранному сценарию, который может дорабатываться в процессе выполнения для более полного исследования приложения.
* Сессионное тестирование - компромисс между исследовательским и скриптовым подходом. Процесс тестирования разбивается на сессии определенной длины (Обычно сессия длится не более 90 минут).

**По степени автоматизации:**

* Ручное тестирование — это процесс проверки программного обеспечения без использования автоматизированных инструментов. Тестировщик взаимодействует с программой как обычный пользователь, выполняя различные сценарии использования и тестовые сценарии, вводя данные, наблюдая за результатами и проверяя наличие ошибок или неожиданного поведения.
* Автоматизированное тестирование — это процесс проверки программного обеспечения с использованием специальных программных инструментов, которые выполняют тесты автоматически, без участия человека. Тестировщик создаёт скрипты или сценарии тестирования, содержащие инструкции для выполнения определённых действий и проверки результатов. Эти сценарии запускаются на специальных инструментах для автоматизации тестирования, которые эмулируют действия пользователя и анализируют результаты выполнения.
* Полуавтоматизированное тестирование — это процесс, при котором часть тестов выполняется автоматически с помощью автоматизированных инструментов, а другая часть — вручную тестировщиками. Такой подход является наиболее распространенным, так как в некоторых случаях автоматизировать сценарий бывает сложно, невыгодно либо просто невозможно.

Так же к последнему подходу относится применение различных средств, например, регулярных выражении (Regex), средств мониторинга трафика (Charles, Fiddler, Chrome Dev Tools), средств автоматизации вызовов и анализа HTTP-запросов (Postman).

Это позволяет сочетать преимущества ручного и автоматизированного тестирования, обеспечивая более полный и качественный анализ программного обеспечения.

**По принципам работы с приложениями:**

* Позитивное — это тестирование с использованием сценариев, соответствующих нормальному поведению системы. Оно помогает определить, что система выполняет свои функции должным образом.
* Негативное тестирование, наоборот, использует сценарии, соответствующие нештатному поведению системы. Оно проверяет устойчивость системы к различным воздействиям, правильности обработки ошибок и работу с некорректными данными.

**По запуску кода на исполнение:**

* Статическое тестирование — это процесс проверки программного обеспечения без запуска кода на исполнение. Оно включает обзоры, анализ кода и статический анализ. Статическое тестирование проводится на ранних этапах жизненного цикла программного обеспечения и помогает обнаружить дефекты в документах и коде.
* Динамическое тестирование — это процесс проверки программного обеспечения путём запуска кода на исполнение. Динамическое тестирование проводится на более поздних этапах жизненного цикла программного обеспечения и помогает проверить реальное поведение и функциональность приложения.

**По целям и задачам:**

* Функциональное тестирование — это проверка способности программного обеспечения решать задачи, необходимые пользователям. Оно включает тестирование функциональной пригодности, точности, способности к взаимодействию, соответствия стандартам и правилам, а также защищённости.
* Нефункциональное тестирование направлено на проверку реализуемости нефункциональных требований, таких как производительность, удобство использования, портируемость (установка) и надёжность (отказ/восстановление), возможность обновления, безопасности.

Нефункциональный вид тестирования в свою очередь делится на несколько подвидов:

* Тестирование производительности — это вид тестирования, который проводится для определения скорости или эффективности работы системы под нагрузкой. Оно включает в себя определение скорости обработки, надежности и масштабируемости системы.
* Тестирование совместимости — это процесс проверки, как ваше программное обеспечение будет работать в различных средах, включая различные операционные системы, браузеры, сетевые окружения и т.д. Целью является убедиться, что ваше ПО совместимо со всеми потенциальными рабочими средами.
* UX-тестирование (тестирование удобства пользования) — это процесс, в ходе которого определяется, насколько удобно и просто пользователю взаимодействовать с продуктом. Это включает в себя понимание, насколько интуитивно понятен интерфейс, как легко осуществляется навигация, и насколько удовлетворены пользователи в целом.
* Тестирование надежности — это вид тестирования, который проводится для определения способности системы или компонента выполнять требуемые функции без сбоев в определенных условиях и в течение определенного периода времени.
* Тестирование безопасности — это процесс идентификации уязвимостей и слабых мест в системе с целью предотвращения несанкционированного доступа, потери данных, вторжений и других угроз.
* Тестирование портируемости — это процесс проверки, насколько легко программное обеспечение может быть перенесено с одной среды в другую. Это может включать в себя перенос между различными операционными системами, различными версиями одной и той же операционной системы или даже различными оборудованиями.

**По доступу к коду и архитектуре программного обеспечения:**

* Тестирование белого ящика — это процесс проверки кода и внутренней структуры программного обеспечения.
* Тестирование чёрного ящика — это поведенческое тестирование, при котором специалист по тестированию не знает внутреннюю структуру и устройство системы.
* Тестирование серого ящика — это комбинация тестирования белого и чёрного ящиков, где специалист имеет частичное представление о внутренних процессах системы.

**По уровню тестирования:**

* Модульное тестирование — это тестирование отдельных модулей или компонентов программы, например, функций или классов. Цель модульного тестирования — выявление ошибок в реализации алгоритмов и определение степени готовности системы к разработке и тестированию.
* Интеграционное тестирование — это тестирование взаимодействия между несколькими модулями системы. Основная задача интеграционного тестирования — поиск дефектов, связанных с ошибками в интерфейсном взаимодействии между модулями.
* Системное тестирование рассматривает тестируемую систему в целом и оперирует на уровне пользовательских интерфейсов. Основная задача системного тестирования — выявление дефектов, связанных с работой системы в целом, таких как неправильное использование ресурсов, несовместимость с окружением и неудобство в применении.
* Приемочное тестирование фокусируется на проверки и демонстрации готовности системы к разворачиванию, что означает удовлетворение системой пользовательских потребностей. Основными формами приемочного тестирования являются: пользовательское приемочное тестирование, эксплуатационное приемочное тестирование, контрактное и нормативное приемочное тестирование, альфа-тестирование и бета-тестирование.

**По природе программного обеспечения:**

* Тестирование веб-приложений
* Тестирование настольных приложений
* Тестирование мобильных приложений

Эти виды тестирования отличаются концентрацией на определённых функциях и особенностях приложений, использованием специфических инструментов и техник.

Выводы:

В тестировании существует множество разнообразных видов тестирования программного обеспечения, каждый из которых имеет свои особенности и применяется в зависимости от преследуемых целей и поставленных задач.

Виды тестирования программного обеспечения классифицируется по:

1. По природе программного обеспечения;
2. По уровню тестирования;
3. По доступу к коду и архитектуре программного обеспечения;
4. По целям и задачам;
5. По запуску кода на исполнение;
6. По принципам работы с приложениями;
7. По степени автоматизации;
8. По подходам к исполнению тестирования.

Таким образом, разнообразие видов тестирования и подходов позволяет выбирать наиболее подходящий способ и направление для обеспечения качества программного обеспечения в соответствии с требованиями и целями проекта.

**Библиографический список**

1. ISO/IEC TR 19759:2015. Разработка программного обеспечения. Руководство к Своду Знаний по программной инженерии (SWEBOK)
2. 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology
3. Скрябин А. М., Кардаш Д. И. Жизненный цикл композиционно-адаптируемого программного обеспечения // Аспирант и соискатель. 2008. № 2. С. 171-174.

1. ISO/IEC TR 19759:2015. Разработка программного обеспечения. Руководство к Своду Знаний по программной инженерии (SWEBOK) [↑](#footnote-ref-1)
2. 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology [↑](#footnote-ref-2)
3. Скрябин А. М., Кардаш Д. И. Жизненный цикл композиционно-адаптируемого программного обеспечения // Аспирант и соискатель. 2008. № 2. С. 171-174. [↑](#footnote-ref-3)