Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Безгузиков Артемий Валерьевич

Автоматизация GUI тестирования в проекте WMP

Бакалаврская работа

Научный руководитель:
к.т.н., доц. Брыксин Т.А.

Рецензент:
Смирнов К.К.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Artemii Bezguzikov

Automation of GUI testing in the WMP project

Graduation Thesis

Scientific supervisor:
Candidate of Engineering Sciences Timofey Bryksin

Reviewer:
Kirill Smirnov

Saint-Petersburg
2017

Оглавление

Введение

Во многих проектах помимо функционального и интеграционного тестирования используется и автоматизированное тестирование графического интерфейса, или GUI тестирование (Graphical User Interface). Само по себе GUI тестирование — это имитация действий пользователя. Оно предназначено для проверки выполнения всех основных сценариев работы с приложением. В широком смысле графическое тестирование позволяет:

- выявлять ошибки в функциональности посредством графического интерфейса;
- выявлять необработанные исключения, возникающие при взаимодействии с графическим интерфейсом;
- выявлять некорректные ситуации при работе с выводимыми данными;
- проверять успешность выполнения основных пользовательских сценариев.

На кафедре системного программирования СПбГУ силами студентов и преподавателей разрабатывается проект WMP[1]. Он является аналогом проекта QReal[2] и служит для создания средств визуального программирования на основе построения диаграмм. Ярким примером такой среды, построенной с помощью QReal, является TRIK Studio[3], предназначенная для программирования роботов. В отличие от QReal, WMP и все порождаемые им среды программирования находятся в web пространстве в рамках одной системы. На данном этапе развития WMP включает в себя редактор диаграмм для роботов и BPMN (Business Process Model and Notation) редактор. Первый полностью повторяет

[1]Web Modeling Project — URL: https://github.com/qreal/wmp

[2]Среда для создания новых визуальных языков и интегрированных сред программирования для них — URL: https://github.com/qreal/qreal

[3]Среда программирования роботов — URL: http://blog.trikset.com/p/trik-studio.html

язык TRIK Studio. Платформа WMP предоставляет библиотеки, с помощью которых процесс создания каждого такого редактора существенно упрощается. Учитывая модульность архитектуры, каждый редактор является по сути подключаемым модулем к платформе, то есть легко интегрируемым с ней.

Как и любой другой программный продукт, WMP должен тестироваться, но, в отличие от большинства web проектов, главная функциональность WMP — составление диаграмм. Существуют инструменты GUI тестирования, позволяющие имитировать нажатие на кнопку, ввод текста и другие элементарные действия, но они не подходят для конструктора диаграмм ввиду того, что они слишком низкоуровневые. Необходимы операции более высокого уровня, которые позволят производить элементарные манипуляции с диаграммой в рамках одной команды. Также в силу того, что речь идет об имитации действий пользователя, хорошим следствием применения технологий GUI тестирования было бы создание обучающих анимаций, которые позволили бы упростить введение пользователя в предметную область.

Постановка задачи

На основе имеющихся библиотек GUI тестирования сайтов разработать инфраструктуру для графического тестирования в проекте WMP.

Для достижения этой цели были поставлены следующие задачи.

1. Разработать средства тестирования редактора диаграмм WMP.
2. Разработать средства конфигурирования полученного решения.
3. Реализовать инструменты, позволяющие создавать обучающие анимации.

1. Обзор

1.1. Пользовательский интерфейс проекта WMP

WMP — это web проект, и, как любой другой сайт, он представляется пользователю как набор страниц. В настоящий момент в проекте существуют страницы аутентификации, панели роботов, редактора диаграмм для роботов и BPMN редактора. Чтобы зайти на любую страницу, кроме первой, необходимо авторизироваться. Интерес представляют именно страницы редакторов.

1.1.1. Графический интерфейс редактора диаграмм

Редактор диаграмм для роботов имеет внешний вид, представленный на Рис. 1.

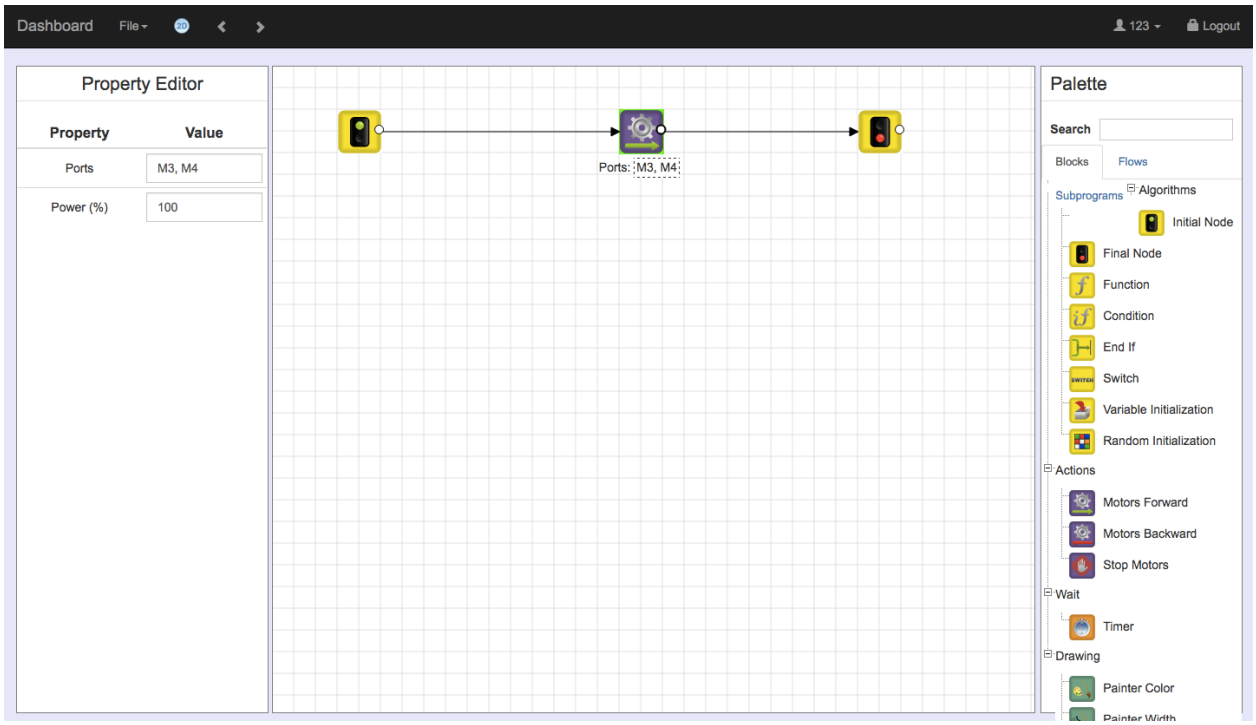

Рис. 1: Редактор диаграмм для роботов проекта WMP.

Ниже перечисляются компоненты редактора.

- Сцена — область экрана на которой размещаются диаграммы, которые состоят из блоков и стрелок, соединяющих некоторые из

них.

- Палитра — область экрана, где расположен контейнер с различными типами блоков. Их можно перемещать на сцену с помощью мыши.

- Редактор свойств — специальная область экрана, на которой отображаются имена свойств и их значения для выделенного в данный момент времени блока диаграммы.

- Верхнее меню — горизонтальное меню в верхней области экрана, используется для сохранения/открытия диаграммы.

Редакторы WMP однотипны, и BPMN редактор также состоит из приведенных компонент.

1.1.2. Критерии выбора инструмента тестирования

Данная работа выполняется в рамках выпускной бакалаврской работы, поэтому выбранный инструмент GUI тестирования должен быть свободно распространяемым.

Для манипуляций с диаграммами необходим поиск элемента, захват элемента и его перемещение. Наличие такой функциональности требуется, чтобы выбрать элемент из палитры и добавить его на сцену. Также это позволит производить элементарные операции над элементами внутри диаграммы.

Исходя из того, что каждый редактор WMP — это среда визуального программирования, к ним необходимо создавать инструменты для обучения. Одним из таких инструментов являются обучающие анимации. Каждая анимация — это записанные действия, которые совершил пользователь, чтобы достигнуть какого-то конкретного результата. Такие анимации можно строить автоматически, и некоторые средства GUI тестирования способны решать эту задачу.

Таким образом, можно выделить следующие критерии.

- Свободное распространение инструмента.

- Наличие функций поиска, захвата и перемещения элемента.
- Доступность визуализации тестирующих скриптов.

1.2. Основные инструменты GUI тестирования

Существуют несколько флагманов среди инструментов автоматизации GUI тестирования, которые обычно используются в проектах. В частности, TestComplete [13], Ranorex Studio [11] и Selenium WebDriver [9, 3]. TestCompile и Ranorex Studio обладают всей необходимой функциональностью, но имеют закрытый исходный код и не являются свободно распространяемыми. Они не отвечают первому критерию и поэтому не подходят.

Также следует выделить Sikuli [10] и SoapUI [12]. Они бесплатны для использования. Sikuli построен на поиске изображений. Для того, чтобы найти какой-либо элемент, необходимо сначала загрузить его картинку. В диаграммах WMP могут содержаться разные элементы с одинаковыми изображениями: можно составить диаграмму из двух действий — два элемента "мотор вперед", соединенных стрелкой. Диаграмма корректна, но неизбежно возникнут сложности с обработкой. SoapUI чаще применяют как инструмент интеграционного тестирования, чем GUI. Он больше ориентирован на тестирование серверной части приложения.

1.3. Selenium WebDriver

Одним из самых популярных средств тестирования GUI является Selenium WebDriver. Он предоставляет удобный низкоуровневый API (Application Programming Interface) для управления браузером, оформлен в виде библиотек, и при тестировании запускает браузер, на котором визуально отображаются все посылаемые команды. WebDriver способен работать со многими браузерами и на разных языках программирования. Он легко интегрируем с другими инструментами, так как представляет из себя набор библиотек. Для данного проекта был

выбран браузер Google Chrome и язык Java, потому что и WMP и сам Selenium написаны на Java.

Принцип работы WebDriver отражает диаграмма, представленная на Рис. 2.

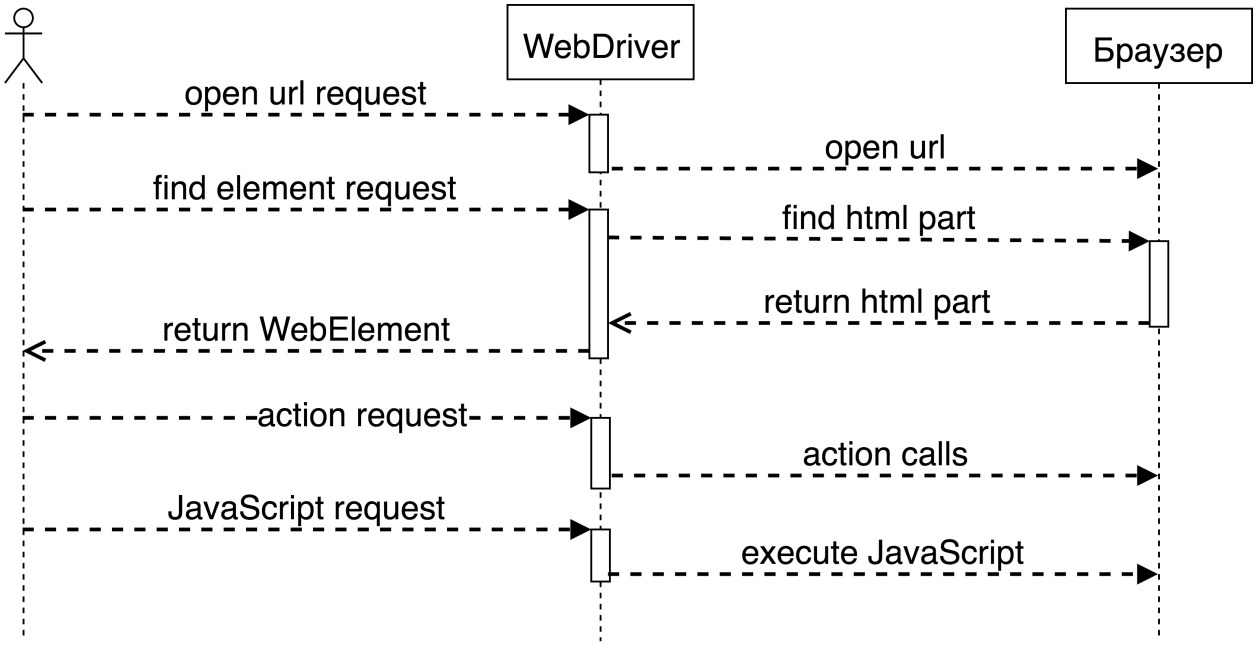

Рис. 2: Принцип работы Selenium WebDriver.

Пользователь обращается к библиотеке WebDriver, и она уже посылает команды браузеру. Большинство команд можно разделить на четыре типа: перейти по URL (Uniform Resource Locator), найти элемент на web-странице, выполнить действие над элементом и исполнить скрипт. WebDriver посылает команды и не ждет, когда действия завершатся, поэтому возникает необходимость выставлять времена ожидания самостоятельно.

Основным ограничением Selenium WebDriver является то, что при тестировании открывается новая копия браузера со своей версией тестируемой страницы, производить работу над открытой пользователем страницей не допускается. Однако, это не является проблемой исключительно Selenium. Большинство инструментов для графического тестирования также не способны справиться с этой задачей.

1.4. Selenide и Java.awt.Robot

WebDriver позиционируется как стандарт интерфейсов для управления браузерами и, как следствие, он реализует только необходимые команды. Однако, существует множество библиотек, упрощающих часто используемые операции. Одной из таких библиотек является Selenide [8].

Ниже перечислены некоторые возможности Selenide.

- Автоматическое управление жизненным циклом браузера (открыть/закрыть/перезапустить).
- Селекторы для элементов реализованы в стиле jQuery.
- Поддержка Ajax.

Selenide включает в себя множество встроенных проверок, определяющих состояние объекта (visible, shouldExist, и др.). Также она предоставляет широкие возможности для работы с коллекциями элементов.

В настоящей работе также используется класс java.awt.Robot [5]. Если Selenium посылает команды браузеру, то Robot фактически захватывает мышь и взаимодействует с web-страницей с помощью имитации кликов и перемещений. java.awt.Robot накладывает серьезные ограничения на GUI тестирование, поскольку требует абсолютного бездействия пользователя в момент выполнения теста. Однако, он имеет более широкий спектр действий, потому что имитирует действия пользователя на более низком уровне. Помимо этого он отображает мышь и ее движение.

2. Архитектура разработанного решения

Решение представляет из себя набор связанных между собой сервисов, с помощью которых можно производить GUI тестирование проекта WMP. Рис. 3 отражает общую архитектуру решения.

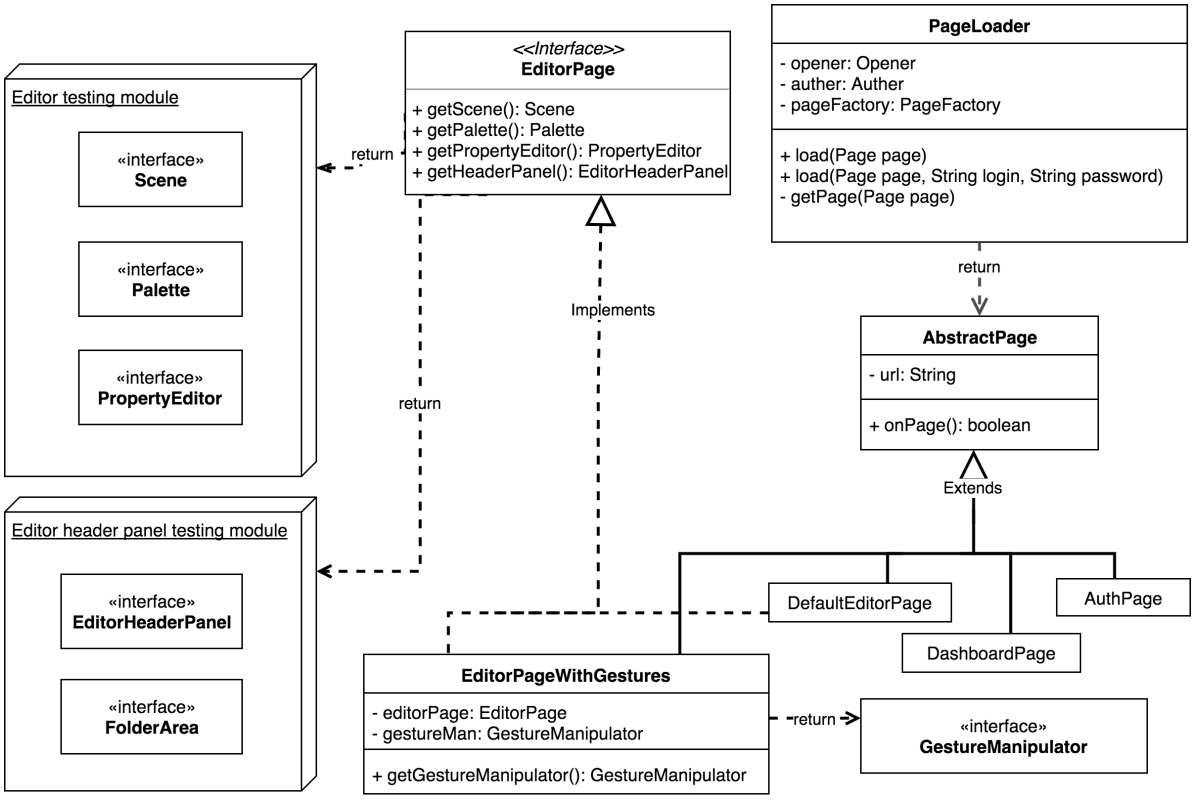

Рис. 3: Архитектура разработанного решения.

Интерфейсы, описанные на диаграмме, и класс PageLoader отражают реализованные сервисы. Все они доступны разработчику графических тестов при проектировании сценариев тестирующих скриптов. Приведенные интерфейсы будут подробно описаны в последующих главах.

Все сервисы работают с HTML-представлением web-страниц WMP, поиск конкретных элементов осуществляется либо по их идентификатору, либо с помощью CSS-селекторов. Для большей гибкости был реализован сервер, который предоставляет необходимую для поиска элементов информацию для каждой страницы WMP. Детальное его описание будет представлено далее.

3. Сервисы навигации по проекту

Как отмечалось выше, для доступа к страницам редактора необходима предварительная авторизация. Чтобы тестировать редакторы, первоначально требуется зайти на страницу аутентификации, ввести логин с паролем и уже потом переходить на нужную страницу браузера.

3.1. Архитектура инструментов навигации

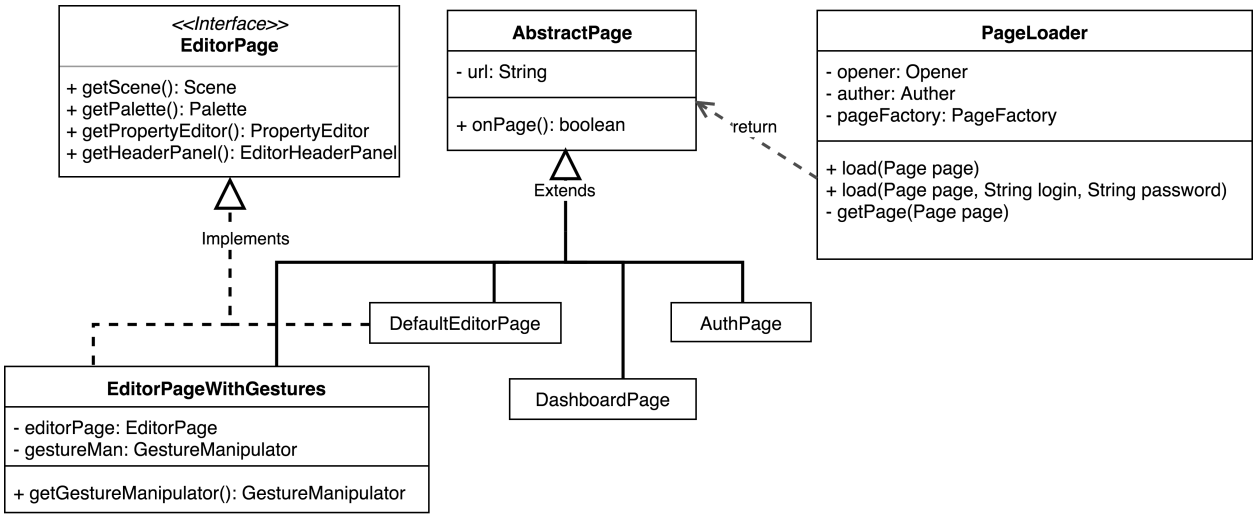

Рис. 4: Архитектура системы навигации разработанного решения для GUI тестирования WMP.

В предлагаемом решении за загрузку страниц отвечает класс PageLoader. Вызов метода load с одной стороны открывает нужную нам страницу в браузере, с другой — возвращает программное представление этой страницы, наследника класса AbstractPage. На данный момент WMP поддерживает два редактора — редактор роботов и BPMN редактор. Оба они состоят из сцены, палитры, редактора свойств и верхней панели меню, но при этом редактор роботов имеет несколько большую функциональность — в нем доступно построение диаграмм с помощью жестов мыши, поэтому на базе основного класса DefaultEditorPage реализован DefaultEditorPageWithGestures. Он выступает в роли декоратора, добавляя новую функциональность.

3.2. Авторизация и создание страниц

PageLoader работает с сервисами Opener и Auther. Они необходимы, чтобы перейти на нужную страницу и авторизироваться соответственно. При загрузке страницы с помощью PageLoader производится первичная попытка перехода на нужный URL. Если доступ запрещен, открывается страница аутентификации, вводятся логин и пароль, затем нажимается кнопка подтверждения, после чего происходит повторный переход на нужный нам URL. Все эти действия происходят автоматически, предоставляя возможность визуально наблюдать за процессом. После загрузки страницы с помощью класса PageFactory возвращается объект наследник класса AbstractPage, который содержит все доступные на этой странице сервисы для тестирования. К ним можно обращаться и вызвать требуемые методы.

В проекте WMP также встречаются случаи перехода между страницами по клику на кнопку, которая отображается пользователю на web-странице. Один из реализованных в настоящей работе сервисов предоставляет возможность имитировать такой переход. В этом случае класс PageLoader не используется по причине того, что переход по URL происходит без физического заполнения адресной строки браузера. Однако, с помощью класса PageFactory наследник класса AbstractPage все равно возвращается.

4. Сервисы тестирования редактора

Как уже было сказано, редакторы однотипны и состоят из сцены, палитры, редактора свойств и верхней панели меню. Различаются они наполнением палитры, свойствами элементов и внутренней логикой. Основной задачей данной дипломной работы является описание API высокого уровня для манипуляций с диаграммами.

4.1. Сервисы тестирования конструктора диаграмм

В разработанном решении сцена, палитра и редактор свойств являются сервисами, которые предоставляют интерфейсы для конструирования диаграмм. Вместе они позволяют имитировать следующие действия:

- перетаскивать элементы из палитры на сцену;
- перемещать элементы внутри самой сцены;
- удалять элементы со сцены;
- добавлять/удалять связи между элементами;
- задавать свойства элементам.

Также реализован весь необходимый API для проверки корректности результата перечисленных операций. Чтобы обратиться к любому из этих сервисов, необходимо загрузить страницу редактора и вызвать соответствующий метод у полученного объекта класса EditorPage.

4.1.1. Архитектура

Описанные выше сервисы взаимодействуют с объектами класса SceneElement, являющимся классом-родителем для всех элементов, которые находятся на сцене. Каждый такой объект имеет координаты и содержит в себе WebElement — основной объект библиотеки Selenium

WebDriver, который описывает HTML-элемент. Блоки и стрелки наследуются от SceneElement и дополняются нужными им полями. Вышесказанное иллюстрируется на Рис. 5.

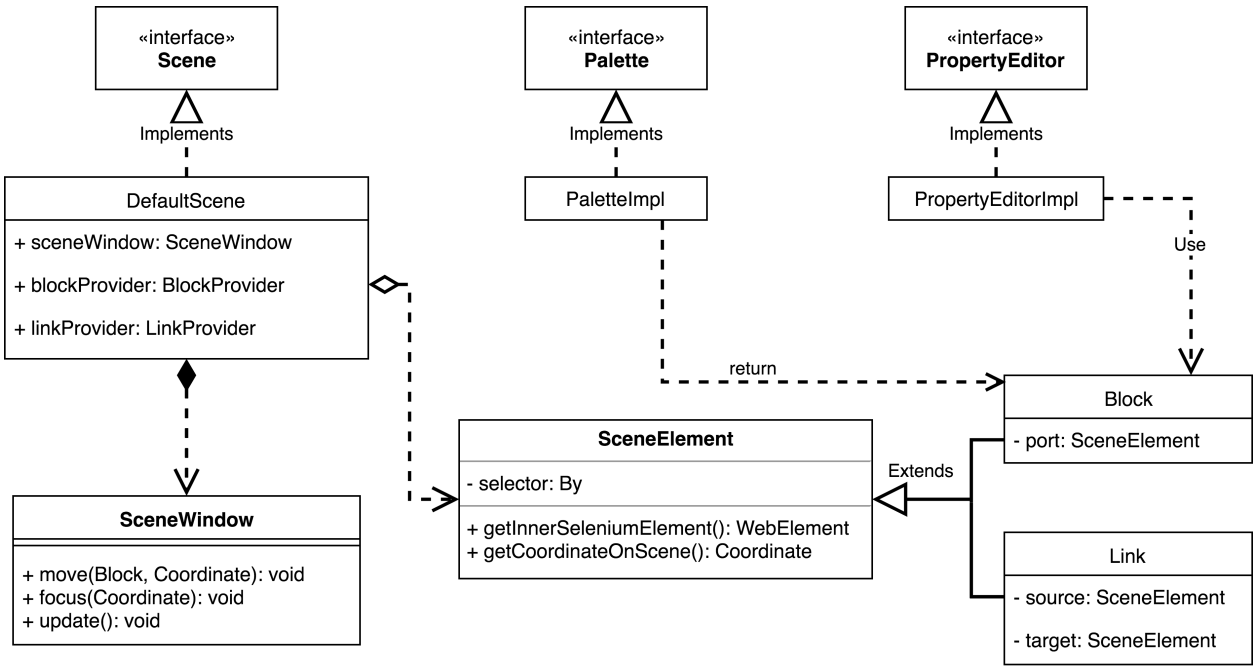

Рис. 5: Архитектура сервисов тестирования конструктора диаграмм.

В совокупности приведенные сервисы полностью описывают конструктор диаграмм. Для удобства проектирования тестов, реализован класс EditorPageFacade, который имеет доступ ко всему предоставляемому API рассматриваемых сервисов. Он не доступен разработчику тестов, однако передается в конструкторе всем элементам описываемой структуры. Благодаря этому, в частности, появляется возможность добавить классу Block метод moveToCell(int cellX, int cellY), который с помощью сервиса Scene осуществит перемещение блока в указанную клетку.

4.1.2. Сервис Palette

Сервис Palette служит для тестирования палитры редактора диаграмм и имеет внешний вид, представленный на Рис. 6.

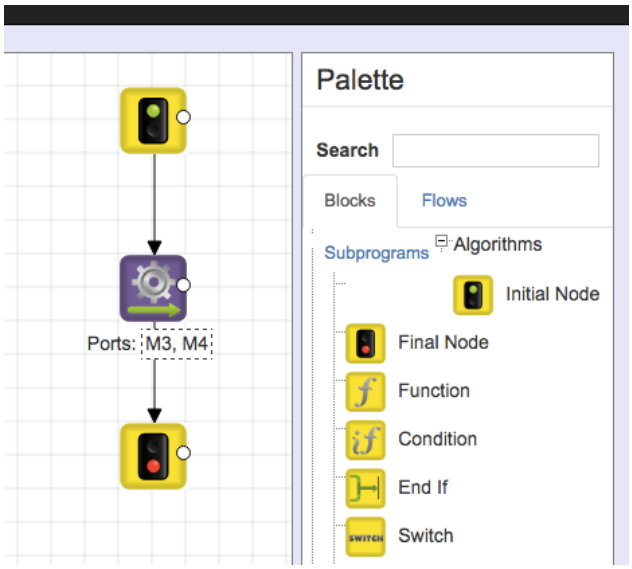

Рис. 6: Палитра редактора диаграмм.

На данном этапе разработки WMP, для тестирования палитры необходим только следующий метод:

```
PaletteElement getElement(String elementName)
```

Приведенный метод возвращает элемент класса PaletteElement, если имя, указанное в параметре, совпадает с отображаемым именем запрашиваемого элемента в окне браузера. Таким образом можно идентифицировать любой элемент палитры с указанным именем, вне зависимости от того, какой редактор открыт. С объектами класса PaletteElement уже может работать сервис Scene, который будет подробно описан далее. Следует отметить, что с помощью данного метода производится только поиск объекта, без его перемещения на сцену.

4.1.3. Сервис PropertyEditor

Сервис PropertyEditor служит для тестирования редактора свойств редактора диаграмм и предоставляет методы, описанные в Код 1.

```
// Имитирует присвоение указанного значения соответствующему
// свойству указанного элемента
void setProperty (
        SelenideElement element,
        String propertyName,
        String propertyValue);

// Возвращает значение указанного свойства выбранного элемента
String getProperty(SelenideElement element, String propertyName);
```

Код 1: Методы сервиса PropertyEditor.

В некоторых случаях свойство задается как строковое значение, в некоторых как окошко с возможностью выбора варианта. В зависимости от свойств выбранного элемента на сцене редактор свойств может представляться по-разному. На Рис. 7 показаны состояния редактора свойств при выбранных элементах, имеющих типы ”Motor Forward” и ”Painter Color” соответственно.

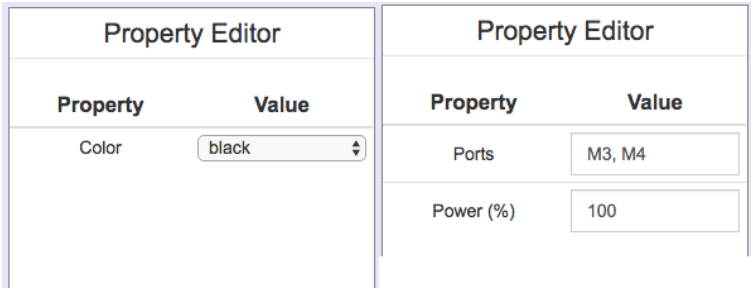

Рис. 7: Состояния редактора свойств редактора диаграмм.

Использование методов приведенного интерфейса позволяет разработчику тестов не думать, как именно представляются свойства в редакторе. Достаточно указать название свойства и его значение, и внутренняя логика PropertyEditor вне зависимости от представления запра-

шиваемого свойства корректно сымитирует установку нужного значения.

4.1.4. Сервис Scene

Сервис Scene служит для тестирования сцены редактора диаграмм. Ниже представлена часть сцены с диаграммой, расположенной на ней.

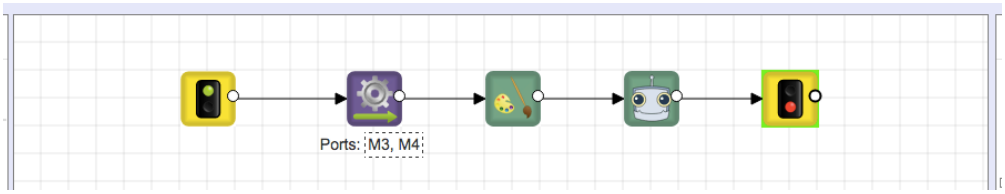

Рис. 8: Часть сцены с диаграммой.

Сервис Scene включает в себя методы, перечисленные в Код 2.

```
// Имитирует перетаскивание элемента из палитры на сцену
Block dragAndDrop(PaletteElement paletteElement);
// Имитирует перетаскивание элемента из палитры на сцену
// в указанную клетку
Block dragAndDrop(PaletteElement element, int cellX, int cellY);
// Имитирует перемещение элемента внутри сцены
void moveToCell(Block block, int cellX, int cellY);
// Проверяет, что элемент находится на сцене
boolean exist(SceneElement element);
// Имитирует удаление элемента со сцены
void remove(SceneElement element);
// Имитирует соединение двух блоков стрелкой
Link addLink(Block source, Block target);
// Возвращает список всех блоков на сцене
List<Block> getBlocks();
// Имитирует очистку сцены
void clean();
```

Код 2: Методы сервиса Scene.

Сервис Scene работает с наследниками класса SceneElement — Block и Link. Block описывает элемент диаграммы и характеризуется иденти-

фикатором и портом для соединения с другими блоками. Link описывает существующую связь между элементами и характеризуется идентификатором и ссылками на начало и конец этой связи.

В WMP сцена реализована в виде контейнера, где могут располагаться объекты, при этом пользователю предоставляется только ограниченная часть сцены, которую он может смещать с помощью стрелок клавиатуры. Данное представление отражается на Рис. 9.

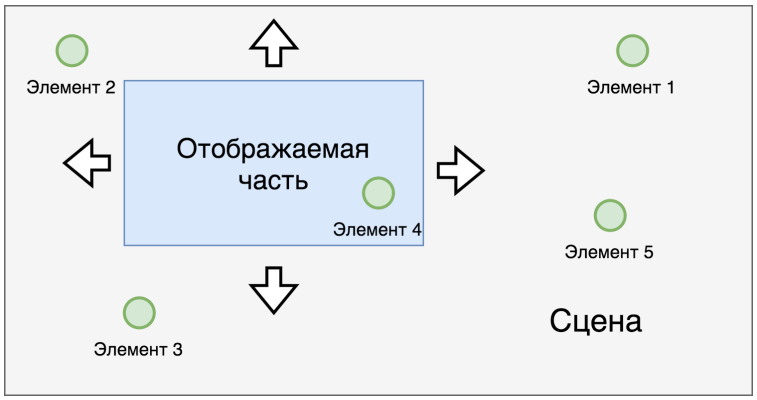

Рис. 9: Представление сцены редактора диаграмм.

Чтобы переместить элемент из одного края сцены в другой, требуется одновременно двигать элемент в пределах экрана и смещать сам экран. Эту функциональность реализует класс SceneWindow. Он описывает видимую пользователю часть сцены. HTML-представления объектов на сцене содержат их координаты внутри сцены, но не предоставляют координаты внутри видимой части. Selenium предоставляет API для идентификации элемента и его перемещения по заданному смещению, однако, чтобы смещать объект внутри некоторого контейнера, нужно знать его координаты внутри конкретно этого контейнера. Для получения этой информации к HTML-представлению страницы добавляются два скрытых div-элемента, в которых с помощью посылаемых браузеру скриптов прописываются координаты смещения видимой части сцены по отношению к самой сцене. С помощью значений этих div-элементов и вычисляются реальные координаты объекта на видимой области сцены.

4.2. Сервис тестирования верхнего меню

На странице редактора диаграмм доступно верхнее меню, внешний вид которого представлен на Рис. 10.

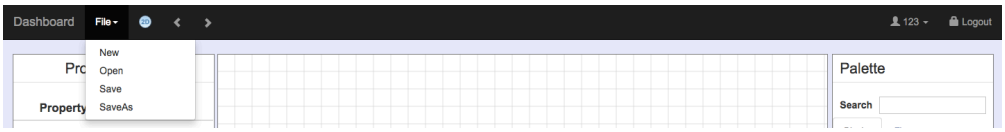

Рис. 10: Верхнее меню редактора диаграмм.

В верхнем меню редактора диаграмм доступны кнопки Dashboard и File. По клику на Dashboard осуществляется переход на страницу редактора роботов, а по клику на File открывается дополнительное меню для манипуляций с диаграммами.

Одним из результатов данной работы является сервис для тестирования действий пользователя над рассматриваемым меню. Код 3 иллюстрирует API этого сервиса.

```
// Имитирует клик по Dashboard
DashboardPage toDashboard();
// Имитирует создание новой диаграммы
void newDiagram();
// Имитирует открытие дерева папок (подробнее описано ниже)
FolderArea getFolderArea();
// Имитирует сохранение диаграммы по указанному пути
void saveDiagram(String path);
// Имитирует сохранение диаграммы под текущем именем
void saveDiagram();
// Имитирует открытие диаграммы по указанному пути
void openDiagram(String path);
// Проверяет, что диаграмма по указанному пути существует
boolean isDiagramExist(String path);
// Проверяет текущую диаграмму и диаграмму, сохраненную
// по указанному пути, на эквивалентность
boolean equalsDiagrams(String path);
```

Код 3: Методы сервиса верхнего меню редактора диаграмм.

4.2.1. Сервис тестирования дерева папок

Чтобы открыть или сохранить диаграмму, необходимо нажать на пункт File из верхнего меню редактора и выбрать подпункты open или save. В результате появится окно, представленное на Рис. 11.

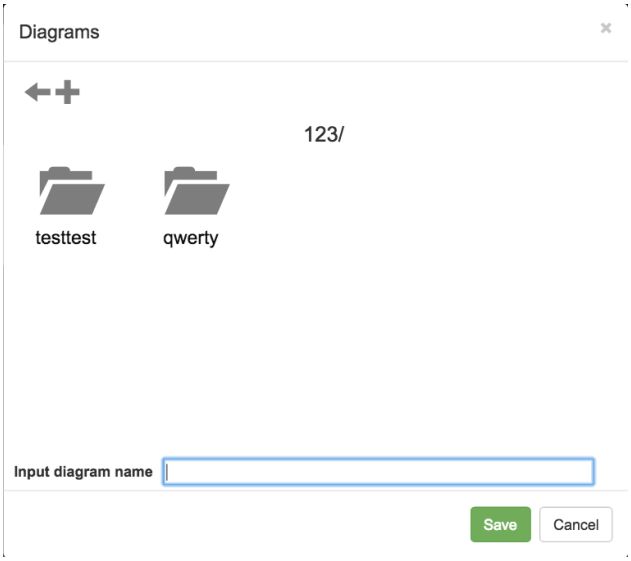

Рис. 11: Окно дерева папок редактора диаграмм.

Внутри приведенного окна можно создавать и удалять папки и подпапки, переходить вверх и вниз по каталогу. В рамках данной работы реализован API для имитации и тестирования всех основных сценариев работы с ним. Код 4 отражает список реализованных функций.

```
// Имитация создания папки с указанным именем в текущем каталоге
FolderArea createFolder(String name);
// Имитация перемещения вверх по каталогу в папку с указанным именем
FolderArea moveForward(String name);
// Имитация перемещения вниз по каталогу
FolderArea moveBack();
// Возвращает текущий путь каталога
String getCurrentPath();
// Имитация перемещения в указанный каталог в дереве папок.
// Если каталога не существует, он создастся
FolderArea move (String path);
// Имитация удаления папки с указанным именем в текущем каталоге
FolderArea deleteFolder (String name);
```

Код 4: Методы сервиса дерева папок.

При реализации указанных методов была обнаружена ошибка, связанная с удалением папок. При удалении одной из папок, происходило удаление всех ее предков, что влекло, в конечном итоге, удаление корневой папки. Последствия этого, однако, проявлялись только при повторном открытии дерева папок. Например, при последующей попытке сохранить диаграмму, клик по кнопке подтверждения не влек за собой никаких изменений. Ошибка была локализована и исправлена в коде платформы WMP в рамках настоящей работы.

4.2.2. Сервис хранения диаграмм

Некоторые методы сервиса тестирования верхнего меню редактора диаграмм можно выделить в отдельную подгруппу. К ней мы будем относить функции, которые служат для следующих целей:

- сохранение диаграммы;
- проверка на существование диаграммы;
- проверка на эквивалентность двух диаграмм.

Методы описанной подгруппы исполняются в два шага, приведенных ниже.

1. Автоматическое открытие дерева папок и переход в конечную папку, указанную в качестве параметра пути.

2. Вызов необходимого метода сервиса хранения диаграмм.

Сервис хранения диаграмм реализуется классом DiagramStoreService. Он является вспомогательным для сервиса верхнего меню и не доступен для разработчика тестов. Методы данного сервиса перечислены в Код 5.

```
// Сохраняет диаграмму в открытой в данный момент папке
// под именем, указанным в качестве параметра
void saveDiagram(String key).
// Сохраняет диаграмму в открытой в данный момент папке
// под последним известным именем
void saveDiagram().
// Проверяет, существует ли диаграмма с именем, указанным в качестве
// параметра, в открытой в данный момент папке
boolean isDiagramExist(String key).
// Проверяет на эквивалентность диаграмму на сцене и диаграмму
// с именем, указанным в качестве параметра, в открытой в данный
// момент папке
boolean equalsDiagrams(String key).
```

Код 5: Методы сервиса хранения диаграмм.

При использовании DiagramStoreService подразумевается, что нужная папка уже открыта в дереве папок.

В рамках настоящей работы две диаграммы называются эквивалентными, если эквивалентны их специальным образом преобразованные HTML-представления. Ниже представлено описание и обоснование алгоритма преобразования.

1. Выделить HTML-представление диаграммы.

2. Очистить все автоматически генерируемые идентификаторы.

3. Отсортировать полученные строки.

4. Объединить их и добавить в хранилище.

Диаграмма однозначно определяет свое HTML-представление, однако, разные HTML-представления могут определять одну и ту же диаграмму. WMP сохраняет диаграмму как объект, состоящий и блоков и стрелок, и при открытии каждый из этих элементов последовательно добавляется на сцену, причем порядок добавления может отличаться от порядка, в котором пользователь изначально добавлял элементы на сцену. Также при открытии автоматически генерируются идентификаторы, используемые библиотекой JointJS [7]. Их значения при каждом открытии диаграммы разные. Однако, на этом различия в HTML-представлении заканчиваются. В силу вышеперечисленных причин достаточно убрать все автоматически сгенерированные идентификаторы и отсортировать HTML-строки.

4.3. Сервис тестирования жестов мыши

Редактор роботов содержит механизм распознавания жестов мышью для создания элементов на диаграммах. Если зажать правую кнопку мыши и начать ее перемещать, мышь будет оставлять след. Если траектория движения описывает какой-либо из элементов палитры, то он появится на сцене. На Рис. 12 представлено описание в виде жеста элемента "таймер".

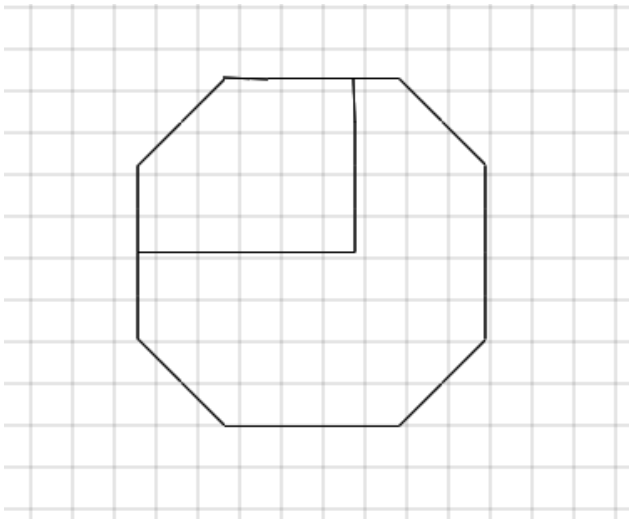

Рис. 12: Траектория жеста мыши, описывающая элемент "таймер".

Кроме того, если навести курсор на блок на сцене и зажать правую кнопку мыши, а после этого переместить мышь к другому блоку на сцене, и отпустить правую кнопку, эти блоки соединятся стрелкой.

В рамках данной работы реализован API для имитации и тестирования этой функциональности, методы которого представлены в Код 6.

```
// Имитация рисования жеста, описывающего некоторый элемент
Block draw (String name);
// Имитация соединения двух блоков стрелкой с помощью жестов мыши
Link drawLine (Block sourceBlock, Block targetBlock);
// Имитация произвольного рисунка пользователя
Optional<SceneElement> drawByOffsets(
        Coordinate start,
        List<Integer> offsetsX,
        List<Integer> offsetsY);
```

Код 6: Методы сервиса жестов мыши.

Selenium WebDriver не предоставляет API для зажатия правой кнопки мыши, поэтому данный сервис использует библиотеку java.awt.Robot. Если WebDriver посылал команды браузеру, то Robot напрямую захватывает мышь и производит манипуляции непосредственно над ней. Чтобы сопоставить координаты в браузере с координатами на экране, перед открытием страницы WMP открывается специальная HTML-страница, где с помощью бинарного поиска происходит сопоставление координат внутри браузера с координатами на экране.

При разработке данного модуля тестирования было также выявлено и исправлено несколько ошибок в работе средств распознавания жестов мышью.

5. Средства конфигурирования созданного решения

Selenium WebDriver осуществляет поиск элемента по его CSS-селектору. При изменении селектора элемента на стороне клиента (напимер, при изменении кода web-страницы) возникает необходимость изменить все обращения к этому элементу и на стороне GUI тестирования. Кроме того, человек, не участвовавший в разработке графических тестов, не знает, идентификаторы каких элементов можно менять безопасно, а каких нельзя. В силу этого ему неизбежно придется разбираться в структуре классов GUI тестирования. Отсюда возникает потребность в файлах конфигураций, где будут указываться элемент, участвующий в тестировании, и его CSS-селектор или идентификатор. В таком случае разработчику потребуется просмотреть только небольшой набор простых файлов.

5.1. Структура и ограничения конфигурационных файлов

Для записи файлов конфигураций был выбран формат JSON. В силу большого числа элементов, участвующих в тестировании, отношение вложенности элементов друг в друга в HTML-представлении было перенесено на файлы конфигураций. В Код 7 представлен пример части файла конфигураций, описывающего дерево папок.

```
{
    ''id'' : ''diagramArea'',
    ''contextMenu'' : {
        ''id'' : ''open-diagram-context-menu'',
        ''deleteItem'' : {
            ''id'' : ''contextMenuDelete''
        }
    },
    ''closeItem'' : {
        ''id'' : ''foldersCloseItem''
    },
    ''folders'' : {
        ''selector'' : ''#diagramArea .folders''
    },
    ''diagrams'' : {
        ''selector'' : ''#diagramArea .diagrams''
    }
}
```

Код 7: Часть файла конфигураций, описывающего дерево папок.

Если рассматриваемый элемент представляет из себя объект, существующий в единственном экземпляре, то ему можно однозначно сопоставить идентификатор. Однако, некоторые тестируемые элементы предоставляются сторонними библиотеками, и не всегда возможно задать им идентификаторы из окружения программы. В таких случаях объект описывается своим CSS-селектором. Часто при тестировании приходится обращаться к коллекциям элементов, их также было решено описывать CSS-селекторами. WMP активно развивается, и модуль графического тестирования продолжит развиваться вместе с ним. В перспективе существует вероятность использования классов вместо CSS-селекторов для коллекций элементов.

WMP содержит страницы редактора диаграмм для роботов и BPMN-редактора. Они схожи по внутренней структуре, однако есть и различия. Например, в первом присутствует модуль для жестов мыши, а

во втором нет. Учитывая стратегию вложенности, одни и те же селекторы должны быть указаны в описаниях обеих этих страниц. Чтобы избежать дублирования, было принято решение использовать не один большой файл конфигураций, а множество небольших. В Код 8 отражен способ их объединения.

```
{
    ''robotEditor'': {
        ''scene'' : ''$path:editor/scene.json'',
        ''palette'' : ''$path:editor/pallete.json'',
        ''propertyEditor'' : ''$path:editor/propertyEditor.json'',
        ''editorHeaderPanel'' : ''$path:editor/editorHeaderPanel.json'',
        ''gestures'' : ''$path:/editor/robots/robotGestures.json''
    },
    ''bpmnEditor'': {
        ''scene'' : ''$path:editor/scene.json'',
        ''palette'' : ''$path:editor/pallete.json'',
        ''propertyEditor'' : ''$path:editor/propertyEditor.json'',
        ''editorHeaderPanel'' : ''$path:editor/editorHeaderPanel.json''
    }
}
```

Код 8: Объединение файлов конфигураций.

Чтобы добавить отдельный JSON в качестве компонента описываемого элемента, следует разместить путь до него с префиксом ”$path:”.

В рамках данной работы был реализован REST сервис, который собирает маленькие JSON-файлы в один большой и предоставляет API для получения каждой из страниц WMP нужной ей части полученного JSON-файла. Сервис реализован с помощью Spring Boots [6].

5.2. Интегрирование конфигураций в WMP

Страницы WMP устроены как отдельные приложения, реализованные с помощью Spring MVC [2]. Если пользователь вводит в браузере

адрес одной из страниц WMP, срабатывает соответствующий контроллер, после чего открывается запрашиваемая станица. Требуется передать этой странице файл конфигураций в качестве параметра. Для достижения этой цели реализовано следующее решение.

Контроллеры посылают http-запросы сервису селекторов, ответом на которые являются JSON-файлы конфигураций для каждой конкретной страницы. Эти файлы и передаются в качестве параметра к web-представлению. Редакторы диаграмм проекта WMP реализованы на стороне клиента с помощью AngularJS [1] и TypeScript [4]. В связи с этим, был разработан сервис на TypeScript, который позволил полностью интегрировать конфигурации в проект.

5.3. Интегрирование конфигураций в модуль GUI тестирования

Для встраивания конфигураций в модуль GUI тестирования был реализован класс SelectorService. Он позволяет, указывая путь к элементу через все вложенные, получить либо его идентификатор, либо CSS-селектор. Поскольку цепочки вложенности могут быть довольно длинными, сервис предоставляет возможность выделять подуровни. Данный сервис недоступен для разработчика тестов и используется только внутри модуля тестирования.

6. Обучающие анимации

Реализованного API достаточно для составления сложных сценариев графического тестирования. В Код 9 приведен пример содержательного теста.

```
// Загрузка страницы редактора роботов
EditorPage editorPage = pageLoader.load(Page.EditorRobots);
Scene scene = editorPage.getScene();
Palette palette = editorPage.getPalette();
// Перемещение блока ”Initial Node” из палитры на сцену
Block initial = scene
        .dragAndDrop(palete.getElement(”Initial Node”), 4, 4);
// Перемещение блока ”Motors Forward” из палитры на сцену
Block motorForward = scene
        .dragAndDrop(palete.getElement(”Motors Forward”), 10, 4);
// Соединение блоков стрелкой
scene.addLink(initial, motorForward);
// Соединение блоков стрелкой
Block end = editorPage.getGestureManipulator().draw(”Final Node”);
// Проверка, что создался правильный блок
assert (”Final Node”).equals(end.getName());
// Перемещение блока end в клетку (30, 30)
end.moveToCell(30, 30);
// Соединение блок стрелкой с помощью жестов мыши
editorPage.getGestureManipulator().drawLine(motorForward, end);
// Сохранение диаграммы в папке test под именем test
editorPage.getHeaderPanel().saveDiagram(”test/test”);
// Проверка, что диаграмма сохранилась
assert editorPage.getHeaderPanel().exist(”test/test”);
```

Код 9: Пример тестирующего скрипта.

После запуска приведенного кода откроется браузер, и шаг за шагом все описанные действия будут исполнены. Если включить захват видео с экрана, можно записать обучающую анимацию. При отправке тому или иному web-элементу действия с помощью Selenium происходят задержки, связанные с тем, что эти действия выполняются в отдельных

потоках. На качество тестов данный факт не влияет, но он существенно ухудшает анимацию. Анимация работает корректно, но для того, чтобы она работала красиво, потребуется большая дополнительная работа.

7. Заключение

В ходе данной работы были получены следующие результаты.

1. Разработаны сервисы GUI тестирования для навигации по проекту WMP.
2. Разработаны сервисы GUI тестирования для редактора диаграмм.
   - Сервисы для тестирования конструктора диаграмм.
   - Сервис для тестирования верхнего меню редактора.
   - Сервис для тестирования дерева папок.
   - Сервис для тестирования жестов мыши.
3. Спроектированы и интегрированы средства конфигурации.
4. Реализованы инструменты, позволяющие создавать обучающие анимации.
5. Выявлен ряд ошибок в коде платформы WMP, некоторые из них исправлены.

Результаты данной работы позволяют создавать автоматизированные тесты графического интерфейса проекта WMP при помощи высокоуровнего API. Методы разработанных сервисов позволяют имитировать все основные манипуляции с диаграммами, все возможные действия пользователя с верхним меню редактора, включая сохранение и открытие диаграмм, а также жесты мыши. Благодаря спроектированным и внедренным средствам конфигурирования, появляется возможность изменять идентификаторы элементов, участвующих в тестировании, только в одном месте, что позволяет разрабатывать WMP практически независимо от GUI тестирования.

Список литературы

[1] AngularJS. AngularJS API Docs. — Режим доступа: https://docs.angularjs.org/api (дата обращения: 01.09.2016).

[2] Johnson Rod и др. Spring Framework Reference Documentation. — Режим доступа: https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/ (дата обращения: 01.09.2016).

[3] Kumar Yogesh. Comparative Study of Automated Testing Tools: Selenium, SoapUI, HP Unified Functional Testing and Test Complete. — 2015. — Режим доступа: http://www.jetir.org/papers/JETIR1509007.pdf (дата обращения: 01.09.2016).

[4] Microsoft. TypeScript Documentation. — Режим доступа: https://www.typescriptlang.org/docs/home.html (дата обращения: 01.09.2016).

[5] Oracle. Java Platform Standard Edition 7 Documentation. — Режим доступа: https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html (дата обращения: 01.09.2016).

[6] Phillip Webb и др. Spring Boot Reference Guide. — Режим доступа: https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/ (дата обращения: 01.09.2016).

[7] Rappid. Joint API. — Режим доступа: http://resources.jointjs.com/docs/jointjs/v1.1/joint.html (дата обращения: 01.09.2016).

[8] Selenide. Selenide documentation. — Режим доступа: http://ru.selenide.org/documentation.html (дата обращения: 01.09.2016).

[9] Selenium. Selenium Documentation. — Режим доступа: http://www.seleniumhq.org/docs/ (дата обращения: 01.09.2016).

[10] Team Sikuli Doc. Sikuli Documentation. — 2012. — Режим доступа: http://doc.sikuli.org/ (дата обращения: 01.09.2016).

[11] Wikipedia. Ranorex. — Режим доступа: https://en.wikipedia.org/wiki/Ranorex (дата обращения: 01.09.2016).

[12] Wikipedia. SoapUI. — Режим доступа: https://en.wikipedia.org/wiki/SoapUI (дата обращения: 01.09.2016).

[13] Wikipedia. TestComplete. — Режим доступа: https://en.wikipedia.org/wiki/TestComplete (дата обращения: 01.09.2016).