Содержание

[ВВЕДЕНИЕ 2](#_Toc185211124)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫЕ И АСПЕКТЫ ТЕСТИРОВАНИЯ 2](#_Toc185211125)

[1.1. О жизненном цикле ПО 2](#_Toc185211126)

[1.2. Определение тестирования и его видов 2](#_Toc185211127)

[1.3. Многослойная архитектура ПО 2](#_Toc185211128)

[1.4. Основные типы тестирования [3] 2](#_Toc185211129)

[1.5. Виды тестирования [4] 2](#_Toc185211130)

[1.6. Значение тестирования и автоматизированного тестирования в разработке 2](#_Toc185211131)

[*1.6.1* *Почему автоматизирование тестирования важно* 2](#_Toc185211132)

[*1.6.2.* *Типы автоматизированных тестов* 2](#_Toc185211133)

[1.7. Цикл тестирования ПО 2](#_Toc185211134)

[1.8. Задача тестировщика 2](#_Toc185211135)

[1.9. Основные понятия 2](#_Toc185211136)

[Выводы по первой главе 2](#_Toc185211137)

[ГЛАВА 2. УСТАНОВКА НЕОБХОДИМЫХ УТИЛИТ И РАЗРАБОТКА АВТОМАТИЗИРОВАННЫХ ТЕСТОВ 2](#_Toc185211138)

[2.1. Описание использованных программных продуктов 2](#_Toc185211139)

[*2.1.1. Selenium* 2](#_Toc185211140)

[*2.1.2.* *Архитектуры* 2](#_Toc185211141)

[*2.1.3.* *Стек Selenium* 2](#_Toc185211142)

[*2.1.4.* *Браузер FireFox* 2](#_Toc185211143)

[*2.1.5.* *Пакеты и расширения для Visual Studio* 2](#_Toc185211144)

[2.2. Интерфейсы 2](#_Toc185211145)

[*2.2.1. Selenium IDE* 2](#_Toc185211146)

[*2.2.2.* *Интерфейс Visual Studio 2013* 2](#_Toc185211147)

[2.3. Возможности Selenium какого рода тесты создаются этим продуктом 2](#_Toc185211148)

[2.4. Техническое задание 2](#_Toc185211149)

[2.10. Практическая часть 2](#_Toc185211150)

[*2.10.2.* *Работа в Visual Studio 2013* 2](#_Toc185211151)

[*2.10.3.* *Демонстрация корректного выполнения написанных тестов* 2](#_Toc185211152)

[*2.10.4.* *Диаграмма IDEF0 «Модель написания тестов»* 2](#_Toc185211153)

[Вывод по второй главе 2](#_Toc185211154)

[ЗАКЛЮЧЕНИЕ 2](#_Toc185211155)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 2](#_Toc185211156)

[ПРИЛОЖЕНИЯ 2](#_Toc185211157)

# ВВЕДЕНИЕ

Разработка и сопровождение программного продукта — это ресурсоемкий процесс, который зависит от множества факторов как технических, так и практических. Создание моделей и введение подходов разработки ПО – искусство, которое предает стройность и эффективность разработке и позволяет избежать множества денежных убытков. Одним из основных факторов качества ПО является процесс тестирования. От корректного введения и подборки корректной модели процесса тестирования в разработку зависит:

* качество ПП;
* время разработки;
* оценка трудозатрат на разработку;
* общее время на разработку;
* промежутка времени до следующего проекта;
* настрой и работоспособность разработчиков;

**Актуальность**. Данная работа актуальная, т.к. такие технологии востребована на рынке и используется в различных компаниях. Российское сообщество тестировщиков присоединившееся к разработчикам, которые осуществляют перевод документации и обмен опытом.

**Цель работы.** Основной целью является демонстрация создания автоматизированных тестовых сценариев на основе стека технологий Selenium.

**Задачи:**

1. Дать общее представление о цикле разработки и тестировании ПО.
2. Перечислить и описать основные видов тестирования.
3. Описать технологии реализации (использованные ПП).
4. Дать представление о внутренней структуре исполнения тестов с помощью Selenium.
5. Описать алгоритм написания теста (ТЗ).
6. Создать тесты на основе функциональных возможностей Yandex.
7. Создать диаграмму «Модель написания тестов» для наглядного представления.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫЕ И АСПЕКТЫ ТЕСТИРОВАНИЯ

# **О жизненном цикле ПО**

Этапы цикла разработки могут варьироваться от проекта к проекту. Это как облегчает, так и привносит определенные затруднения, т.к. от корректного введение и выбора подхода к разработке зависят слаженность, стройность проекта, финансовые и временные затраты. Введение характера проекта зависит от опыта команды и степени зависимости ПО от человеческих жизней, который определяет, то с какой точностью надо описать детальность проекта и уровень формальности документации. Определение того какой подход эффективнее – итеративный или последовательный – либо же совмещение их с определенной степенью соотношения. Влияет на введение тестирования и характер проведения, например, после создания системы или постепенно с появлением программных модулей.

**Цикл разработки ПО:**

1. Разработка проекта продукта.
2. Создание документации (технические задания, спецификации, чеклисты).
3. Кодирование.
4. Исполнение тестирования и ремонт багов.

# Определение тестирования и его видов

**Определение тестирования**

**Цель тестирования**: повышение надежности и ценности программы.

***Тестирование*** – процесс выполнения программы (или части программы) с намерением (или целью) найти ошибки.

***Тестирование*** – процесс, подтверждающий корректность программы и демонстрирующий, что ошибок в программе нет.

***Тестирование*** – фактор успеха, влияющий на качество разрабатываемой системы.

# 1.3. Многослойная архитектура ПО

Тестирование можно определять по архитектуре программы и выделять по модулям, слоям, уровням. Тестирование с помощью Selenium находиться в слое *Presentation layer* пункт *UI components*, это полностью не определяет применение продукта возможно специализированное применение. Для white box тестировщика очень важно знать тонкости реализации системы – это позволит выполнить тестирование

***Слои (layers) = пакеты = пространства имен*** – описывают логическую группировку функциональности в приложении.

***Уровни (tires)*** – описывают физическое распределение функциональности по разным устройствам.

Presentation layer:

1. UI components.
2. Presentation logic components.

Service layer:

1. Service interface.
2. Message Type.

Business layer:

1. Application facade:
   1. Business workflow.
   2. Business components.
   3. Business entities.

External system:

1. Service consumers.

Cross-cutting layer:

1. Security.
2. Operational management.
3. Communication.

Data layer:

1. Data access components.
2. Data helpers / utilities.
3. Service agents.

# **Основные типы тестирования [3]**

***Нисходящее, восходящее*** – так называют инкрементальные подходы, отличаются охватом системы, нисходящее – тестируются модули, восходящее – вся система.

***Статическое, динамическое*** – при статическом программный код вообще не выполняется только посредством анализа листингов логическим способом. Динамический тип подразумевает исполнение программы с вводом данных после исполнения анализируются результаты.

# Виды тестирования [4]

Классификация видов тестирования.

1. По знанию реализации системы:
   1. Черный ящик (black box testing);
   2. Серый ящик (grey box testing);
   3. Белый ящик / стеклянный ящик (white / glass box testing).
2. По объекту тестирования:
   1. Функциональное тестирование (functional testing);
   2. Тестирование UI (UI testing);
   3. Тестирование локализации (localization testing);
   4. Тестирование скорости и надежности (load/stress/performance testing);
   5. Тестирование безопасности (security testing);
   6. Тестирование опыта пользователя (usability testing);
   7. Тестирование совместимости (compatibility testing).
3. По субъекту тестирования:
   1. Альфа-тестирощик (alpha testing);
   2. Бета-тетсировщик (beta testing).
4. По времени проведения тестирования:
   1. До передачи пользователю – альфа-тестирования (alpha testing);
      1. Тест приемки (smoke test, sanity test или confidence test);
      2. Тестирование новых функциональностей (new feature testing);
      3. Регрессивное тестирование (regression testing);
      4. Тест сдачи (acceptance or certification test);
   2. После передачи пользователю – бета-тестирование (beta testing).
5. По критерию «позитивности» сценариев:
   1. Позитивное тестирование (positive testing);
   2. Негативное тестирование (negative testing).
6. По степени изолированности тестируемых компонентов:
   1. Компонентное тестирование (component testing);
   2. Интеграционное тестирование (integration testing);
   3. Системное тестирование (system / end-to-end testing).
7. По степени автоматизированности тестирования:
   1. Ручное тестирование (manual testing);
   2. Автоматизированное тестирование (automated testing);
   3. Смешанное / полуавтомтатизированное тестирование (semi automated testing).
8. По степени подготовки к тестированию:
   1. Тестирование по документации (formal / documented testing);
   2. Эд хок-тестирование (ad hoc testing).

Опишем только некоторые виды, которые специфичны и по названию не раскрывают своей сути.

2.6. ***тестирование опыта пользователя*** – предназначено для оценки и проверки опыта пользователя, которые позволяют решить какой сложности возможно создать интерфейс для понимания пользователем. Основные критерии: интуитивность интерфейса, исходный опыт пользователя.

2.7. ***тестирование совместимости*** – предназначено для проверки как продукт взаимодействует с аппаратными и программными средствами: сервера, пользователь и т.п.

4.1.1. ***тестирование приемки*** – предназначен для проверки основных функциональностей системы.

***Smoke testing*** – минимальный набор тестов на явные ошибки направлено вширь для покрытия максимального функционала системы.

***Sanity testing (Санитарное тестирование или проверка согласованности/исправности)*** – специализированное тестирование определенной функции, направленно вглубь проверяемой функции.

4.1.4. ***тестирование сдачи*** – предназначен для повторной проверки уже оттестированной системы, своего рода подстраховка.

5. позитивное ввод допустимых значений, негативное ввод не валидных значений, например: null; SQL injection; HTML injection; значения, которые исходно не предусматриваются данным полем.

6.2. ***Интеграционное тестирование*** – предназначено для тестирования взаимодействия на уровне нескольких компонентов.

8.2. ***Эд хок-тестирование*** – интуитивное тестирование не по сценариям или документации (что придет в голову).

# Значение тестирования и автоматизированного тестирования в разработке

***Автоматизированное тестирование* –** создание тестов программными средствами, направленное на повышение производительности разработки, для выполнения и проверки результатов выполнения тестовых-сценариев.

# *Почему автоматизирование тестирования важно*

1. ***Regression*** – во время добавления функциональности нужно проверять (тестировать) модули, которые уже были написаны, т.к. некоторые модули могут упасть (ошибка в программном коде, несовместимость), что требует быстрой проверки всего приложения.

В основе регрессионного тестирования лежит идея повторного использования уже созданных тестов.

Существует два вида регрессионного тестирования:

* 1. Тестирование исправленного модуля.
  2. Тестирование всех модулей для предотвращения возникновения ошибок в других частях программы.

1. Время выпуска продукта отображает затраты на его разработку: чем больше времени уходит на разработку продукта, тем стоимость проекта в целом возрастает. Автоматизированные тесты позволяют создать тест один раз и более не тратить время.

Время разработки

Стоимости проекта

Рис. 1. График роста затрат в течение разработки

Таблица 1

**Влияние невыполнения предварительных условий на последовательный и итеративный проекты [1]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Проход 1: последовательный подход без выполнения предварительных условий** | | **Проход2:**  **итеративный подход без выполнения предварительных условий** | |
| Степень завершенности проекта | | Затраты на работу | Затраты на исправление дефектов | Затраты на работу | Затраты на исправление дефектов |
| 1 | 20% | $100 000 | $0 | $100 000 | $75 000 |
| 2 | 40% | $100 000 | $0 | $100 000 | $75 000 |
| 3 | 60% | $100 000 | $0 | $100 000 | $75 000 |
| 4 | 80% | $100 000 | $0 | $100 000 | $75 000 |
| 5 | 100% | $100 000 | $0 | $100 000 | $75 000 |
| Затраты на исправление дефектов в конце проекта | | $0 | $500 000 | $0 | $0 |
| Сумма | | $500 000 | $500 000 | $500 000 | $375 000 |
| Общая сумма | | $1 000 000 |  | $875 000 |  |

Таблица 2

**Влияние выполнения предварительных условий на последовательный и итеративный проекты [1]**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Проход 3: последовательный подход с выполнения предварительных условий** | | **Проход 4:**  **итеративный подход с выполнения предварительных условий** | |
| Степень завершенности проекта | | Затраты на работу | Затраты на исправление дефектов | Затраты на работу | Затраты на исправление дефектов |
| 1 | 20% | $100 000 | $20 000 | $100 000 | $10 000 |
| 2 | 40% | $100 000 | $20 000 | $100 000 | $10 000 |
| 3 | 60% | $100 000 | $20 000 | $100 000 | $10 000 |
| 4 | 80% | $100 000 | $20 000 | $100 000 | $10 000 |
| 5 | 100% | $100 000 | $20 000 | $100 000 | $10 000 |
| Затраты на исправление дефектов в конце проекта | | $0 | $0 | $0 | $0 |
| Сумма | | $500 000 | $100 000 | $500 000 | $50 000 |
| Общая сумма | | $600 000 |  | $550 000 |  |

1. Автоматизированное в сравнении с Документационным.
   1. Документационное тестирование человеком:

Хорошо подходит при нахождении неизвестных случаев.

Эстетика и дизайн.

Полный охват опыта пользователя.

* 1. Автоматизированное тестирование:

Доступно для запуска множество раз

Быстрее в прогоне тестов

Работают в любое время

Механически/логическая проверка

1. Высокоуровневые тесты, тесты, которые воспроизводят в точности действия пользователя.
2. При автоматизации возможно провести тестирование одной функциональности в различных комбинациях по одному клику.
3. ***Exact requirements (требования к ПО)*** – точные строгие требования, составляются на основе запросов – заказчиков, разработчиков, пользователей.

В основном объектом тестирования является функциональные качества системы. За исключением особых видов тестирования (документация, скорость работы).

***Функциональность (functionality, feature)*** – это средство для решения задачи.

# *Типы автоматизированных тестов*

Unit

Integration

UI

Рис. 2. Иерархия автоматизированных тестов

1. ***UI (User Interface)*** – тестирование интерфейса пользователя.
2. ***Integration*** – объединяющее, программные модули (units) объединяются и тестируются вместе.
3. ***Unit*** – модульное тестирование, позволяет изолировать определенный блок и выполнить тест оного.

Чем ближе к верхушке пирамиды, тем больше охват программного кода. Нижние быстрые в выполнении, устойчивы по отношению к другим частям программы. Верхние медленные, более неустойчивы.

# Цикл тестирования ПО

Цикл разработки ПО тесно взаимосвязан с циклом тестирования ПО для ясного понимания процесса тестирования следует абстрагироваться и фокусироваться на рассмотрении подхода и построения гибкой модели к конкретному продукту.

1. Изучение и анализ предмета тестирования.

При изучении задавать себе вопросы:

* 1. Какие функциональности предстоит протестировать?
  2. Как эти функциональности работают?

1. Планирование тестирования постановка задачи.
2. Проектирование, написание тестов.
3. Тестирование тестов.
4. Исполнение тестирования.
5. Изучение результатов тестирования

# Задача тестировщика

Самим тестировщиком может выступать как сам тестировщик, так и разработчик (при тестировании программных модулей), и сам пользователь может являться косвенно при введении продукта в релиз (при обращении в техподдержку) или на бето-версии продукта (при котором тестировщиками могут являться любой свободный сотрудник кампании).

***Цель*** – воспроизвести как можно больше приоритетных багов. Воспроизводить баги тестрировщику необходимо, это видимый измеримый *результат* работы, и чем их больше, тем больше ценим труд тестировщика.

***Покрытие* *[2]*** – это термин, применяющийся в тестировании, который показывает, какие компоненты были затронуты тестовыми сценариями.

Виды:

Покрытие программного кода – какие строки программных кодов выполнялись.

Покрытие постусловий – какие постусловия учитывались при выполнении сценариев.

Покрытие элементов моделей – какие классы и отношения модели были использованы в тестовых сценариях.

**Риски: [2]**

1. Эффективность тестирования доказана временем, что требует введения её на значимые позиции в процессе разработки.
2. Планирование и оценка масштаба разрабатываемой системы требуют аналогичного тестового покрытия. Анализ приоритетных для тестирования частей ПО.
3. В связи с множеством появлений различных моделей систем (расширение модели MVC – MVW, MCRV, MV\* ) стратегии тестирования могут различаться от проекта к проекту, что требует внесения определенных изменений в цели теситрования.

# Основные понятия

Большое развитие тестирования прошло заграницей, как следствие много терминов перешло в практику отечественного тестирования.

1. ***Quality assurance (QA)*** – обеспечение, управление качеством процесса. Предвидение и предотвращение возможных проблем. Так же может применяться – SQA (от англ. software quality assurance).

Обязанности инженеров QA входят такие задачи как подбор: практик, методов, подходов, инструментов решения задач тестирования.

1. ***Quality control (QC)*** – контроль качества, обработка данных в ходе разработки и тестирования с целью улучшения качества продукта.

Обязанности:

* + 1. Улучшение качества продукта.
    2. Анализ результатов тестирования.
    3. Анализ качества сборок и релизов.
    4. Сбор и анализ метрик качества.

1. ***Тестирование (Testing)*** – это процесс достижения ответа на входные данные программный продукт выдавал результаты, соответствующие установленным требованиям.
2. ***Test case priority –*** приоритетность тестируемых ситуаций (приоритетные баги).
3. ***Test-case*** – тестируемая ситуация оболочка для теста.
4. ***Test-suite*** – набор из test-case’ов.
5. ***Valued input*** – действительный ввод (буквы, пробелы – whitespace).
6. ***Unvalued input*** – недействительный ввод (числа, специальные символы).
7. ***Null input*** – пустой ввод. Зависит от спецификации, она определяет, что есть действительный или недействительный ввод.
8. ***Критерий эффективности тестирования*** – это количество багов, найденных до релиза.

**Тестирование и QA**

***Тестирование*** – нахождение багов до релиза.

***QA (quality assurance)*** – это улучшение качества продукта посредством превентирование появления багов при помощи увеличении эффективности процесса разработки. Более широкая область, которая включает в себя как QC так и тестирование.

По стандарту ISO 9126 программный продукт определяется по следующим основным характеристикам:

* Надёжность
* Сопровождаемость
* Практичность
* Эффективность
* Мобильность
* Функциональность

Состав и содержание документации определяется международным стандартом IEEE 829-1998 Standard for Software Test Documentation.

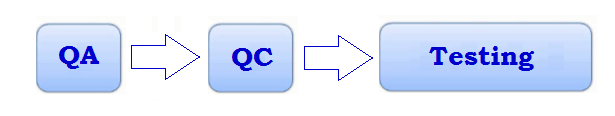


Рис. 3. Схема влияния тестирования c QA и QC

Testing и QA улучшают ПО. Различие заключается в том, что тестирование улучшает ПО через с помощью обнаружения багов, а QA с помощью улучшения процесса разработки.

# Выводы по первой главе

В первой главе были рассмотрена теоретическая часть тестирования, основные типы и виды тестирования. В чем особенность ведения проектов, цикла разработки и какое значение в нем несёт тестирование, а также аспекты автоматизированного тестирования. Разобрано общее представление ведения цикла тестирования ПО. Перечислены особенности видов тестирования. Описаны различия и взаимосвязь quality assurance, quality control и тестирования. В следствии обширной зависимости в множестве аспектов разработки ПО от тестирования, ему следует подходить основательно и подробно анализировать сферу, для которой разрабатывается продукт и детально проработать критерии выполнимости тестирования.

# ГЛАВА 2. УСТАНОВКА НЕОБХОДИМЫХ УТИЛИТ И РАЗРАБОТКА АВТОМАТИЗИРОВАННЫХ ТЕСТОВ

# 2.1. Описание использованных программных продуктов

# *2.1.1. Selenium*

С помощью Selenium предоставляется возможность автоматизированного управления браузерами. Широкий спектр возможностей запуска тестов на различных браузерах и операционных системах. Главная особенность — это моделирование действий пользователя HTTP запросами.

Линейка продуктов с открытым кодом дает множество путей решения задачи автоматизации тестирования, т.к. проект Selenium содержит богатый набор библиотек для браузерных драйверов (Firefox, IE, Safari, iOS, Android; Opera, Chrome) на множестве языков программирования. Официально в проекте состоят такие языки как: C#, Java, Python, JavaScript, Ruby. Помимо сопровождаемых языков есть неофициальные реализации на языках Perl, Haskell, PHP. Данный набор языков позволяет быстро включиться к непосредственно решению задачи.

# *Архитектуры*

Работа архитектуры в системе Selenium WebDrive состоит из трех модулей: зависимости, API (application program interface), движки браузеров. Исполнение теста проходит последовательно с помощью каждого модуля и в результате отправляются HTTP запросы, которые позволяют в точности воспроизвести действия пользователя.

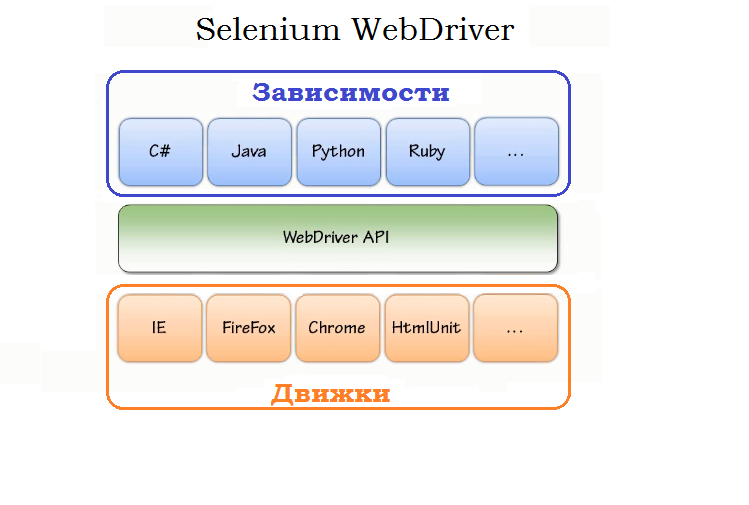


Рис. 4. Архитектура Selenium WebDriver.

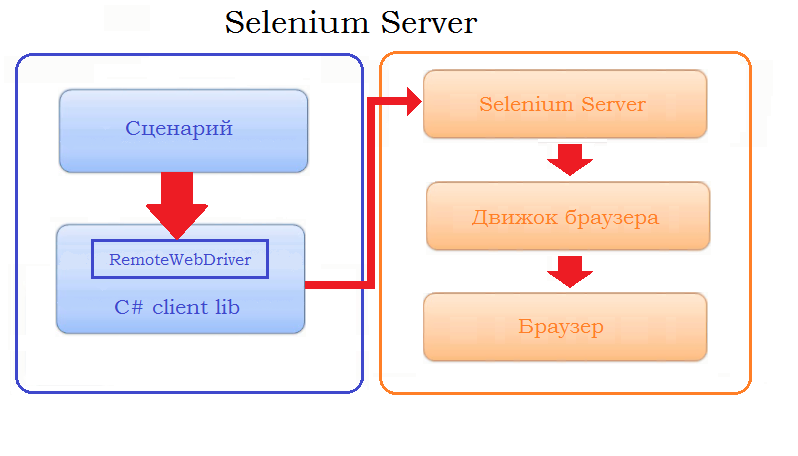


Рис. 5. Архитектура Selenium Server.

# *Стек Selenium*

1. ***Selenium IDE*** – средство первичного создания тестов реализованного в виде плагина браузера. Позволяет составить первичный алгоритм теста, проверить работоспособность через браузер FireFox и перевести заготовку на используемый язык программирования в проекте.
2. ***Selenium Remote Control или WebDrive*** – набор библиотек реализованных на множестве языков. Созданные test suit’ы позволяют автоматизировано воспроизвести баги на сервере для проверки исправности приложения.
3. ***Selenium Server и Grid*** – организуют платформу для воспроизведения массового запуска тестов с использованием множества машин и возможностью распределенного уделенного запуска.

# *Браузер FireFox*

Браузер, предоставляет множество инструментов для веб-разработки.

Плагины:

1. ***Selenium IDE*** – дополнение (плагин) для браузера fireFox, которое(ый) позволяет создавать тесты через веб-интерфейс пользователя с помощью HTTP запросов.
2. ***FireBug*** – универсальный инструмент предоставляет возможности:

2.1. просмотра и редактирования HTML документа;

2.2. возможности визуализировать CSS вёрстку как видит её браузер. Также редактировать и просматривать CSS документ;

2.3. отлаживать и оптимизировать JavaScript код.

2.4. изучат DOM структуру.

2.5. мониторинг активности страницы.

1. ***FirePath и Xpath Checker*** – помощники при написании локатора веб-элемента.
2. Visual Studio Express 2013 по программе DreamSpark Premium. Даёт возможность использовать продукты MicroSoft студентам, аспирантам и преподавателям ВУЗов.

# *Пакеты и расширения для Visual Studio*

1. ***NuGet*** – менеджер, упрощающий работу с библиотеками.
2. ***NUnit*** – фреймворк для модульного тестирования .NET языков.
3. ***Библиотеки RC и WB*** – библиотеки для работы с тестами средствами в данном случае языка C# в среде VS.

# Интерфейсы

# *2.2.1. Selenium IDE*

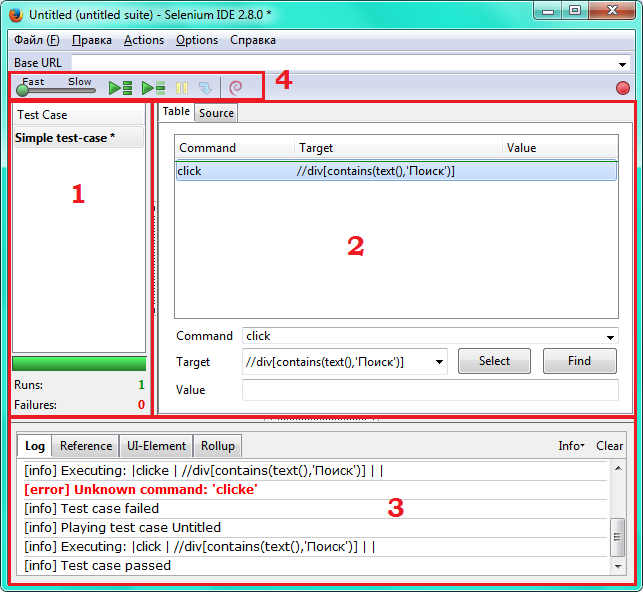


Рис. 6. Интерфейс SIDE.

1. Область обозревателя для управления кейсами.

Нижнее панель сообщает сколько тестов выполнилось корректно, сколько прекратились или в процессе выполнения были ошибки.

1. Основная область создания теста.

В примере одна строка (команда) соответствует одному действию. Тесты выполняются итеративно, команда за командой.

Области ввода ***Command***, ***Target***, ***Value*** – соответствуют одной команде, отвечают за ***действие*** требуется знать языки XPath или CSS (селекторы), ***определение элемента*** в дереве DOM, ***значение*** – несколько видов действий с этим полем самое распространенное данные которые необходимо занести в область ввода.

1. Дополнительной информации:

Вкладки:

***Log*** – журнал регистрации выполненных действий.

***Reference*** – сноска на определение используемой команды.

***UI-E***lement – расширение функций IDE.

***Rollup*** – группировка.

1. Панель настройками запуска:

../_images/chapt3_img06_IDE_features1.png

– запускает все тестовые кейсы из левого меню.



run – запускает один текущий тест.

– пауза.



resume – продолжение выполнение текущего теста.

– при остановке позволяет запустить следующую команду.



rollup – группировка, эта функция группирует повторяющиеся последовательности в одно действие.

record – записывает действия пользователя в браузере.

# *Интерфейс Visual Studio 2013*

1. TestExplorer
2. Пакеты NuGet
3. Selenium Server
4. Установить вкладку Test Explorer.

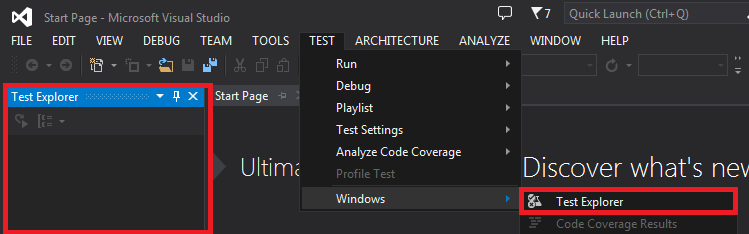
****

Рис. 7. Область Test Explorer.

Вкладка предназначена для отображения состояния прохождения тестов. Используется тестами написанными или переведенными на Remote Control или WebDriver.

1. В вкладке Tools выбрать Extensions and Updates…

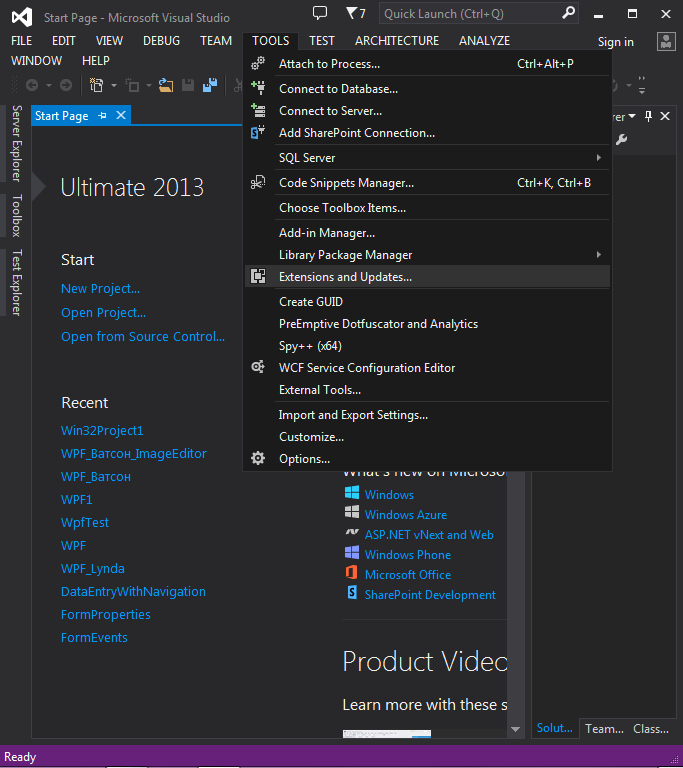


Рис. 8. Расположение утилиты установщика «Расширения и обновления».

Ищем пакеты NuGet и устанавливаем, в дальнейшем надо будет внедрить зависимости в проект.

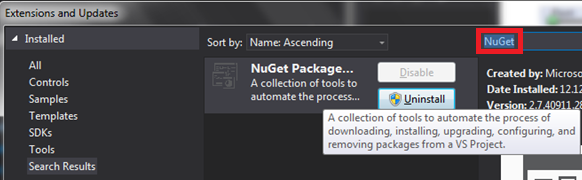


Рис. 9. Порядок установки расширения

Либо проходим по сайту, где можно скачать пакеты.

<http://www.nuget.org/packages/Selenium.RC>

<http://www.nuget.org/packages/Selenium.WebDriver>

1. Selenium Server

<http://docs.seleniumhq.org/download/>

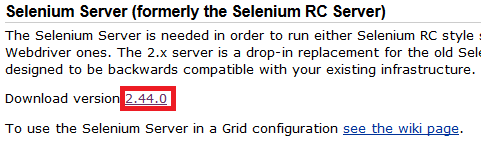


Рис. 10. Установка Selenium Server.

Предназначен только для запуска Selenium RC.

# **Возможности Selenium какого рода тесты создаются этим продуктом**

Selenium RC, WD – чаще всего применяется для регрессионно-функционального тестирования UI.

* + 1. ***Функциональное тестирование***

Позволяет тестировать на уровне пользователя (UI) и автоматизировать интеграцию компонентов на том же UI. Удостоверение, что ПО и их компоненты реализуют необходимые требования на высоком, т.е. верификация системы.

Очень хорошо подходит для завершения вертикально тестирования. Тестирование UI корректно для всей системы и ориентированно на особенности функциональность навигации и вывод ошибок.

Функциональное тестирование плохо подходит для тестирования дизайна (CSS, расположение элементов) и детализированного тестирования всей системы, т.к. не охватывает аспекты кодирования.

* + 1. ***Стили ведения тестирования***

***Физическая объектная модель*** – как объекты расположены на странице.

***Логическая функциональная модель*** – создаются с помощью одного или множества тестов, составляется по задачам, которые должен(должны) решать модуль(модули) ПО, в конечном виде выглядит, как несколько последовательных тестов или с помощью создания методов уже в C# коде. (LFM logical functional model).

# Техническое задание

Разработаем тесты на основе услуг, предоставляемых Yandex. Предварительно составим эвристически несколько сценариев, которые будут выступать в виде обобщенного алгоритма с определенным количеством критериев при прохождении, своего рода техническое задание, оформленное в виде чеклиста.

Всего в test-suit’е будет использоваться четыре аккаунта Yandex почты, которые отсортированы по двум группам: первая – AT\_01, AT\_11; вторая – AT\_21 AT\_22. Это необходимо для создания логики в сценариях.

Таблица 3

**Чеклист проверяемых функций системы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Задание: | Определение: | Необходимо: | Алгоритм: |
|  | Вход/выход | Проверить работу аутентификации. | Один почтовый ящик. | 1. Аутентификация 01. 2. Подтверждение входа. 3. Выход из аккаунта. |
|  | Тестовая отправка письма. | Проверить работу отправки текущему пользователю. | Один почтовый ящик. | 1. Авторизация 01. 2. Отправка письма 01. 3. Проверка прихода. 4. Удаление письма. 5. Выход из аккаунта. |
|  | Проверка прихода почты. | Проверка модуля получения нового письма от 01. | Два почтовых ящика. | 1. Авторизация 01. 2. Отправка письма 11. 3. Проверка отправки. 4. Выход из аккаунта. 5. Авторизация 11. 6. Проверка появления в ленте нового письма. 7. Удаление письма. 8. Выход из аккаунта. |
|  | Переотправка письма другому пользователю. | Реализовать переотправку письма. | Три почтовых ящика. | 1. Авторизация 01. 2. Отправка письма 11. 3. Проверка отправки. 4. Выход из аккаунта. 5. Авторизация 11. 6. Проверка получения. 7. Переотправка письма 21. |

Окончание таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 1. Удаление письма. 2. Выход из аккаунта. 3. Авторизация 21. 4. Проверка прихода письма от 11. 5. Удаление письма.   Выход из аккаунта. |
|  | Отправка документа MS Word. | Предварительно создав документ, реализовать отправку. | 1. Два почтовых ящика. 2. Документ. | 1. Авторизация 01. 2. Отправка документа 11. 3. Проверка отправки. 4. Выход из аккаунта. 5. Авторизация 11. 6. Проверка получения. 7. Удаление письма. 8. Выход из аккаунта. |

# Практическая часть

* + 1. ***Работа в Selenium IDE***

Большую часть процесса создания первичных test-case’ов проходит в браузере Firefox с запушенными плагинами Selenium IDE и при необходимости Firebug.

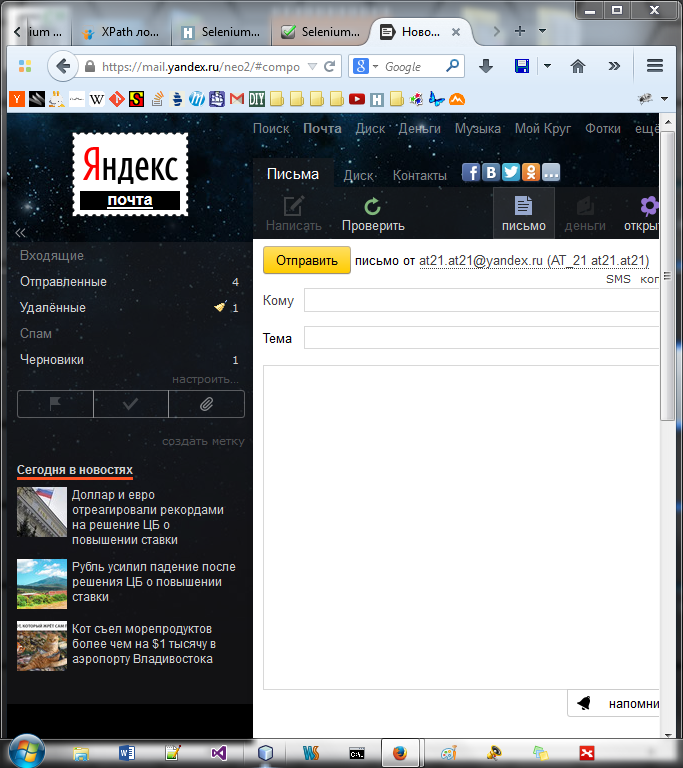


Рис. 11. Левая часть рабочей области

В левой области «Test Case» видно, как отображается корректно написанный тест вы данном случае «1. Вход-выход», и не корректно – «3. Проверка прихода пи…», test case выделенный жирным шрифтом текущий тест алгоритм, которого предоставлен в правой области рисунка.

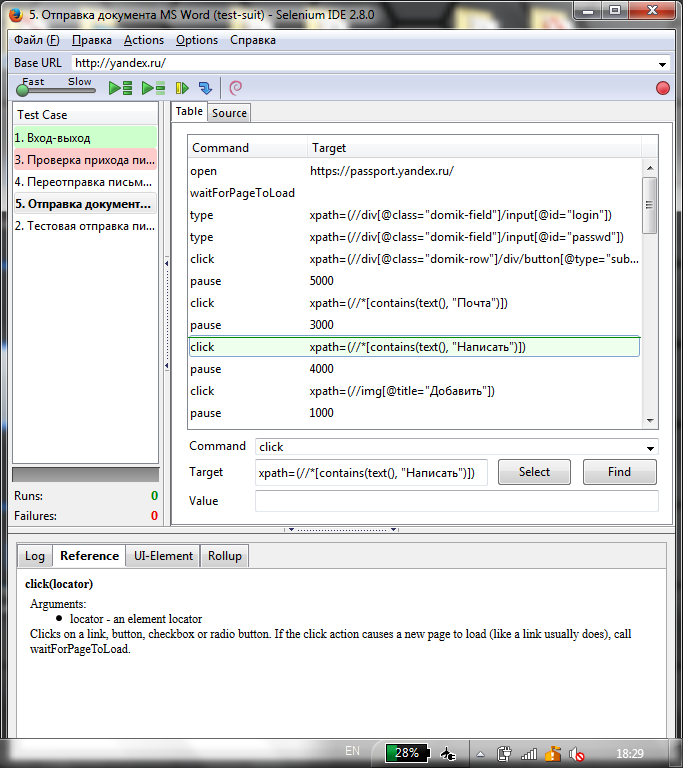


Рис. 12. Правая часть рабочей области

# *Работа в Visual Studio 2013*

В качестве основы при создании проекта следует выбрать проект в категории: Комбинация «Ctrl + Shift + N» – Visual C# – Tests – UnitTestProject.

Во время исполнения и редактирования тестов при запуске тест открывает чистый движок Firefox с журналом событий. Ниже предоставлены рабочие области тестироващика.

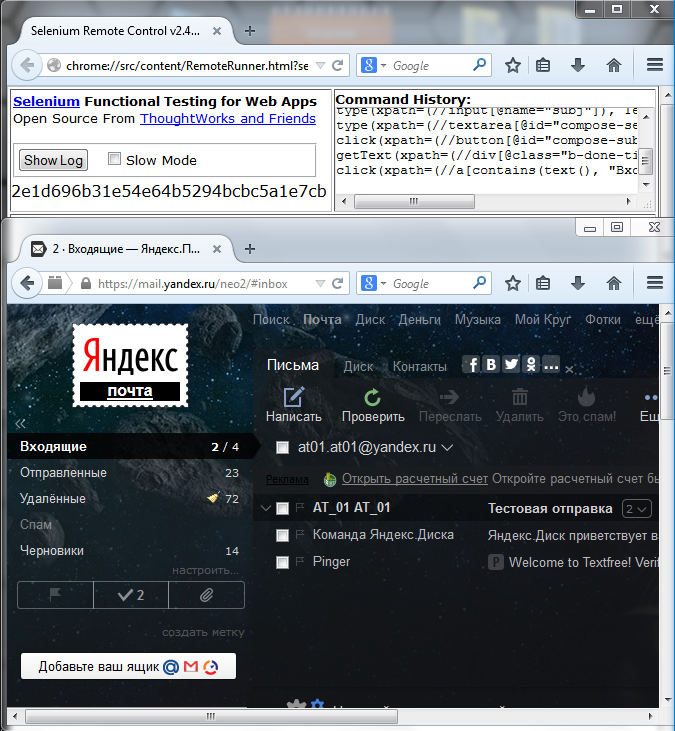


Рис. 13. Левая часть рабочей области

В панель «Test Explorer» демонстрирует состояние тестов на данный момент времени: ***«Failed Tests»*** – провалившиеся тесты, ***«Passed Tests»*** – успешно пройденные тесты и ***«Not Run Tests»*** – тесты которые ещё не были запушены. Выделенный синим цветом тест выполняется на момент снятия скриншота, изображение которого можно наблюдать на предыдущем рисунке.

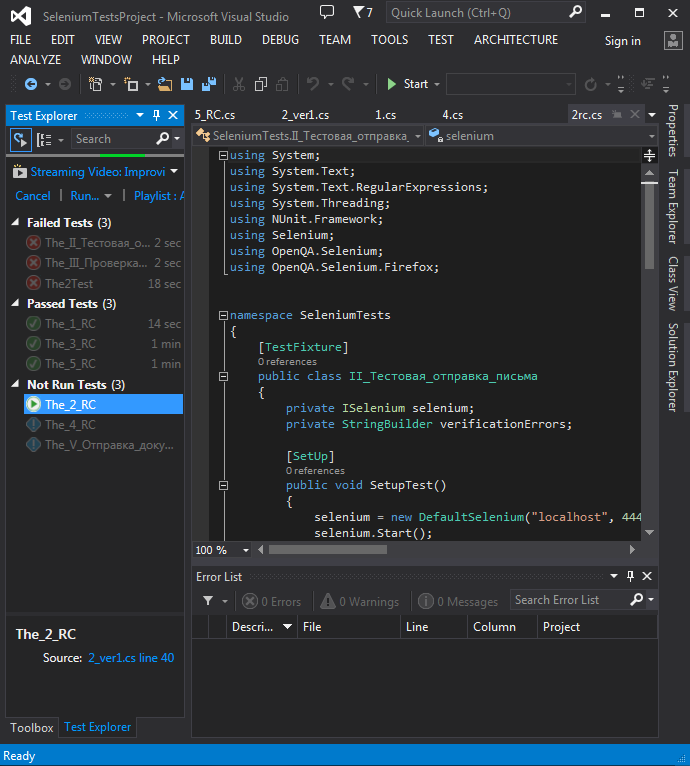


Рис. 14. Правая часть рабочей области

# *Демонстрация корректного выполнения написанных тестов*

Область 1 – также показано напротив теста время выполнения оного.

Область 2 – также есть функция в провалившихся тестах перехода в строку оператора, который не выполнился, в данном случае в тесте – The2Test, время выполнения – 18 сек.

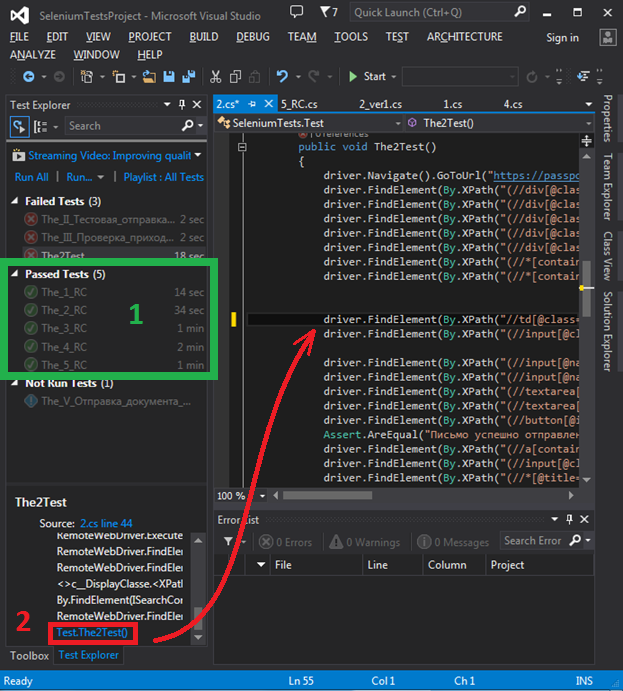


Рис. 15. Корректно выполненные все test-case’ы

# *Диаграмма IDEF0 «Модель написания тестов»*

**Цель:**

Дать основное представление о процессе создания автоматизированных тестов.

**Определение:**

Модель предназначена для отображения, как тесты из эвристического определения задачи проходят через несколько технологий и становятся автоматизированным.

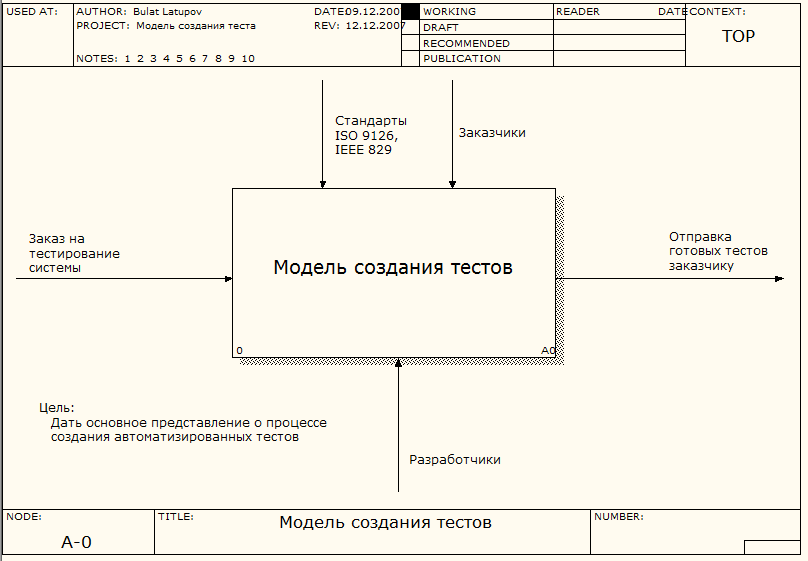


Рис. 16. Родительская диаграмма

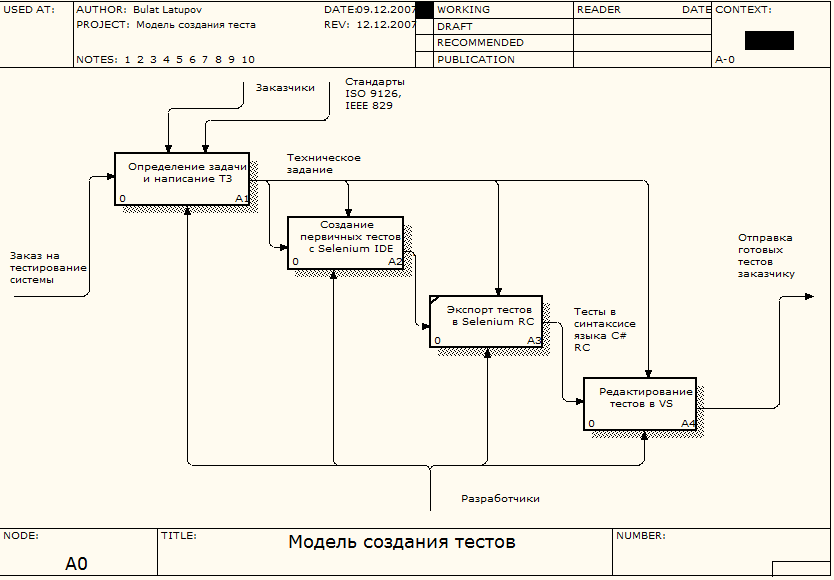


Рис. 17. Первый уровень декомпозиции

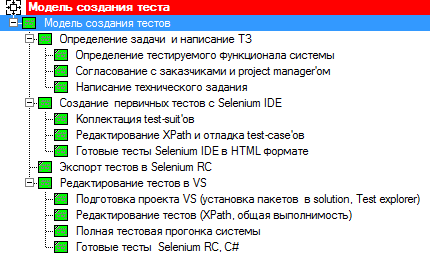


Рис. 18. Иерархическое дерево декомпозиции диаграммы

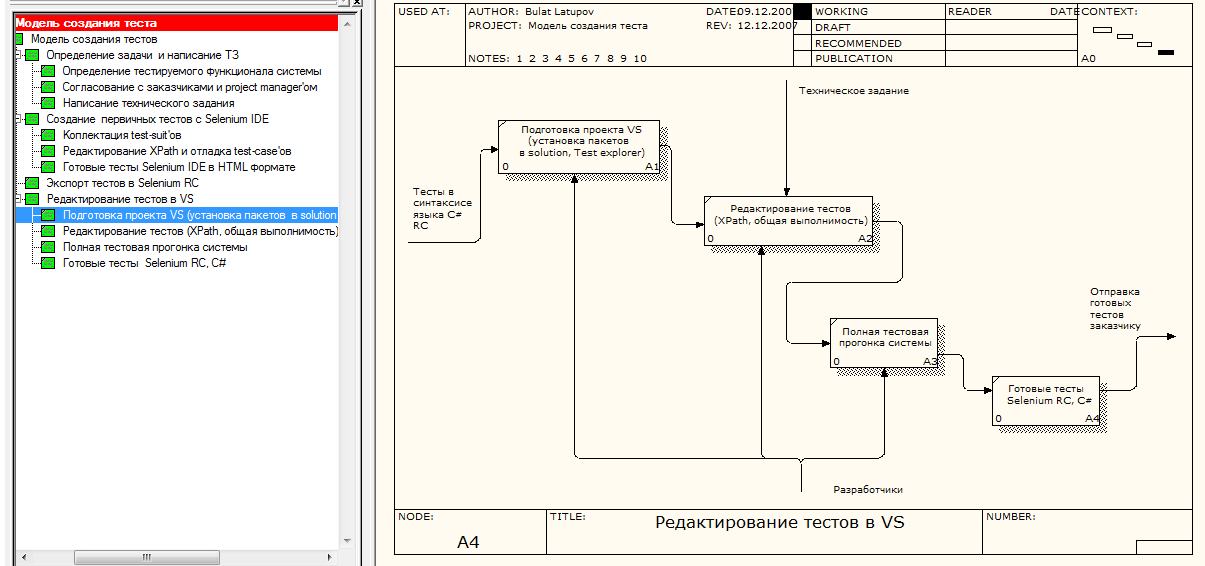


Рис. 19. Второй уровень декомпозиции четвертого блока

# Вывод по второй главе

В начале главы были затронуты все аспекты установки и запуска необходимых утилит и ПП для работы с средой. Продемонстрированы основные механизмы взаимодействия сред с разработчиком

Также в практической части курсовой составлено техническое задание из пяти test-case’ов, которые затрагивают функциональности почтового мессенджера Yandex.mail. Создан и проверен на корректное выполнение test-suit в синтаксисе C#, который дает представление о написании автоматизированных тестов стеком технологий Selenium с использованием в качестве основы модульную среду .NET-язык C#.

Создана диаграмма «Модель написания тестов», которая наглядно показывает основные функции декомпозированных блоков, за какую часть процесса разработки отвечает каждый блок и какие «механизмы» и «управления» имеют значение на определенных уровнях.

# ЗАКЛЮЧЕНИЕ

В связи с тенденцией миграции не только бизнеса, экономики, а и всех аспектов жизни в Интернет. Требования к разработке веб-приложений и темпы их создания возрастают. Так как веб-продукты будут востребованы ещё огромный промежуток времени, как следствие необходимость в поддержке качества и эффективного сопровождения увеличиваются. Эти факты позволяет с уверенностью сказать, что тестирование и поддержание качества ПО с помощью продуктов линейки Selenium является актуальной сферой.

В данной работе были исследованы основные возможности создания автоматизированных тестов с помощью продуктов Selenium. Этот широкий аспект возможностей и размах предоставляемых языков с технологиями может уменьшить срок, как тестирования, так и в целом создания веб-приложения и при грамотном подходе реализовать надлежаще тестирование. Всё это позволяет сказать, что Selenium будет держать позиции в своей нище решения задач автоматизированного тестирования и поддержания качества ПО.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Совершенный код. Стив Макконнелл.
2. Тестирование ОО ПО. Практическое пособие. Джон Макгрегор, Девид Сайкс.
3. Тестирование ПО. Фундаментальные коцепции менеджмента бизнес-приложений. Сэм Канер, Джек Фолк, Енг Кек Нгуен.
4. Роман Савин. Тестирование Дот ком.
5. <http://selenium2.ru/docs/selenium-ide>
6. <http://release.seleniumhq.org/selenium-core/1.0.1/reference.html>
7. <http://docs.seleniumhq.org/download/>
8. <https://code.google.com/p/selenium/>
9. <http://selenium.googlecode.com/git/docs/api/dotnet/index.html>
10. <http://www.nuget.org/>
11. <http://docs.nuget.org/docs/start-here/videos>
12. <http://www.software-testing.ru>
13. <http://www.w3schools.com/xpath/default.asp>
14. <http://internetka.in.ua/>
15. <http://stackoverflow.com>
16. <http://bugscatcher.net/>
17. <http://www.testingexcellence.com>
18. <http://siderulezzz.wordpress.com>
19. <http://www.w3.org/TR/xpath/>

Закончить список литературы, перекрестные ссылки, содержание

# ПРИЛОЖЕНИЯ

**Приложение 1**

Test-case 5 – Отправка документа MS Word.html

<?**xml version="1.0" encoding="UTF-8"**?>  
<!DOCTYPE **html** PUBLIC **"-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"**>  
<**html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"**>  
<**head profile="http://selenium-ide.openqa.org/profiles/test-case"**>  
<**meta http-equiv="Content-Type" content="text/html; charset=UTF-8"** />  
<**link rel="selenium.base" href="http://yandex.ru/"** />  
<**title**>5. Отправка документа MS Word</**title**>  
</**head**>  
<**body**>  
<**table cellpadding="1" cellspacing="1" border="1"**>  
<**thead**>  
<**tr**><**td rowspan="1" colspan="3"**>5. Отправка документа MS Word</**td**></**tr**>  
</**thead**><**tbody**>  
<**tr**>  
 <**td**>open</**td**>  
 <**td**>https://passport.yandex.ru/</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>waitForPageToLoad</**td**>  
 <**td**></**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-field**&quot;**]/input[@id=**&quot;**login**&quot;**])</**td**>  
 <**td**>at21.at21</**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-field**&quot;**]/input[@id=**&quot;**passwd**&quot;**])</**td**>  
 <**td**>a.t2\_1.21</**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-row**&quot;**]/div/button[@type=**&quot;**submit**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>5000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//\*[contains(text(), **&quot;**Почта**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>3000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//\*[contains(text(), **&quot;**Написать**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>4000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//img[@title=**&quot;**Добавить**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>1000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//span[@class=**&quot;**abook-entry-name-content**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>1000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**abook-popup-footer nb-with-l-left-gap nb-with-xs-bottom-gap js-abook-popup-footer**&quot;**]/button)</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>700</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//input[@name=**&quot;**att**&quot;**])</**td**>  
 <**td**>C:\Selenium\document.docx</**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>4000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//input[@name=**&quot;**subj**&quot;**])</**td**>  
 <**td**>Отправка документа MS Word</**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//textarea[@id=**&quot;**compose-send**&quot;**])</**td**>  
 <**td**>Отправка документа MS Word</**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//button[@id=**&quot;**compose-submit**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>4000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>assertText</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**b-done-title**&quot;**])</**td**>  
 <**td**>Письмо успешно отправлено.</**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//a[contains(text(), **&quot;**Входящие**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>7000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//a[@data-metric=**&quot;**Меню сервисов:Выход**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>7000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>open</**td**>  
 <**td**>https://passport.yandex.ru/</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>waitForPageToLoad</**td**>  
 <**td**></**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-field**&quot;**]/input[@id=**&quot;**login**&quot;**])</**td**>  
 <**td**>at22.at22</**td**>  
</**tr**>  
<**tr**>  
 <**td**>type</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-field**&quot;**]/input[@id=**&quot;**passwd**&quot;**])</**td**>  
 <**td**>a.t2\_2.22</**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//div[@class=**&quot;**domik-row**&quot;**]/div/button[@type=**&quot;**submit**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>5000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//\*[contains(text(), **&quot;**Почта**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>3000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>5000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//\*[contains(text(), **&quot;**Отправка документа MS Word**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>700</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>assertTextPresent</**td**>  
 <**td**>at21.at21@yandex.ru</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>700</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>assertTextPresent</**td**>  
 <**td**>document.docx</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//a[contains(text(), **&quot;**Входящие**&quot;**)])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>7000</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//input[@class=**&quot;**b-messages\_\_message\_\_checkbox\_\_input**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//\*[@title=**&quot;**Удалить (Delete)**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>pause</**td**>  
 <**td**>700</**td**>  
 <**td**></**td**>  
</**tr**>  
<**tr**>  
 <**td**>click</**td**>  
 <**td**>xpath=(//a[@data-metric=**&quot;**Меню сервисов:Выход**&quot;**])</**td**>  
 <**td**></**td**>  
</**tr**>  
</**tbody**></**table**>  
</**body**>  
</**html**>

**Приложение 2**

Test-case 5 – Отправка документа MS Word

using System;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading;

using NUnit.Framework;

using Selenium;

namespace SeleniumTests

{

[TestFixture]

public class The\_5\_Отправка\_документа\_MS\_Word

{

private ISelenium selenium;

private StringBuilder verificationErrors;

[SetUp]

public void SetupTest()

{

selenium = new DefaultSelenium("localhost", 4444, "\*firefox C://Program Files (x86)/Mozilla Firefox/firefox.exe", "http://yandex.ru/");

selenium.Start();

verificationErrors = new StringBuilder();

}

[TearDown]

public void TeardownTest()

{

try

{

selenium.Stop();

}

catch (Exception)

{

// Ignore errors if unable to close the browser

}

Assert.AreEqual("", verificationErrors.ToString());

}

[Test]

public void The\_5\_RC()

{

selenium.Open("https://passport.yandex.ru/");

Thread.Sleep(3000);

selenium.Type("xpath=(//div[@class=\"domik-field\"]/input[@id=\"login\"])", "at21.at21");

selenium.Type("xpath=(//div[@class=\"domik-field\"]/input[@id=\"passwd\"])", "a.t2\_1.21");

selenium.Click("xpath=(//div[@class=\"domik-row\"]/div/button[@type=\"submit\"])");

Thread.Sleep(5000);

selenium.Click("xpath=(//\*[contains(text(), \"Почта\")])");

Thread.Sleep(3500);

selenium.Click("xpath=(//\*[contains(text(), \"Написать\")])");

Thread.Sleep(4000);

selenium.Click("xpath=(//img[@title=\"Добавить\"])");

Thread.Sleep(1000);

selenium.Click("xpath=(//span[@class=\"abook-entry-name-content\"])");

Thread.Sleep(1000);

selenium.Click("xpath=(//div[@class=\"abook-popup-footer nb-with-l-left-gap nb-with-xs-bottom-gap js-abook-popup-footer\"]/button)");

Thread.Sleep(700);

selenium.Type("xpath=(//input[@name=\"att\"])", "C:\\Selenium\\document.docx");

Thread.Sleep(4000);

selenium.Type("xpath=(//input[@name=\"subj\"])", "Отправка документа MS Word");

selenium.Type("xpath=(//textarea[@id=\"compose-send\"])", "Отправка документа MS Word");

selenium.Click("xpath=(//button[@id=\"compose-submit\"])");

Thread.Sleep(4000);

Assert.AreEqual("Письмо успешно отправлено.", selenium.GetText("xpath=(//div[@class=\"b-done-title\"])"));

selenium.Click("xpath=(//a[contains(text(), \"Входящие\")])");

Thread.Sleep(7000);

selenium.Click("xpath=(//a[@data-metric=\"Меню сервисов:Выход\"])");

Thread.Sleep(7000);

selenium.Open("https://passport.yandex.ru/");

selenium.WaitForPageToLoad("");

selenium.Type("xpath=(//div[@class=\"domik-field\"]/input[@id=\"login\"])", "at22.at22");

selenium.Type("xpath=(//div[@class=\"domik-field\"]/input[@id=\"passwd\"])", "a.t2\_2.22");

selenium.Click("xpath=(//div[@class=\"domik-row\"]/div/button[@type=\"submit\"])");

Thread.Sleep(5000);

selenium.Click("xpath=(//\*[contains(text(), \"Почта\")])");

Thread.Sleep(3000);

Thread.Sleep(5000);

selenium.Click("xpath=(//\*[contains(text(), \"Отправка документа MS Word\")])");

Thread.Sleep(700);

Assert.IsTrue(selenium.IsTextPresent("at21.at21@yandex.ru"));

Thread.Sleep(700);

Assert.IsTrue(selenium.IsTextPresent("document.docx"));

selenium.Click("xpath=(//a[contains(text(), \"Входящие\")])");

Thread.Sleep(7000);

selenium.Click("xpath=(//input[@class=\"b-messages\_\_message\_\_checkbox\_\_input\"])");

selenium.Click("xpath=(//\*[@title=\"Удалить (Delete)\"])");

Thread.Sleep(700);

selenium.Click("xpath=(//a[@data-metric=\"Меню сервисов:Выход\"])");

}

}

}