

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК** |

**Кафедра компьютерных систем**

Садаев Федор Андреевич

Инструменты автоматизации тестирования

|  |
| --- |
|  |

**КУРСОВАЯ РАБОТА**

по направлению подготовки 09.04.02 «Информационные системы и технологии», магистерская программа «Большие данные и облачные технологии»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | Студент гр. М8118 | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | | |  |  | | | | подпись |
|  | | |  | Руководитель: к.ф.-м.н.,  доцент кафедры  компьютерных систем  ШЕН ДВФУ | | | | |
|  | | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | В.Ю. Капитан | |
|  | | |  |  | | | | |
| Регистрационный № | | \_\_\_\_\_\_\_\_\_\_\_ |  | Оценка | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
| \_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |  | \_\_\_\_\_\_\_\_\_\_\_ | | В.Ю. Капитан | | |
| подпись | И.О. Фамилия | |  | подпись | | И.О. Фамилия | | |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | | |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | | | | |

**г. Владивосток**

**2020**

**Оглавление**

[1. Введение 3](#_Toc30101475)

[2. Применение автоматизированного тестирования 4](#_Toc30101476)

[3. Преимущества и недостатки автоматизированного тестирования 5](#_Toc30101477)

[4. Инструменты автоматизации тестирования 7](#_Toc30101478)

[4.1 Selenium 7](#_Toc30101479)

[4.2 Katalon Studio 8](#_Toc30101480)

[4.3 UFT 8](#_Toc30101481)

[4.4 Watir 9](#_Toc30101482)

[4.5 IBM Rational Functional Tester 9](#_Toc30101483)

[4.6 TestComplete 10](#_Toc30101484)

[4.7 TestPlant eggPlant 10](#_Toc30101485)

[4.8 Tricentis Tosca 11](#_Toc30101486)

[4.9 Ranorex 11](#_Toc30101487)

[4.10 Robot framework 12](#_Toc30101488)

[5. Заключение 13](#_Toc30101489)

[Список используемых источников 14](#_Toc30101490)

1. Введение

Под тестированием принято понимать деятельность, выполняемую для оценки и улучшения качества ПО. В общем случае тестирование базируется на обнаружении дефектов и проблем в программных системах [1].

Автоматизированное тестирование ПО — процесс тестирования программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически с помощью инструментов для автоматизированного тестирования.

В свою очередь, инструмент для автоматизированного тестирования — это программное обеспечение, посредством которого осуществляется создание, отладка, выполнение и анализ результатов прогона тест-скриптов (Test Scripts — это наборы инструкций для автоматической проверки определенной части программного обеспечения).

Тестирование программных систем состоит из динамической верификации поведения программ на конечном наборе тестов. При этом тесты выбираются из обычно выполняемых действий прикладной области и обеспечивают проверку соответствия ожидаемому поведению системы.

1. Применение автоматизированного тестирования

Первым пунктом в этом списке стоит тестирование производительности. Нагрузочное, стрессоустойчивое, тестирование на стабильность. Без автоматизации его выполнение трудно себе представить. По этой причине имеется широкий выбор продуктов от разных производителей и столь же высокие цены, даже в случае неудобного и слабо функционального инструмента [2].

Следом идёт регрессионное тестирование. Означает оно проверку ПО на корректность функциональности, выпущенной и протестированной в предыдущей версии. Выполняется с регулярной частотой, задаваемой в зависимости от условий: у кого-то с каждым новым билдом, а у кого-то с каждой версией для заказчика.

Конфигурационное тестирование – выполнение одних и тех же тестов в разных условиях. То есть, когда один или несколько компонентов архитектуры системы требуется проверить в разном окружении, обычно заявленном в изначальных требованиях. Например: поддержка СУБД от разных производителей, работа в разных клиентских браузерах, использование в нескольких ОС и т.п. То есть некий аналог регрессионного тестирования, но в рамках одной версии системы.

Функциональное тестирование. Ясно, что здесь речь идёт о проверке нового функционала. Иногда бывает, что без автоматизации никак не обойтись. Даже если нужно выполнить тестирование только один раз. Обычно, впоследствии эти тесты и используются для регресса [3].

Установочное тестирование, выполняется для проверки условий инсталляции (и настройки) продукта с учётом тех или иных требований к системе от заказчика.

1. Преимущества и недостатки автоматизированного тестирования

Преимущества:

* Исключен «человеческий фактор». Сильное достоинство. Все мы люди и никто из нас не застрахован от ошибок. Выполняемый же тест-скрипт не пропустит тест по неосторожности и ничего не напутает в результатах.
* Быстрое выполнение – автоматизированному скрипту не нужно сверяться с инструкциями и документациями.
* Меньшие затраты на поддержку – когда скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время чем на проведение того же объема тестирования вручную.
* Отчеты – автоматически рассылаемые и сохраняемые отчеты о результатах тестирования.
* Выполнение без вмешательства – во время выполнения тестов инженер-тестировщик может заниматься другими полезными делами, или тесты могут выполняться в нерабочее время.

Недостатки:

* Повторяемость – все написанные тесты всегда будут выполняться однообразно. Это одновременно является и недостатком, и преимуществом, так как тестировщик, выполняя тест вручную, может обратить внимание на некоторые детали и найти возникший дефект. Скрипт этого, увы, сделать не может.
* Затраты на поддержку – чем чаще изменяется приложение, тем они выше.
* Большие затраты на разработку – разработка автоматизированных тестов — это сложный процесс, так как фактически идет разработка приложения, которое тестирует другое приложение.
* Стоимость инструмента для автоматизации – в случае, если используется лицензионное ПО, его стоимость может быть достаточно высока. Свободно распространяемые инструменты, как правило, отличаются более скромным функционалом и меньшим удобством работы.
* Пропуск мелких ошибок — автоматический скрипт может пропускать мелкие ошибки, на проверку которых он не запрограммирован.

1. Инструменты автоматизации тестирования

Для автоматизации тестирования имеется множество разнообразных инструментов [4], ниже я опишу наиболее известные.

## Selenium

Считается, что Selenium является самым популярным фреймворком с открытым исходным кодом [5], предназначенным для автоматизации тестирования веб-приложений. Разработанный где-то в далеких двухтысячных и развивающийся в течение последующего десятилетия, Selenium представляют собой фреймворк для автоматизации процесса тестирования веб-приложений. За это время он успел завоевать сердца многих тестировщиков, особенно тех, у кого в распоряжении имеются продвинутые навыки программирования и опыт написания скриптов. Selenium можно рассматривать в качестве родоначальника некоторых современных инструментов автоматизации тестирования с открытым исходным кодом, например: Katalon Studio, Watir, Protractor и Robot Framework.

Фреймворк Selenium поддерживается несколькими ОС (Windows, Mac, Linux), а также многими браузерами (Chrome, Firefox, IE, и браузерами Headless). Скрипты для данного фрейма можно написать на большинстве популярных сегодня ЯП: Java, Groovy, Python, C#, PHP, Ruby и Perl.

Однако стоит отметить, что данный фреймворк имеет как плюсы, так и минусы. К преимуществам можно отнести гибкость, а также возможность написания сложных и эффективных скриптов для тестирования разрабатываемых приложений. С другой стороны, для того чтобы начать работать с Selenium, тестировщик должен обладать незаурядными знаниями в программировании и быть готовым уделять некоторое количество своего времени и энергии для написания специальных фреймов и библиотек, обеспечивающих выполнение определенных функций в процессе тестирования.

## 4.2 Katalon Studio

Katalon Studio — это эффективный инструмент для автоматизации процесса тестирования веб-приложений, мобильных приложений и веб-сервисов. Katalon Studio является потомком таких фреймворков, как Selenium и Appium. Он перенял у последних множество преимуществ, связанных с интегрированной автоматизацией тестирования ПО [6].

Для начала работы с данным инструментом вы можете как обладать начальными знаниями в тестировании ПО, так и быть настоящим гуру своего дела. Люди, далекие от программирования, могут с легкостью запустить свой проект по автоматизации тестирования (например, запустив функцию Object Spy для записи тестовых скриптов), а для программистов и опытных тестировщиков Katalon Studio окажется полезным с точки зрения экономии времени при написании новых библиотек и поддержке существующих скриптов.

Katalon Studio может быть интегрирован в CI/CD, он прекрасно работает в связке с популярными инструментами во время тестирования ПО: qTest, JIRA, Jenkins и Git. Для него предусмотрена приятная функция — Katalon Analytics, благодаря которой пользователи получают полное представление о процессе тестирования. Для этого предусмотрены специальные отчеты, которые выводятся на экран пользователей в виде метрики, диаграмм и графиков.

## 4.3 UFT

Unified Functional Testing (с англ. комплексное функциональное решение для тестирования ПО) или UFT – это популярный коммерческий инструмент для функционального тестирования [7]. Он предоставляет полный набор функций для тестирования API, веб-сервисов, а также для тестирования графического интерфейса десктопных, мобильных и веб-приложений на всех существующих платформах. Для данного инструмента предусмотрена расширенная функция распознавания объектов на основе изображений, многоразовые тестовые компоненты и документация по автоматическому тестированию.

UFT использует Visual Basic Scripting Edition, который может пригодиться для записи информации о выполненном тестировании, а также для управления объектами. UFT интегрирован с Mercury Business Process Testing и Mercury Quality Center. Инструмент поддерживает CI с помощью интеграции с инструментами CI, такими как Jenkins.

## 4.4 Watir

Watir — это инструмент с открытым исходным кодом для автоматизации тестирования веб-приложений, использующий библиотеки Ruby. Для Watir предусмотрена возможность кросс-браузерного тестирования в большинстве существующих браузеров: Firefox, Opera, headless-браузеров и IE. Он также поддерживает управляемое данными тестирование и интегрирован с инструментами BBD, такими как: RSpec, Cucumber и Test / Unit.

## 4.5 IBM Rational Functional Tester

IBM RFT — это платформа для управляемого данными тестирования функциональности и регрессии ПО. Она поддерживает возможность тестирования широкого спектра приложений, написанных на различных языках программирования, таких как: .Net, Java, SAP, Flex и Ajax. RFT использует Visual Basic .Net и Java в качестве языков сценариев. RFT имеет уникальную функцию – Storyboard testing. Благодаря ей все связанные с автоматическим тестированием действия пользователей записываются и визуализируются в виде последовательных изображений – скриншотов приложений на различных этапах их тестирования [8].

Еще одной интересной особенностью платформы RFT является возможность ее интеграции с системами управления жизненным циклом приложений IBM Jazz (например, IBM Rational Team Concert и Rational Quality Manager).

## 4.6 TestComplete

TestComplete, созданный SmartBear, является эффективным инструментом для тестирования десктопных, мобильных и веб-приложений. TestComplete поддерживает различные языки сценариев, такие как: JavaScript, VBScript, Python и C ++ Script. Также, как и в случае с рассмотренным ранее Katalon Studio, с помощью TestComplete тестировщики могут выполнять тестирование с использованием ключевых слов и управляемое данными тестирование. В инструменте также предусмотрена удобная функция записи и воспроизведения процесса тестирования.

TestComplete обладает схожей с UTF функцией распознавания объектов GUI, благодаря которой происходит автоматическое обнаружение и обновление объектов пользовательского интерфейса, что помогает избежать лишних хлопот по поддержанию тестовых скриптов при изменении AUT. Данный инструмент также интегрируется с Jenkins в течение CI-процесса.

## 4.7 TestPlant eggPlant

TestPlant eggPlant – это инструмент автоматического функционального тестирования, основанный на анализе изображений, который позволяет тестировщикам эффективно выполнять AUT. Что касается методов тестирования, то TestPlant eggPlant полностью отличается от традиционных инструментов тестирования: в нем моделирование процесса происходит таким образом, как если бы пользователь занимался тестированием приложений, а не тестировщик, для которого такой процесс заключается в написании тест-скриптов. Такая особенность позволяет тестировщикам, которые не обладают большими знаниями в программировании, применять данный инструмент автоматизации тестирования интуитивно. TestPlant eggPlant поддерживается различными платформами, для него также предусмотрена возможность управления лабораторией и CI-интеграции.

## 4.8 Tricentis Tosca

Tricentis Tosca – это модельно-ориентированный инструмент автоматизации тестирования, который предоставляет довольно широкий набор функций для непрерывного тестирования, включая тестирование с последующим выведением данных, их анализом и интеграцией для поддержки гибких методологий программирования и DevOps-методологий.

Tricentis Tosca помогает пользователям оптимизировать использование ресурсов, необходимых для проведения повторного тестирования. Как в случае с другими подобными инструментами, с помощью Tricentis Tosca можно проводить тестирование веб-приложений, мобильных приложений и API. Для данного инструмента также предусмотрена возможность управления интеграцией, анализа рисков и распределенного исполнения.

## 4.9 Ranorex

Ranorex – это платный универсальный инструмент для автоматизации тестирования веб-, мобильных и десктопных приложений. Инструмент характеризуется расширенными возможностями для распознавания GUI, применением многоразовых тестовых сценариев и возможностью записи/воспроизведения этапов тестирования ПО. Еще одной полезной чертой данного инструмента является возможность создания тестовых сценариев без необходимости писать код. Для тех тестировщиков, кто находится на самом начале пути, такая особенность окажется прекрасным подспорьем: тестировщикам не нужно будет обладать углубленными знаниями в программировании, для того чтобы проводить автоматическое тестирование своих проектов.

Инструмент поддерживает интеграцию Selenium для тестирования веб-приложений. Тестировщики могут группировать определенные результаты тестирований для разных платформ и в различных браузерах с использованием сетки Selenium. Для Ranorex также предусмотрены скидки для бизнес-клиентов.

## 4.10 Robot framework

Robot Framework — это фреймворк для автоматического тестирования с открытым исходным кодом, в котором реализуется подход тестирования на основе ключевых слов для приемочного тестирования и разработки через приемочное тестирование (ATDD). Robot Framework предоставляет возможность решения различных задач автоматизации тестирования. Однако его возможности могут быть расширены благодаря внедрению дополнительных библиотек с помощью Python и Java. Например, Selenium WebDriver — популярная внешняя библиотека, используемая в Robot Framework.

Инженеры-тестировщики могут использовать Robot Framework в качестве фреймворка для автоматического тестирования не только веб-приложений, но для приложений под Android и iOS. Robot Framework покажется простым в изучении тем тестировщикам, кто уже знаком с тестированием на основе ключевых слов.

1. Заключение

В этой работе я рассмотрел основные типы тестирования, преимущества и недостатки автоматизации тестирования и кратко описал основные инструменты для автоматизации тестирования.

У каждого инструмента есть свои плюсы и минусы. Но в большинстве случаев для автоматизации используется Selenium, т.к. он весьма гибкий и масштабируемый. Его можно использовать как для малых проектов, так и для больших проектов разворачивая целую инфраструктуру для автоматизации тестирования.

Также Selenium поддерживает множество языков программирования, что позволяет его интегрировать с разнообразными системами. И использовать не только для автоматизации тестирования, но и для других прикладных задач.

Список используемых источников

1. Савин Р. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапах.— М.: Дело, 2007. — 312 с.

2. Рекс Блек. Ключевые процессы тестирования - М.: Издательство Лори, 2014. - 544 с.

3. Канер Сэм Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: Пер. с англ./Сэм Канер, Джек Фолк, Енг Кек Нгуен. — К.: Издательство «ДиаСофт», 2001. — 544 с.

4. Автоматизация тестирования программных систем [Электронный ресурс] URL: https://habr.com/ru/post/160257/

5. Сравнение инструментов для автоматизированного тестирования [Электронный ресурс] URL: https://www.software-testing.ru/library/testing/testing-automation/2889-automation-testing-tools

6. Тестирование. Фундаментальная теория [Электронный ресурс] URL: https://habr.com/ru/post/279535/

7. Топ 10 инструментов автоматизации тестирования 2018 [Электронный ресурс] URL: https://habr.com/ru/post/342234/

8. 100+ Best Software Testing Tools Reviewed [Электронный ресурс] URL: https://www.qasymphony.com/blog/100-plus-best-software-testing-tools/