Оглавление

[1 Глоссарий 2](#_Toc449171088)

[2 Введение 2](#_Toc449171089)

[3 Аналитическая часть. 2](#_Toc449171090)

[3.1 Введение в предметную область 2](#_Toc449171091)

[3.2 Тестирование программного обеспечения 3](#_Toc449171092)

[3.3 Ручное тестирование 3](#_Toc449171093)

[3.4 Регрессионное тестировние 4](#_Toc449171094)

[3.5 Автоматизированное тестирование 4](#_Toc449171095)

[3.5.1 Selenium webdriver 5](#_Toc449171096)

[3.5.2 JUnit 5](#_Toc449171097)

[3.6 Автоматическое тестирование 5](#_Toc449171098)

[3.6.1 Готовые решения для автоматического тестирования. 6](#_Toc449171099)

[3.6.2 Robotester 6](#_Toc449171100)

[3.6.3 Yandex.Gemini 7](#_Toc449171101)

[3.6.4 FLB 7](#_Toc449171102)

[3.7 Типы ошибок верстки (с картинками) 8](#_Toc449171103)

[3.7.1 Перекрывание блоков одного уровня 8](#_Toc449171104)

[3.7.2 Смещение выровненных элементов 8](#_Toc449171105)

[3.7.3 Выезд элементов за пределы видимости. 9](#_Toc449171106)

[3.7.4 Выезд элементов за пределы видимости 9](#_Toc449171107)

[3.7.5 Ховер-поведение 9](#_Toc449171108)

[3.8 Постановка задачи (что на вход, особенности при тестировании, преимущества, что на выходе) 9](#_Toc449171109)

[3.8.1 Планшетная и мобильная верстки 9](#_Toc449171110)

[3.8.2 Описание бизнес-процесса 9](#_Toc449171111)

[3.8.3 IDEF0 10](#_Toc449171112)

**ЧТО нужно:**

**Список библиографии**

**Написать про ховеры!!!**

**И что там является типом ошибок при ховер-поведении**

# Глоссарий

|  |  |
| --- | --- |
| **Термин** | **Определение** |
| Аккаунт | Совокупность данных, описывающих пользователя |
| Аутентификация | Процесс сопоставления личности пользователя с его учетной записью |
| Авторизация | Процесс назначения прав пользователю, после проведенной им аутентификации |
| Бекенд | Сервер, отвечающий за работу (запись, чтение, обработка, передача) с данными |
| Валидация | Процесс проверки корректности введенных данных |
| Веб-ресурс | Страница или набор страниц, размещенных в сети Интернет, которые могут включать в себя текстовую и графическую информацию |
| Интернет | Всемирная система объединенных компьютерных сетей для хранения и передачи информации |
| Поисковик | Компьютерная система, предназначенная для поиска информации в сети Интернет |
| Сайт | Система электронных документов (файлов данных и кода) частного лица или организации в компьютерной сети под неким адресом |
| Сервис (модуль) | Программа, часть системы, выполняющая определенную функцию |
| Сервер аутентификации | Компьютер, содержащий набор ПО для проведения аутентификации |
| Фронтенд | Сервер, который перенаправляет запросы пользователя на бэкенды, используя их API, и затем возвращает пользователю ответ. Интерфейс между пользователем и бэкендом |

**Принятые сокращения**

|  |  |
| --- | --- |
| Сокращение | Расшифровка |
| URL | Uniform Resource Locator – стандартизированный способ записи адреса ресурса в сети Интернет |
| JSON | JavaScipt Object Notation - текстовый формат обмена данными, основанный на JavaScript |
| API | Application Programming Interface – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением |
| SOA | Cервисно-ориентированная архитектура |
| ПО | Программное обеспечение |
| ОС | Операционная система |
| ИП | Интерфейс пользователя |

# Введение

Данная дипломная работа посвящена одной из самых важных частей процесса разработки программных продуктов – тестированию. В качестве объекта тестирования были выбраны страницы веб-сайта, в частности проверка корректности отображения их интерфейсов в веб-браузере. Выбор обоснован тем, что тестирование корректности верстки является наиболее трудоемкой задачей для тестировщиков. Это обусловлено тем, что в разных веб-браузерах одна и та же страница может отображаться по-разному. Таким образом, одни и те же проверки должны проводиться в разных веб-браузерах, что сильно увеличивает время, необходимое для оценки качества верстки веб-страниц.

Еще одним немаловажным фактором, из-за которого снижается качество ручного тестирования верстки, является необходимость тестировщику быть крайне внимательным в течение долгого времени, выполняя монотонную работу, что неизбежно ведет к усталости, и тестировщик начинает пропускать ошибки.

Целью данной дипломной работы является разработка программного продукта, позволяющего выполнять поиск ошибок верстки на страницах веб-сайтов в автоматическом режиме. Необходимо описать предметную область, рассмотреть и проанализировать имеющиеся решения, описать классы ошибок верстки, разработать алгоритмы их выявления и на их основе разработать программный продукт.

# Аналитическая часть.

## Введение в предметную область

На данный момент количество зарегистрированных доменов в интернете превысило миллиард штук. При этом число активных пользователей сети за последний год увеличилось более чем на двести пятьдесят миллионов [http://www.netcraft.com].

Рынок производства веб-сайтов растет. Существует множество сервисов, существование которых зависит от работоспособности сайта, таких как интернет-доставка, интернет-магазин и другие. Несложно посчитать, что, например, при ста продажах в день неработоспособность сайта в течение часа приводит к потере около четырех сделок. К тому же, пользователи, имевшие негативный опыт в работе с некоторым сервисом, в следующий раз могут выбрать другой, что приводит к потере клиента навсегда.

Таким образом, обеспечение качества программных продуктов, в частности, веб-сайтов, является важной частью процесса их разработки.

## Тестирование программного обеспечения

**Тестирование программного обеспечения** - проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. [IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004]. Основная цель тестирования – оценить качество программного продукта. Тестирование программных продуктов может проводиться в ручном, либо автоматическом режиме.

## Ручное тестирование

При ручном тестировании происходит сравнение человеком ожидаемого и фактического результатов выполнения той или иной функции. На входе тестировщик получает программный продукт и набор требований, которым он должен соответствовать. Рассматривается каждое требование, для них создаются тестовые сценарии, или тест-кейсы - артефакты, описывающие совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части [protesting.ru]. Когда все требования покрыты тест-кейсами, тестировщик выполняет их, и выносит вердикт об актуальном состоянии продукта. При наличии ошибок программный продукт возвращается на доработку, после чего снова подвергается тестированию по уже заготовленным тестовым сценариям. При успешном тестировании программа поступает к пользователям.

Через некоторое время появляются новые требования к программному продукту. Теперь тестировщики должны не только осуществить проверки по ноому функционалу, но и проверить, работоспособен ли функционал, который разрабатывался на предыдущей итерации, то есть провести регрессионные тесты.

## Регрессионное тестировние

Регрессионное тестирование - это вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде [protesting.ru]. По описанному примеру нетрудно понять, что количество регрессионных тестов будет расти с каждой новой версией программного продукта прямо пропорционально количеству новых требований. Соответственно вырастет и время, необходимое тестировщикам для ручной проверки всех тестовых сценариев. Для решения этой проблемы применяется автоматизированное тестирование.

## Автоматизированное тестирование

**Автоматизированное тестирование программного обеспечения** - это процесс верификации программного обеспечения, при котором основные функции и шаги теста, такие как запуск, инициализация, выполнение, анализ и выдача результата, выполняются автоматически при помощи инструментов для автоматизированного тестирования. [http://www.protesting.ru/]

Основными преимуществами автоматизированного тестирования по сравнению с ручным является более высокая скорость выполнения и повышенная, при условии регулярной актуализации, надежность, что исходит их того, что тест-кейсы выполняет машина.

Недостатки автоматизированного тестирования – большие затраты на начальном этапе по сравнению с ручным тестированием, ведь при ручном тестировании для покрытия требования тест-кейсами достаточно их описать, при автоматизированном же тестировании необходимо разработать код, имитирующий действия пользователя и собирающий информацию о результатах выполнения тестовых сценариев. К тому же, в процессе жизненного цикла программного продукта требования к его функциональности могут изменяться, что приводит к необходимости вносить правки в код автоматических тестов, что приводит к дополнительным денежным и временным тратам.

### Selenium webdriver

### JUnit

## Автоматическое тестирование

Проблему необходимости поддерживания автоматизированных тестов в актуальном состоянии решаю автоматические тесты. Основным отличием автоматических тестов от автоматизированных является то, что они не являются уникальными для одного продукта и могут без изменений работать на разных программных продуктах. В таблице 1 систематизированы выкладки по преимуществам и недостаткам рассмотренных видов тестирования.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ручное тестирование | Автоматизированное тестирование | Автоматическое тестирование |
| Ресурсы на внедрение | Низко | Высоко | Низко |
| Ресурсы на поддержку | Низко | Высоко | Не требует поддержки |
| Ресурсы на использование | Высоко | Низко | Низко |

Таблица 1. Преимущества и недостатки видов тестирования

### Готовые решения для автоматического тестирования.

Существует немало сервисов, предоставляющих возможность автоматического нахождения ошибок на веб-страницах. Наибольшего внимания в рамках данной работы заслуживают «Robotester» [], «Yandex.Gemini» [] и «Fighting Layout Bugs». Рассмотрим каждый из них подробнее.

### Robotester

**Robotester –** проект, разработанный в компании Yandex Ильей Кацевым. Его основное достоинство – поисковый робот, который обходит страницы веб-сайта интеллектуально, обходя все состояния всех страниц, на которых можно осуществить различные действия. Имитируются только корректные действия пользователя.

Распознавание элементов на веб-странице осуществляется с помощью Selenium Webdriver []. Анализируются текстовые поля, чтобы определить, какую информацию можно в них внести. К примеру, название города, цену, электронную почту. После определения типов и предназначения текстовых полей происходит генерация тестов. Тесты генерируются на основе заранее заготовленного списка всевозможных корректных данных для ввода в текстовые поля разных типов.

Сами тесты генерируются на основе вероятностного подхода к поиску наиболее корректных действий пользователя. В основе подхода лежит выбор тех действий, которые с наибольшей вероятностью изменят ее текущее состояние. Во время полного обхода веб-страниц собирается статистика, показывающая, какие действия приводят к изменениям. Таким образом, проводя тесты, будут выбираться те действия, которые приводят к изменениям веб-страниц, что убережет нас от действий наподобие многократного ввода логина и пароля и нажатия на отличные от «войти» кнопки.

После выполнения каждого теста идет анализ полученной веб-страницы. Анализируется ее статический код. Идет поиск в его тексте «запрещенных» слов, таких как «error», «ошибка», «404», «exception» и др. При нахождении одного из констатируется факт нахождения ошибки.

Главный недостаток данного подхода заключается в том, что то, что получает конечный пользователь, то есть отображение веб-страницы в браузере, не подвергается проверкам. То есть нечитабельная страница, содержащая ошибки верстки, без ключевых «запрещенных» слов будет идентифицирована как корректная.

### Yandex.Gemini

Недостатки.

### FLB

Опишу после установки

Недостатки

## Типы ошибок верстки (с картинками)

### Перекрывание блоков одного уровня

Суть данной ошибки состоит в том, что видимые части блоков, относящиеся к одному классу, пересекаются друг с другом. Из-за этого пользователь может не видеть часть важного текста, либо необходимая кнопка может оказаться скрытой.

Для определения факта пересечения однородных элементов после эмуляции страницы нужно сравнить области, которые занимают элементы и в случае пересечения их площадей сигнализировать об ошибке.

(ДОПОЛНЕНИЕ ПО ХОДУ РЕАЛИЗАЦИИ, т.к. подробности неизвестны)

### Смещение выровненных элементов

Часто пользователи испытывают необходимость открыть несколько окон с разными программами на одном экране. Для этого они уменьшают ширину окна программы. Наиболее часто встречающаяся ошибка, возникающая при этом – смещение выровненных элементов.

Необходимо учесть, что не всегда однородные элементы должны быть выровнены по одной линии, к тому же, их смещение при изменении ширины экрана может быть предусмотрено разработчиками. Поэтому настройки поиска ошибок рассматриваемого типа необходимо вынести в настройки программы. В частности, это количество элементов, с которого мы начинаем предполагать, что однородные элементы должны быть выровненными.

ЛИБО

Когда элемент уехал с линии, мы будем смотреть, с чем он пересекся. Если с чем-то пересекся – сигнализировать об ошибке, если под ним только фон – все ок.

### Выезд элементов за пределы видимости.

При изменении ширины окна браузера, либо при использовании мобильных устройств, часто встречается ошибка, при которой часть элементов располагается за пределами видимости экрана, то есть пользователь теряет доступ к ним. Для обнаружения ошибок этого типа нужно для каждого элемента страницы определить, не выходит ли его часть за пределы окна браузера.

### Выезд элементов за пределы видимости

### Ховер-поведение

Рассмотрим готовые решения для автоматического тестирования ошибок верстки.

И исходя из недостатков – постановка задачи:

## Постановка задачи (что на вход, особенности при тестировании, преимущества, что на выходе)

### Планшетная и мобильная верстки

Согласно статистике [http://www.w3schools.com/] в период с января 2013 года по январь 2016 года доля мобильных устройств в количестве просмотров страниц сети интернет выросло с 2.18% до 5.38%, то есть более чем в два раза. Мобильные устройства все глубже проникают в нашу жизнь, поэтому качество верстки страниц сайта для отображения на мобильных устройствах становится все более важным. Основными подходами к разработке верстки, подходящей и для полноэкранного отображения на настольных компьютерах, и на мобильных устройствах, являются адаптивная и устройство-ориентированная верстки.

При использовании адаптивной верстки выбор типа построения веб-страницы зависит от ширины экрана. При устройство-ориентированной верстке анализируется браузер, с которого просматривается страница, и в зависимости от этого формируется тем или иным образом. Целью адаптивного веб-дизайна является универсальность отображения содержимого веб-сайта для различных устройств. Для того, чтобы веб-сайт был удобно просматриваемым с устройств форматов и с экранами различных разрешений, по технологии адаптивного веб-дизайна не нужно создавать отдельные версии веб-сайта для отдельных видов устройств. Один сайт может работать на смартфоне, планшете, ноутбуке и телевизоре с выходом в интернет, то есть на всем спектре устройств[Marcotte, Ethan Responsive web design. A List Apart (25 мая 2010)].

Основное разделение по типам верстки – ориентированная на просмотр на планшетной компьютере, на смартфоне и на настольном компьютере. Поэтому эмуляция отображения страниц при тестировании веб-сайтов будет осуществляться для каждого из указанных выше типов верстки.

**Описание бизнес-процесса**

Выбранные типы ошибок – наиболее частые в планшетной и мобильной верстке**.** Также существенным отличием мобильной верстки от настольной является большое количество выпадающих списков. Поэтому анализ страниц будет также проводиться при открытых выпадающих списках.

Также будет проверяться поведение элемента, когда курсор мыши наведен на него. Зачастую в таком состоянии текст кнопки становится слабо различимым на ее фоне.

На вход программа получает ссылку на веб-страницу. За эмуляцию html-страницы отвечает Selenium Webdriver []. На сэмулированной странице происходит анализ элементов по описанным в разделе «Типы ошибок» алгоритмам. Далее та же страница эмулируется на другом разрешении и с ней проводятся аналогичные манипуляции. Необходимые разрешения для теста можно будет выбрать в настройках программы. По окончании анализа генерируется отчет с ошибками и их скриншотами.

### IDEF0

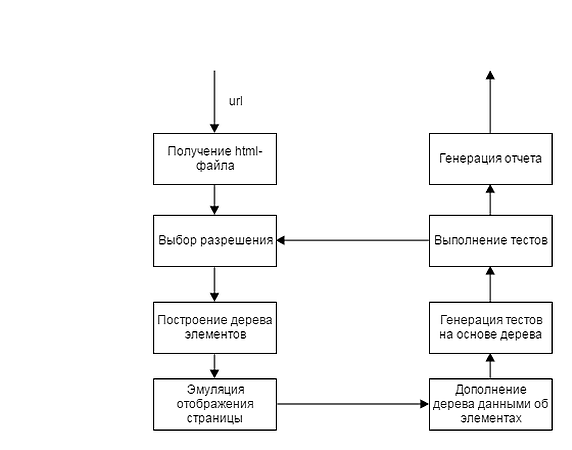
****

Рисунок 3.8.2 – idef0 диаграмма работы программы