Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

УДК 004.05

Ериксонова Ульяна Владимировна

АВТОМАТИЗАЦИЯ ВИЗУАЛЬНОГО ТЕСТИРОВАНИЯ САЙТОВ С ГИБКОЙ АРХИТЕКТУРОЙ С ИСПОЛЬЗОВАНИЕМ APPLITOOLS

АВТОРЕФЕРАТ

на соискание степени магистра информатики и вычислительной техники по специальности 1-40 81 01 - Информатика и технологии разработки программного обеспечения

Научный руководитель
Поттосина Светлана Анатольевна
кандидат физико-математических
наук, доцент

Минск 2018

ВВЕДЕНИЕ

Тестирование программного обеспечения является важной и неотъемлемой частью создания программного продукта. От того, насколько досконально проведены тесты, зависит то, как скоро проект будет сдан окончательно, и будет ли необходимость впоследствии устранять ошибки.

В последнее время из-за постоянно ускоряющегося развития интернет-технологий и увеличивающейся скорости появления все новых и новых веб-приложений привело к тому, что объем материалов на сайтах стал стремительно возрастать. Из-за этого традиционные для прошлого десятилетия «ручные» технологии написания и поддержки веб-сайтов (сайт в то время состоял из статичных HTML-страниц и набора дополнительных специализированных скриптов), стали не успевать за стремительно изменяющимися условиями бизнеса. Это привело к необходимости внедрения подхода, при котором сайты могли бы легко конфигурироваться и видоизменяться, такие сайты также получили название сайтов с гибкой архитектурой.

Соразмерно возникновению такого подхода к созданию сайтов возникла необходимость подобного рода сайты тестировать. Однако классического автоматизированного тестирования в данных условиях становится недостаточно в силу часто изменяющихся требований и наполнения, в результате чего встает вопрос о выборе наиболее качественного и такого же гибкого подхода к тестированию, который, в то же время, был бы выгодным в условиях постоянно происходящих изменений и обновлений.

Классическое автоматизированное тестирование подразумевает под собой разработку и использование специального программного обеспечения для запуска и контроля выполнения тестовых сценариев и сравнения реальных и запланированных результатов согласно спецификации. Однако оно не является абсолютно гибким в условиях постоянно изменяющихся требований, и потребует большого количества времени и материальных вложений, нужных для адаптации уже написанного кода под вновь вышедшую версию сайта, что в случае постоянно происходящих обновлений является недопустимым.

Оптимальным решением данной проблемы в таком случае является использование автоматизированного тестирования в сочетании с визуальным, что позволяет снизить риски и затраты, а также повысить гибкость программного продукта.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования

По причине высоких временных и денежных затрат, которые требуются на проведение тестирования сайтов с гибкой архитектурой, встает вопрос об удешевлении и оптимизации данного процесса. Когда речь идет о тестировании функциональном, то логичным и актуальным средством снижения затрат выступает автоматизация.

Однако, учитывая специфику сайтов с гибкой архитектурой, частые изменения в их визуальном представлении, а также необходимость эти изменения оперативно отслеживать, очевидным выходом, удовлетворяющим всем заданным параметрам, является проведение тестирования визуального. На данный момент число программных средств, предоставляющих одновременно возможности как автоматизации и визуализации является ограниченным, и, зачастую, не удовлетворяет одновременно всем требованиям, предъявляющимся к такого рода средствам.

Таким образом, автоматизация визуального тестирования сайтов с гибкой архитектурой является актуальной на сегодняшний день.

Степень разработанности проблемы

Визуализация в контексте автоматизированного тестирования на момент проведения исследования не является хорошо развитой. Это связано с рядом факторов: во-первых, сайты с гибкой архитектурой существуют не столь продолжительный период времени, соответственно еще не были разработаны гибридные технологии, позволяющие одновременно поддерживать как процесс автоматизации, так и процесс визуализации.

Во-вторых, работа с обработкой визуального ряда не характерна для тестирования прошлых лет, а относится скорее к сфере работы с большими объемами данных.

Анализ существующих средств визуального тестирования также показал неполноту данной сферы. Был составлен перечень инструментов, позволяющих реализовывать ту или иную функциональность по отдельности, однако средства, которое бы удовлетворяло всем поставленным требованиям в процессе исследования найдено не было. Откуда закономерно остро возникает необходимость для поиска или реализации такого средства. При разработке фреймворка были использованы технологии Applitools и Gherkin,

представляющие наиболее эффективное сочетание для автоматизации тестирования сайтов с гибкой архитектурой.

Цель и задачи исследования

Целью данной работы является оптимизация процесса тестирования сайтов с гибкой архитектурой путем его автоматизации и визуализации с использованием технологии Applitools.

Для достижения поставленной цели были сформулированы следующие задачи:

1 Основываясь на оценке эффективности и производительности, привести обоснование рациональности внедрения автоматизации визуального тестирования для сайтов с гибкой архитектурой.

2 Сформировать инструментарий аналитической части системы, в основу которого будет положен анализ существующих средств визуального и автоматизированного тестирования.

3 Разработать фреймворк для автоматизации визуального тестирования сайтов с гибкой архитектурой на основе технологии Applitools с использованием псевдоязыка Gherkin для описания поведения системы.

Область исследования

Содержание диссертационной работы соответствует образовательному стандарту высшего образования второй ступени (магистратуры) специальности 1-40 81 01 «Информатика и технологии разработки программного обеспечения».

Теоретическая и методологическая основа исследования

Основой диссертации стали исследования работ ведущих отечественных и зарубежных специалистов в области визуализации и автоматизации тестирования. При решения поставленных задач использовались методы Tribes&Squards для организации процесса, показатели окупаемости инвестиций для определения рациональности внедрения автоматизированного тестирования, подход BDD для проведения тестирования, методы объектно-ориентированного программирования.

В качестве инструментальных средств использовались объектно-ориентированный язык программирования *Java*, фреймворки Applitools и Gherkin, сервер Bitbucket.

Формирование теоретической и методологической базы исследования проводилось на основе данных, предоставляемых ведущими корпорациями в области визуализации и автоматизации тестирования, а также были использованы статистические архивы компании EPAM Systems.

Научная новизна

Научная новизна заключается в формировании аналитической части системы. Был произведен анализ существующих технологий и средств визуального тестирования, а также возможностей автоматизации данного процесса.

Теоретическая значимость диссертации заключается в описании процесса организации автоматизированного визуального тестирования, применимого к сайтам с гибкой архитектурой.

Практическая значимость результатов исследования заключается в разработанном подходе реализации фреймворка для автоматизации и визуализации тестирования, возможности его масштабируемости и расширяемости, внедрении в более крупные системы.

Основные положения, выносимые на защиту

1 На основе оценки производительности и эффективности процесса визуального тестирования, приведено обоснование рациональности внедрения автоматизированного тестирования сайтов с гибкой архитектурой с использованием Applitools.

2 Сформирован инструментарий аналитической части системы, в основу которого положен анализ существующих средств визуального и автоматизированного тестирования.

3 Разработан фреймворк для автоматизации визуального тестирования сайтов с гибкой архитектурой с использованием Applitools и псевдоязыка Gherkin для описания поведения системы.

Апробация диссертации и информация об использовании её результатов

Основные теоретические результаты и законченные этапы диссертационной работы отражены в материалах 53-й научно-технической конференции аспирантов, магистрантов и студентов БГУИР, международной научно-практической конференции «Современный инновационно-

инвестиционный механизм развития национальной экономики».

Публикации

Изложенные в диссертации основные положения и выводы опубликованы в 3 печатных работах, представленные в виде трех докладов на научных конференциях.

Общий объем публикаций по теме диссертации составляет 12 страниц.

Структура и объем работы

Диссертация состоит из введения, общей характеристики работы, трех глав с краткими выводами по каждой главе, заключения, библиографического списка и приложений.

Во введении рассмотрено современное состояние проблемы организации процесса автоматизации визуального тестирования сайтов с гибкой архитектурой, определены основные цели и задачи исследований.

В первой главе раскрыты понятия качества программного продукта и тестирования в контексте проверки качества. Выявлены проблемы тестирования сайтов с гибкой архитектурой, а также рассмотрены средства и инструменты поддержки визуального тестирования таких сайтов.

Во второй главе были рассмотрены существующие средства автоматизации визуального тестирования. Также проведен их сравнительный анализ, выявлены их преимущества и недостатки, был сделан вывод о том, что существующие технологии не предоставляют инструментария, способного в полной мере удовлетворить потребности тестирования сайтов с гибкой архитектурой. Проведен анализ технологий, применяемых при описании поведения системы и, на основе сравнительного анализа, сделан вывод о том, что технология Gherkin в наибольшей степени удостоверяет заданным условиям тестирования. В этой главе приступили к поиску необходимого подхода автоматизации визуального тестирования, проанализировали существующие технологии, позволяющие создавать фреймворки для тестирования сайтов с гибкой архитектурой. По результатам поиска принято решение об использовании технологии Applitools как лучшего средства для визуализации автоматизированного тестирования.

Третья глава посвящена постановке задачи, а также рассмотрению методов ее решения. Были сформированы требования к разрабатываемому фреймворку. Чтобы сделать процесс разработки понятным, были построены

диаграммы последовательности системы, диаграмма вариантов использования и обобщенный алгоритм работы приложения. Приведен пример организации процесса автоматизации визуального тестирования сайтов с гибкой архитектурой с использованием Applitools, а также создания тестового фреймворка. В заключение, было подробно расписано руководство по развертыванию и запуску тестов.

В приложении представлены публикации и графический материал в виде презентации.

Общий объем диссертационной работы составляет 68 страницы. Из них 57 страниц основного текста, 20 иллюстраций, библиографический список из 24 наименований, список собственных публикаций соискателя из 3 наименований, 3 приложения.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении рассмотрено современное состояние проблемы организации процесса автоматизации тестирования сайтов с гибкой архитектурой, определены основные цели и задачи исследований, определены основные направления исследований, а также дается обоснование актуальности темы диссертации.

В общей характеристике работы показана актуальность проводимых исследований, степень разработанности проблемы, сформулированы цель и задачи диссертации, обозначена область исследований, научная (теоретическая и практическая) значимость исследований, а также апробация работы.

В первой главе тестирование как инструмент проверки качества программного обеспечения.

Тестирование программного обеспечения − процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

Для лучшего понимания того, как тестирование соотносится с программированием и иными видами проектной деятельности, в работе рассматриваются основы − модели разработки ПО (как часть жизненного цикла ПО). При этом сразу известно, что разработка ПО является лишь частью жизненного цикла ПО, и здесь мы говорим именно о разработке.

Выбор модели разработки ПО серьёзно влияет на процесс тестирования, определяя выбор стратегии, расписание, необходимые ресурсы и так далее.

Моделей разработки ПО много, но в общем случае классическими можно считать водопадную, v-образную, итерационную инкрементальную, спиральную и гибкую.

Сайты с гибкой архитектурой – сайты, построенные на основе CMS (англ. Content management system). Проще говоря, CMS − компьютерная программа или информационная система, которая используется для организации и обеспечения процесса по совместному созданию, управлению и редактированию содержимого сайта.

Важно понимать, что сайта как набора страниц при такой схеме просто не существует. Есть отдельно дизайн (шаблоны) и отдельно набор различных материалов - текст, картинки, файлы с архивами, документы MSOffice/PDF и другие материалы. CMS создает страницу пользователю в момент его запроса. При этом, в зависимости от ситуации, пользователю может быть показана какая-то уникальная информация, которая больше никому и никогда не будет видна. Например, содержимое его корзины в интернет-магазине. Эту работу и делает CMS, или "движок сайта".

Исходя из вышеперечисленного, становится понятно, что сайты с гибкой архитектурой достаточно легко конфигурируются и настраиваются, делается это часто и в большом объеме, поскольку любое изменение требований со стороны заказчика может было достаточно быстро применено к текущей версии сайта. Соответственно, изменения происходят одно за другим, и, зачастую эти изменения не касаются внутреннего наполнения, содержимого сайта, а только лишь внешней оболочки, визуального представления.

С одной стороны, можно предположить, что оболочка не столь важна, и какое имеет значение, к примеру, расположение кнопки в правом верхнем углу либо же в левом нижнем. Однако, когда речь идет о глобальной перестройке структуры всей страницы, это неминуемо повлечет за собой определенно изменение последовательности связей между элементами сайта.

Другая не менее важная проблема, встающая, как правило перед огромными корпорациями, представляющими свою продукцию на мировом рынке, это поддержка линейки сайтов со сходными наполнением, имеющим незначительные отличия для различных стран/регионов.

Примеров таких корпораций могут послужить, скажем, Cola-Cola и Nescafe. Несмотря на всю схожесть ассортимента, он будет отличаться от страны к стране: бургеры, представленные на сайте в Соединенных Штатах Америки будут отличаться (пусть и несущественно), от представленных в Беларуси или, скажем, в Японии. К этой же категории нужно отнести проблему восприятия различных языков, когда картинка на сайте одна и та же, расположение полей и кнопок одинаково, однако текст выглядит абсолютно по-разному. Становится крайне экономически невыгодным нанимать команду тестировщиков для каждой страны мира, что повлечет за собой весьма и весьма существенные затраты, в то время как работа, выполняемая командами будет весьма схожей. Примеры подобных отличий для сайтов разных стран удобнее всего наблюдать, разместив их непосредственно друг за другом (рисунок 1)

Рисунок 1 – Пример сайта с гибкой архитектурой

Рассмотрим особенности визуального тестирования сайтов с гибкой архитектурой.

Визуальное тестирование − это проверка того, насколько корректно графический интерфейс приложения отображается пользователю. Данный вид тестирования проводится с целью обнаружения визуальных расхождений с дизайном (ошибки рендеринга страниц, шрифтов, изображений и др.)

Упрощенный механизм визуального тестирования заключается в следующем: это серия тестов, которые проходят по всему сайту и делают скриншоты различных компонентов, сравнивают их с базовыми скриншотами (принятыми как верные по умолчанию либо же заданные явно командой тестировщиков) и предупреждают вас об изменениях.

Если выразить всю их суть очень кратко, то в первую очередь следует учесть:

1 Затраты времени на ручное выполнение тестов и на выполнение этих же тест-кейсов, но уже с использованием визуального тестирования. Чем ощутимее разница, тем более выгодным представляется введение автоматизации на проекте.

2 Количество повторений выполнения одних и тех же тест-кейсов. Чем оно больше, тем больше времени можно сэкономить за счёт визуального тестирования.

3 Затраты времени на отладку, обновление и поддержку визуальных тест-кейсов. Этот параметр сложнее всего оценить, и именно он представляет наибольшую угрозу успеху введения визуального тестирования, потому здесь для проведения оценки следует привлекать наиболее опытных специалистов.

Наличие в команде соответствующих специалистов и их рабочую загрузку. Визуализацией занимаются самые квалифицированные сотрудники, которые в это время не могут решать иные задачи.

При автоматизации визуального тестирования выделяется 2 основных инструмента, при помощи которых осуществляется поддержка данного процесса. К ним относят облачные платформы устройств, а также непосредственно средства визуализации.

Во второй главе будет рассмотрена технология Applitools как средство тестирования сайтов с гибкой архитектурой процессе тестирования будут учтены все особенности автоматизации такого рода приложений.

Технологии и средства визуального тестирования предоставляют возможность проверки правильного отображения GUI приложения для любых браузеров и устройств на любых платформах.

К наиболее распространенным и применяемым на данный момент технологиям в визуальном тестировании можно отнести PhantomCSS, Ocular, Perfecto Devices и Applitools.

PhantomCSS – это инструмент для Node.js для выполнения визуального регрессионного тестирования.

Ocular – простая Java-утилита, предоставляющая возможность добавлять визуальные сравнения внутри уже существующего WebDriver фреймворка.

Perfecto Devices – облачная платформа, поддерживающая интеграцию с различного рода реальными устройствами и эмуляторами, также предоставляющая возможность сравнения изображений.

Поведение любой системы можно описывать при помощи неограниченного количества средств в зависимости от ситуации и необходимых требований. Начиная с самого простого варианта – словесно, при помощи математических формул или уравнений, алгоритмов, блок-схем и так далее.

BDD (или behavior-driven development) − расширение подхода TDD к разработке и тестированию, при котором особое внимание уделяется поведению системы/модуля в терминах бизнеса(заказчика). Как правило, такие тесты иллюстрируют и тестируют различные сценарии, которые интересны непосредственно клиенту системы. В связи с этим при составлении таких тестов часто используется фреймворки, обладающие синтаксисом, обеспечивающим читаемость тестов не только программистом, но и представителями заказчика.

С помощью BDD тестировать систему (или, как сейчас принято говорить, описывать сценарии взаимодействия) может не только сам программист, пишущий код, но и PM (project manager, руководитель проекта), не разбирающийся в деталях реализации, но хорошо знающий систему с точки зрения пользователя. Для новичков BDD-скрипты − самый простой и натуральный пусть ознакомиться с документацией проекта.

Gherkin − это человеко-читаемый язык для описания поведения системы, который использует отступы чтобы задать структуру документа, (пробелы или символы табуляции). Каждая строчка начинается с одного из ключевых слов и описывает один из шагов. Основное преимущество − тесты могут быть разработаны не обязательно тестировщиками, а людьми которые вообще могут не знать программирования, например бизнес-аналитиками. Также это позволяет показать заказчику какой функционал был покрыт тестами. После получения файла со сценариями тестирования тестировщику остается закрепить за каждой строкой необходимый код.

Пример сценария, написанного при помощи языка Gherkin, приведен на рисунке 2.

```
Функциональность:    Калькулятор
Сценарий:            Сложить два числа
Дано:                Я ввожу число <a>
И:                   Я ввожу число <b>
Когда:               Я нажимаю сложить
То:                  Я получаю <result>
Данные:
| a | b | result |
| 2 | 3 |   5    |
```

Рисунок 2 − Пример сценария для тестирования, написанного на языке Gherkin

Как видно, такой сценарий может понять и написать даже человек, далекий от программирования, что является несомненным преимуществом использования такого подхода.

В третьей главе в контексте организации процесса автоматизации визуального тестирования сайтов с гибкой архитектурой ставится задача создания фреймворка для тестирования приложения, позволяющего повысить эффективность контроля качества программного продукта. Также немаловажна организация непрерывного запуска тестов для детального и исчерпывающего анализа результатов.

Для выполнения поставленной цели были сформулированы следующие задачи:

1 Основываясь на оценке эффективности и производительности, привести обоснование рациональности внедрения автоматизации визуального тестирования для сайтов с гибкой архитектурой.

2 Сформировать инструментарий аналитической части системы, в основу которого будет положен анализ существующих средств визуального и автоматизированного тестирования.

3 Разработать фреймворк для автоматизации визуального тестирования сайтов с гибкой архитектурой на основе технологии Applitools с использованием псевдоязыка Gherkin для описания поведения системы.

В связи с поставленными задачами и с целью корректной работы

фреймворка, необходимо придерживаться следующих требований к системе:

– использование операционных систем семейств Windows;

– для разработки фреймворка использовать язык программирования Java;

– использовать технологию Applitools для визуализации тестирования;

– использовать язык Gherkin для описания поведения системы;

– реализовать решение с использованием паттерна PageObject;

– для запуска тестов, хранения кода и организации тестирования использовать Git;

– хранение тест-кейсов, а также входных значений для них в формате BDD сценариев.

Для начала стоит рассказать про общую структуру фреймворка. При разработке тестовых фреймворков «хорошим тоном» считается использование особого подхода организации элементов в коде – применение Page Object. Это популярный в автоматизированном тестировании шаблон проектирования, который упрощает поддержку написанных тестов и уменьшает количество дублируемого кода. Описывается объектно-ориентированный класс, который выступает интерфейсом страницы тестируемого приложения. Методы данного класса используются в тесте при взаимодействии с элементами пользовательского интерфейса приложения. Большим преимуществом является то, что при изменении дизайна пользовательского интерфейса, нужно исправлять не сами тесты, а только лишь код внутри класса Page Object. Соответственно, все изменения, которые необходимо осуществить для реализации поддержки нового интерфейса, будут сосредоточены в одном месте.

В приложении имеется 4 основных модуля:

– VisualObjectFramework – содержит непосредственно описание страничек и форм приложения, а также методы манипуляции элементами странички и создания скриншотов;

– BasicCore – здесь хранятся различные утилиты и методы, не зависящие от интерфейса, странички;

– TestingScenarios – в данном проекте пишутся непосредственно тесты.

Методы для доступа к элементам десктоп-приложений предоставляет утилита WebDriver. Элементы представляются как property в классе, а WebDriver позволяет наглядно описать элемент по какому-либо свойству (Name, Id, ClassName).

В заключение было приведено руководство по развертыванию системы.

ЗАКЛЮЧЕНИЕ

Основные научные результаты диссертации

По итогам проведенной работы можно сделать выводы:

1 Приведено обоснование рациональности внедрения автоматизации визуального тестирования сайтов с гибкой архитектурой, основанное на оценке эффективности.

2 Составлен инструментарий формирования аналитической части системы, основанный на анализе существующих средств визуального тестирования, технологий тестирования и возможностей непрерывной интеграции.

3 Разработан фреймворк для автоматизации визуального тестирования сайтов с гибкой архитектурой на основе технологии Applitools с последующей интеграцией в системе непрерывного запуска тестов.

По результатам магистерской диссертации сделан ряд докладов, а также публикаций в современных научных изданиях.

СПИСОК ПУБЛИКАЦИЙ СОИСКАТЕЛЯ

1 – А. Ериксонова, У.В. Автоматизированное тестирование мобильных приложений с использованием Gherkin/ У.В. Ериксонова, С.А. Поттосина // Проблемы экономической информатики: материалы 53-й науч. конф. аспирантов, магистрантов и студентов, Минск, 25-30 апр. 2017 г. – Минск: БГУИР, 2016.

2 – А. Ериксонова, У.В. Автоматизация тестирования десктоп-приложения с использованием Applitools / У.В. Ериксонова, С.А. Поттосина // Проблемы информатики: материалы 53-й науч. конф. аспирантов, магистрантов и студентов, Минск, 25-30 апр. 2017 г. – Минск: БГУИР, 2017.

3 – А. Ериксонова, У.В. Автоматизация тестирования мобильных приложений в контексте гибких методологий / У.В. Ериксонова // Современный инновационно-инвестиционный механизм развития национальной экономики: материалы 3-й Международной научн. -практич. конфю, Минск, 26 февр. 2017 г. – Полтава: ПНТУ имени Ю. Кондратюка, 2017

РЭЗЮМЭ

Ерыксонава Ульяна Уладзіміраўна

Аўтаматызацыя тэсціравання дэсктоп-прыкладанняў з выкарыстаннем Aplitools

Ключавыя словы: тэставанне, аўтаматызаванае тэставанне, візуалізацыя.

Мэта працы: аптымізацыя працэсу тэставання шляхам яго аўтаматызацыі і візуалізацыі з выкарыстаннем тэхналогіі Applitools.

Атрыманыя вынікі і іх навізна: грунтуючыся на ацэнцы эфектыўнасці і прадукцыйнасці, прыведзена абгрунтаванне рацыянальнасці ўкаранення аўтаматызацыі візуальнага тэставання для сайтаў з гнуткай архітэктурай. Сфарміраваны інструментар аналітычнай часткі сістэмы, у аснову якога пакладзены аналіз існуючых сродкаў візуальнага і аўтаматызаванага тэставання. Распрацаваны фреймворк для аўтаматызацыі візуальнага тэставання сайтаў з гнуткай архітэктурай на аснове тэхналогіі Applitools з выкарыстаннем псевдоязыка Gherkin для апісання паводзін сістэмы.

Вобласць ужывання: тэставанне.

РЕЗЮМЕ

Ериксонова Ульяна Владимировна

Автоматизация визуального тестирования сайтов с гибкой архитектурой с использованием Applitools

***Ключевые слова*:** тестирование, автоматизированное тестирование, визуализация.

***Цель работы*:** оптимизация процесса тестирования путем его автоматизации и визуализации с использованием технологии Applitools.

***Полученные результаты и их новизна*:** основываясь на оценке эффективности и производительности, приведено обоснование рациональности внедрения автоматизации визуального тестирования для сайтов с гибкой архитектурой. Сформирован инструментарий аналитической части системы, в основу которого положен анализ существующих средств визуального и автоматизированного тестирования. Разработан фреймворк для автоматизации визуального тестирования сайтов с гибкой архитектурой на основе технологии Applitools с использованием псевдоязыка Gherkin для описания поведения системы.

***Область применения*:** тестирование.

SUMMARY

Yeryksonava Ulyana

Automation of desktop application testing using Applitools

***Keywords*:** testing, automated testing, visualization.

***The object of study*:** optimizing the testing process through automation and visualization using the technology of Applitools.

***The results and novelty*:** based on the evaluation of efficiency and productivity, the substantiation of the rational automation visual testing for websites with a flexible architecture. Generated analytical instrumentation part of the system, which is based on the analysis of existing visual and automated testing. Developed framework for automation visual testing of websites with a flexible architecture based on the technology Applitools using pseudo Gherkin to describe the behavior of the system.

Sphere of application: testing.