Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра Системного Программирования

Захаров Владимир Александрович

Разработка онлайн среды графического программирования роботов

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
ст. преп. Брыксин Т. А.

Рецензент:
ст. преп. Литвинов Ю. В.

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engineering

Zakharov Vladimir

Online graphical robots programming environment development

Bachelor's Thesis

Admitted for defence.
Head of the chair:
professor A. N. Terekhov

Scientific supervisor:
assistant T. A. Bryksin

Reviewer:
assistant Y. V. Litvinov

Saint-Petersburg
2015

Оглавление

Введение

В последнее время активно развиваются технологии, связанные с робототехникой. Людям становится интересно не только использование роботов с уже заданным поведением, но и программирование роботов. Однако разработка программ для роботов является непростой задачей даже для программистов. Именно поэтому сейчас создаются инструменты, позволяющие упростить эту задачу.

Одним из способов упрощения создания программ являются визуальные средства разработки. В таких инструментах программа представляется в виде последовательности блоков, соединенных линиями по некоторым правилам. Такой подоход нагляден и понятен даже детям.

Примером визуального средства разработки является среда программирования роботов TRIK Studio[1], разрабатываемая в том числе преподавателями и студентами математико-механического факультета СПбГУ. Данная среда позволяет задавать поведение робота при помощи диаграмм, т.е. последовательности картинок, соединенных линиями. TRIK Studio имеет много возможностей, но для использования требует установки на компьютер, а это плохо тем, что для каждого устройства приходится устанавливать приложение, переносить вручную изменения, сделанные на разных компьютерах, кроме того, могут возникать ошибки, связанные с конкретным компьютером.

С развитием интернет технологий появляется возможность создать онлайн редактор программирования роботов. Такой подход исключает ошибки, связанные с конкретными платформами, позволяет получить доступ к инструменту с любого устройства, имеющего браузер, синхронизироваться между устройствами. Кроме того, становится возможным содание онлайн инфраструктры для управления роботами через веб-интерфейс.

[1]Среда TRIK Studio, URL: `http://blog.trikset.com/p/trik-studio.html` (дата обращения 17.05.2015)

Поэтому после анализа существующих решений на кафедре системного программирования СПбГУ было принято решение создать онлайн среду графического программирования роботов.

1. Постановка задачи

Целью данной бакалаврской работы является разработка онлайн среды графического программирования роботов, позволяющей создавать диаграммы поведения робота и отображать движение робота в двумерной модели.

Для достижения этой цели был сформулирован следующий набор задач.

- Проанализировать существующие решения.
- Выбрать необходимые для реализации инструменты.
- Разработать общую архитектуру онлайн среды графического программирования роботов.
- Реализовать серверную часть приложения.
- Реализовать редактор диаграмм поведения робота.
- Реализовать двумерную модель поведения робота.

2. Обзор существующих решений

На данный момент существует несколько инструментов для графического программирования роботов. Некоторые из них предназначены в основном для обучения, другие же для достаточно серьезных практических задач. В зависимости от предназначения данные среды обладают разным функционалом.

2.1. TRIK Studio

Одним из примеров является среда программирования роботов TRIK Studio[9, 11, 12]. В ней имеются средства для создания диаграммы поведения робота и для отображения действий робота в двумерной модели. TRIK Studio позволяет создавать программы для контроллеров TRIK, LEGO EV3, LEGO Mindstorms NXT 2.0. В этом инструменте можно задать программу поведения робота, загрузить ее на самого робота, либо посмотреть поведение в двумерной модели.

Инструмент для создания диаграмм изображен на рис. 1. С его помощью можно составить диаграмму из заранее заданного набора блоков, изменять свойства элементов, конфигурировать порты, сохранять, загружать диаграммы. Кроме того, имеется возможность создания блоков диаграммы при помощи жестов мышью.

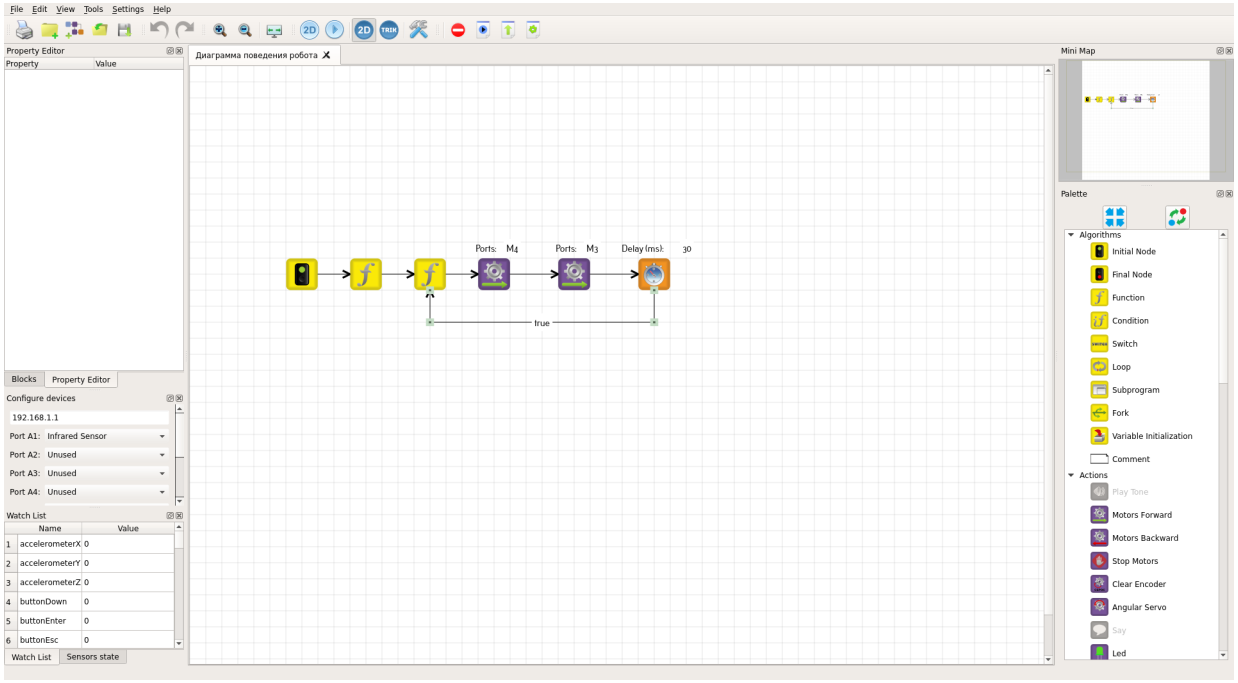

Рис. 1: TRIK Studio — Редактор диаграмм

После создания диаграммы посмотреть поведение робота можно двумя способами: загрузить и выполнить программу на реальном роботе, либо перейти к двумерной модели. Переключение между этими способами выполняется одной кнопкой. Рассмотрим двумерную модель, представленную на рис. 2. Здесь можно создать препятствия для робота, конфигурировать порты, изменять настройки физики и скорости воспроизведения, запустить программу и посмотреть поведение робота. При воспроизведении программы на диаграмме выделяется выполняемый блок, имеется возможность отлаживать программу. Таким образом можно учиться программированию роботов, не имея реального робота.

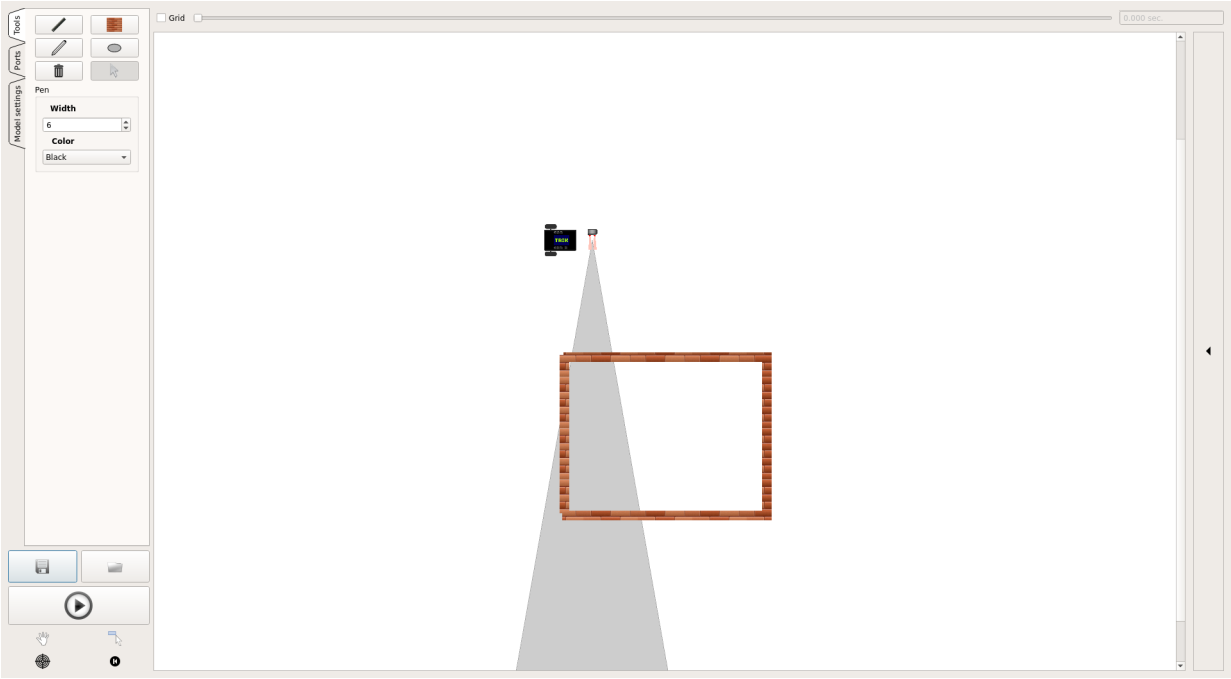

Рис. 2: TRIK Studio — Двумерная модель

Несомненным плюсом данного продукта является кроссплатформенность.

2.2. Microsoft Robotics Developer Studio

Microsoft Robotics Developer Studio[2] — мощная среда для управления роботами и симуляции их поведения. Поставляется только для семейства операционных систем Windows. Данная среда включает в себя следующее: среду для графического программирования роботов на языке Visual Programming Language (VPL), среду симуляции поведения робота в 3D — Visual Simulation Environment, инструменты командной строки, позволяющие взаимодействовать с проектами Visual Studio, поддержку нескольких языков, среди которых C#, Visual Basic .NET, JavaScript и IronPython. Среда для программирования представлена на рис. 3.

[2]Среда Microsoft Robotics Developer Studio, URL: `https://msdn.microsoft.com/en-us/library/bb483024.aspx` (дата обращения 17.05.2015)

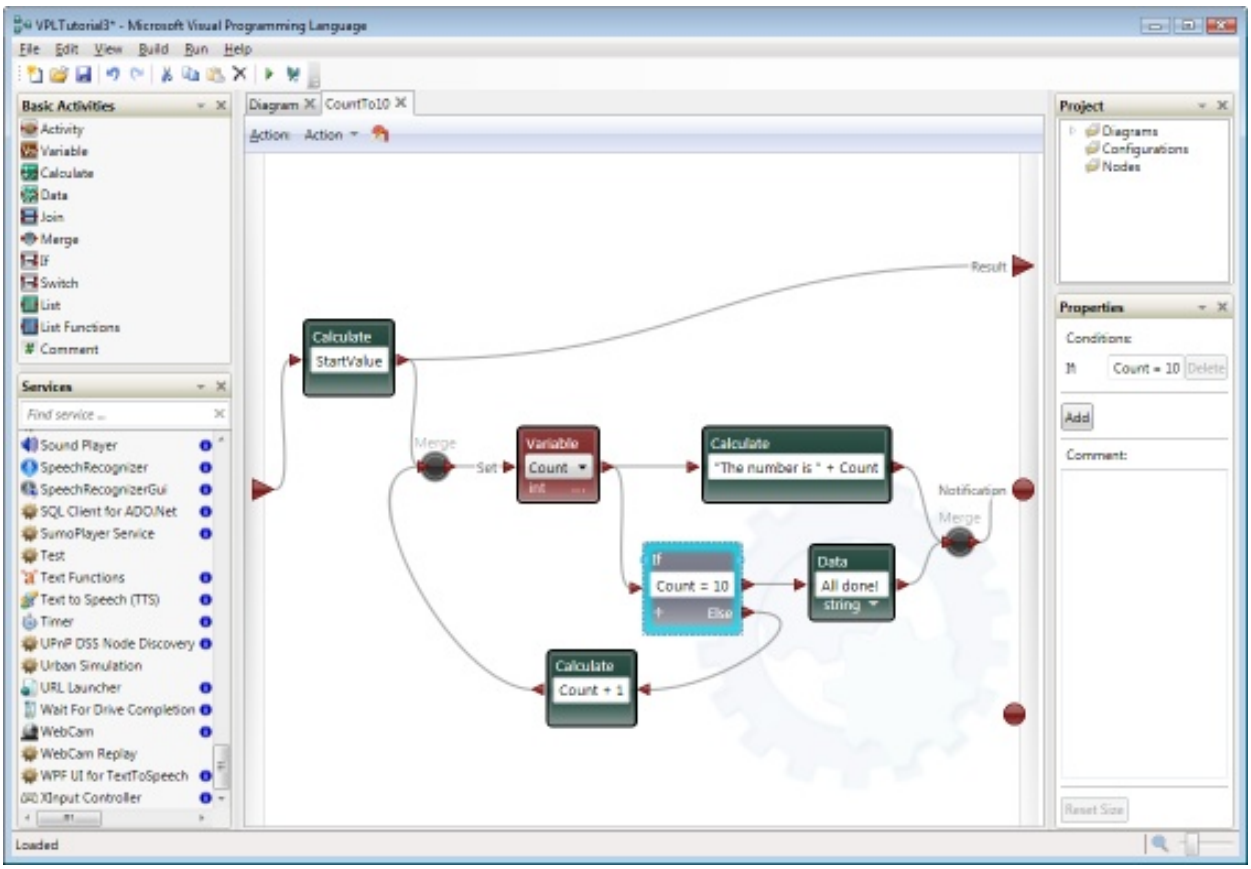

Рис. 3: Microsoft Visual Programming Language

2.3. LEGO Mindstorms EV3 Software

LEGO Mindstorms EV3 Software[3] — набор программ для обучения программированию роботов на основе контроллеров LEGO. В этот набор входит редактор для графического программированя роботов. В этом инструменте пользователь формирует программу из иконок, отвечающих за те или иные функции робота. Для наглядности блоки с одинаковыми функциями объединены в группы по цвету. Редактор изображен на рис. 4. LEGO Mindstorms EV3 Software работает под операционными системами Windows и Macintosh OS X.

[3]Программное обеспечение LEGO Mindstorms EV3, URL: `http://www.us.lego.com/ru-ru/mindstorms/downloads/download-software` (дата обращения 17.05.2015)

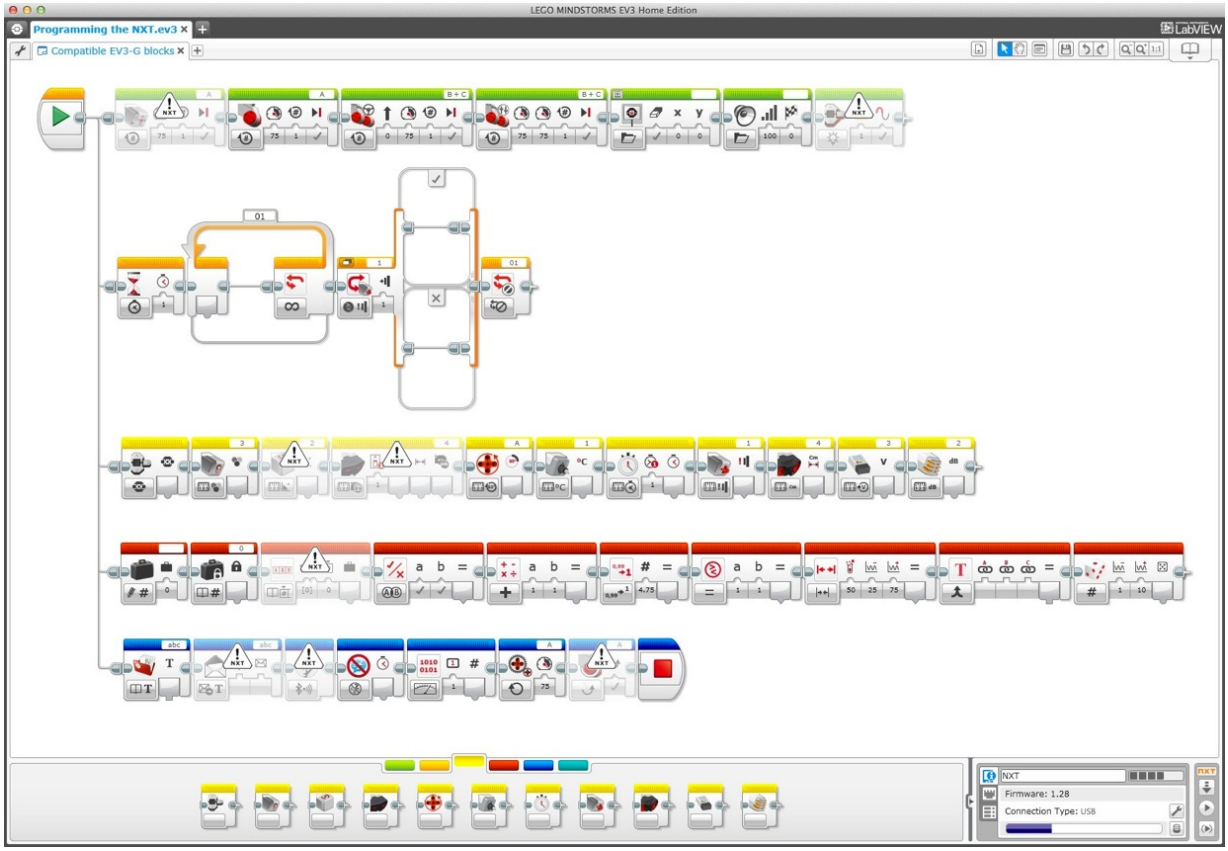

Рис. 4: LEGO Mindstorms EV3 Software

2.4. ROBOLAB-online

ROBOLAB-online[4] — онлайн среда для обучения программированию роботов. В ней имеется простая среда для задания программы робота, где можно изменять уже размещенные блоки. После задания программы можно просмотреть поведение робота в заданной модели мира. Данный инструмент использует плагин Adobe Shockwave Player[5], который доступен лишь для нескольких бразуеров в операционных системах Windows, Macintosh OS X и Macintosh PPC. На рис. 5 и 6 изображена данная среда.

[4]Среда ROBOLAB-online, URL: `http://www.robolabonline.com/home` (дата обращения 17.05.2015)

[5]Adobe Shockwave Player, URL: `http://get.adobe.com/shockwave/` (дата обращения 17.05.2015)

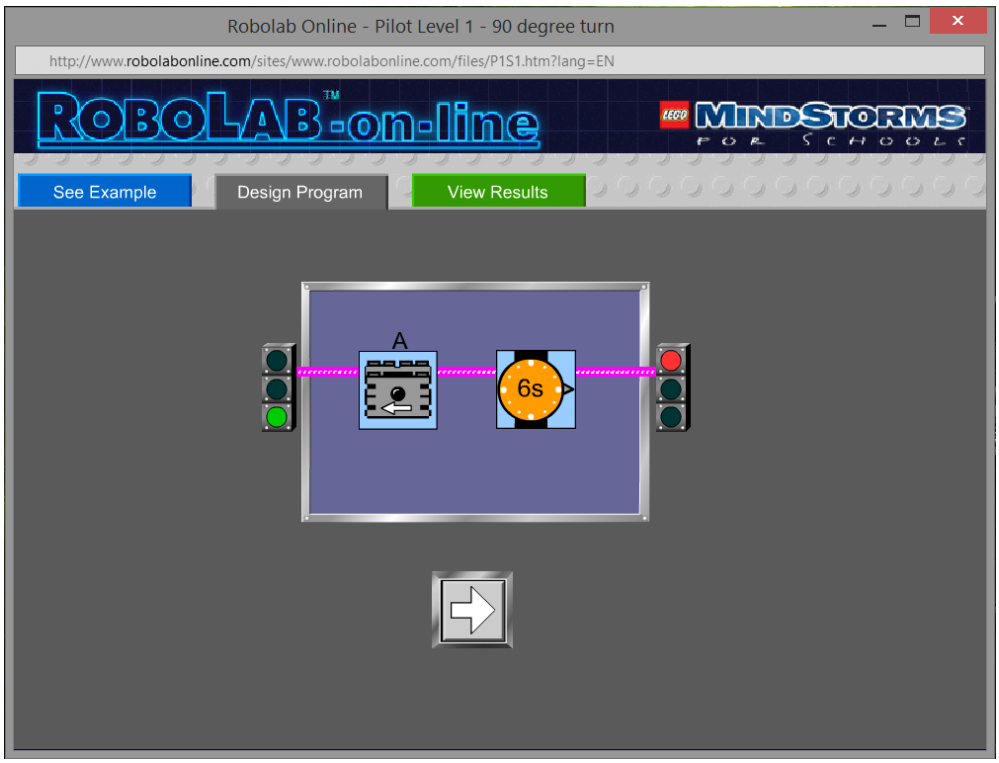

Рис. 5: ROBOLAB — Редактор программы

Рис. 6: ROBOLAB — Модель

2.5. Clooca

Clooca[6] — онлайн среда, разработанная японскими студентами в основном для образовательных целей[3]. Она позволяет создавать предметно-ориентированные языки моделирования и генераторы кода для них. Кроме того, в этой среде имеется редактор, в котором можно нарисовать модель на уже созданном языке и сгенерировать по ней исходный код. Редакторы представлены на рис. 7 и 8.

Одним из примеров использования данного инструмента может быть создание языка моделирования и генераторов для программирования роботов.

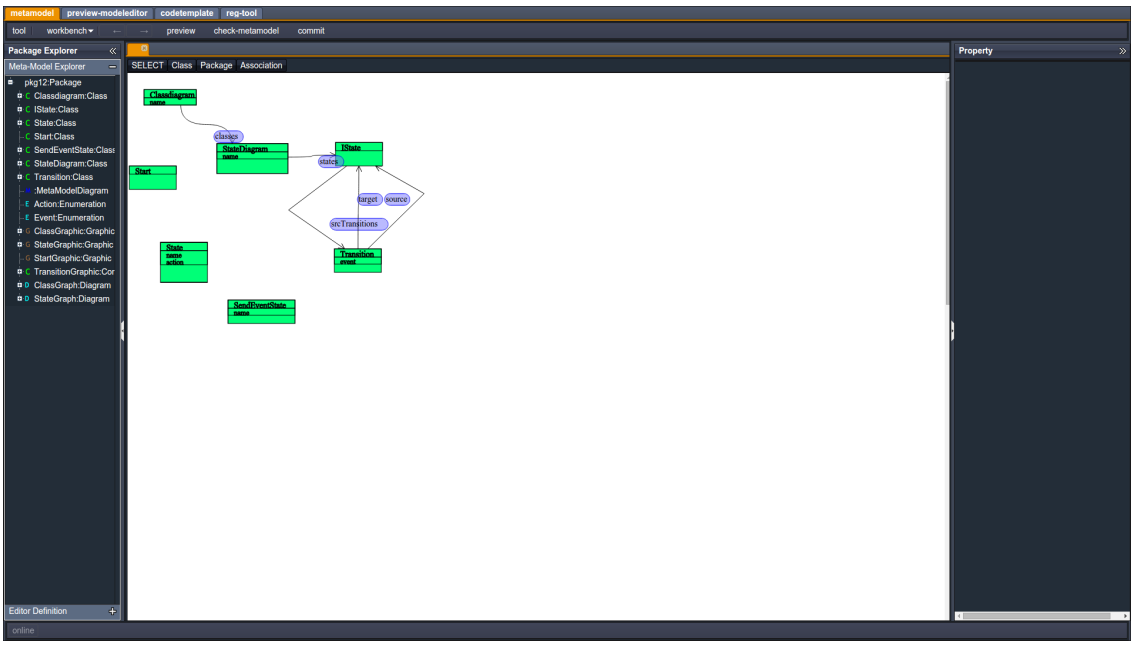

Рис. 7: Clooca — Среда разработки языков

[6]Среда Clooca, URL: `http://www.clooca.com/` (дата обращения 17.05.2015)

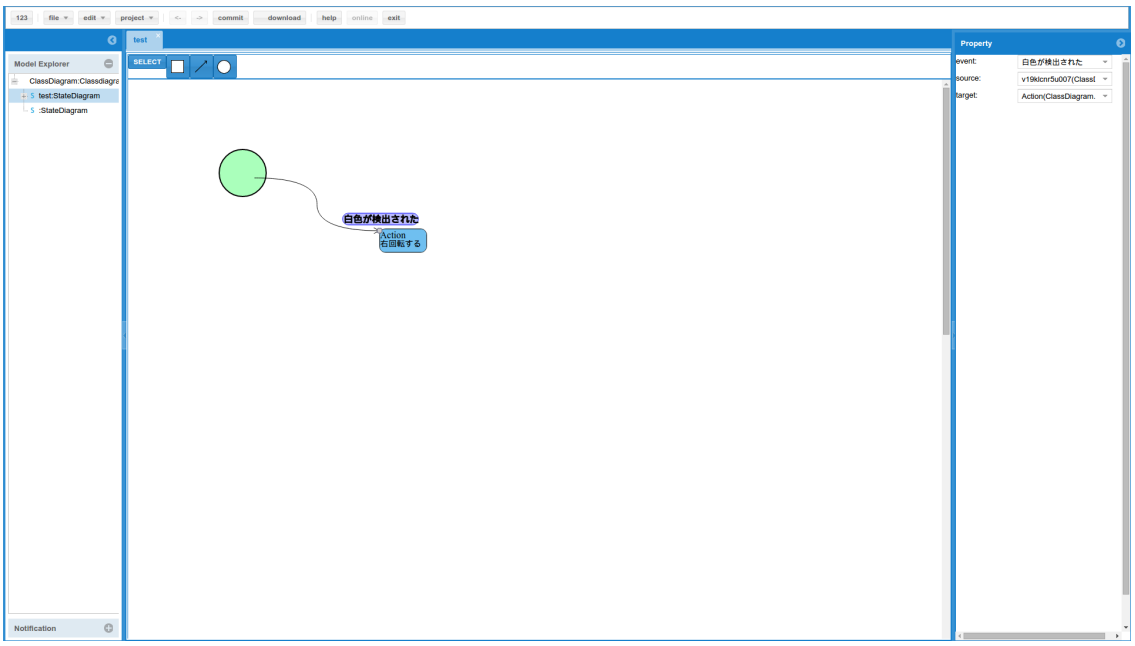

Рис. 8: Clooca — Редактор моделей

2.6. Выводы

После анализа этих решений были сделаны следующие выводы.

- Многие существующие решения требуют установки дополнительного программного обеспечения на компьютер.
- Не все инструменты являются кроссплатформенными.
- Существующие онлайн решения либо не обладают широкими возможностями, либо требуют дополнительных действий для программирования роботов.

Инструмент, реализуемый в рамках данной работы, является веб-приложением, представляющим собой полноценную среду для графического программирования роботов и имитации поведения робота в двумерной модели. Для использования инструмента требуется лишь наличие на устройстве современного браузера. Кроме того, данная среда с некоторыми изменениями может быть использована в онлайн курсах по программированию роботов и для создания многопользовательких соревнований по программированию роботов.

3. Предлагаемое решение

3.1. Выбор инструментов

3.1.1. Серверная часть

Для реализации серверной части приложения были выбраны технологии, описанные ниже.

В качестве основного языка программирования выбран язык Java[7]. Основной причиной данного выбора послужило существование для этого языка различных инструментов для быстрой и качественной разработки веб-приложения.

В качестве каркаса веб-приложения было решено взять Spring MVC Framework[8], реализующий шаблон проектирования Model-View-Controller. Этот фреймворк помогает отделить пользовательский интерфейс от бизнес-логики приложения. Неоспоримым плюсом является хорошая документация фреймворка. С деталями этого фреймворка можно ознакомиться в книге Spring in Practice[8].

За сборку проекта отвечает система сборки Maven[9][7]. Этот инструмент позволяет декларативно описать проект, управлять зависимостями, собирать проект из командной строки. Кроме того, Maven хорошо интегрируется со средами разработки.

Для конвертации Java-модели в формат JSON была взята библиотека Jackson[10]. Она зарекомендовала себя как быстрое и удобное решение.

Для объектно-реляционного отображения выбрана библиотека Hibernate[11][4]. Кроме отображения Java-классов в структуры базы данных, данная библиотека предоставляет средства для автоматического построения запросов и извлечения данных, что позволяет уменьшить время разработки, которое тратится на написание SQL и JDBC кода.

[7]Язык программирования Java, URL: `http://docs.oracle.com/javase/7/docs/api/` (дата обращения 17.05.2015)

[8]Spring MVC Framework, URL: `http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html` (дата обращения 17.05.2015)

[9]Система сборки Maven, URL: `https://maven.apache.org/` (дата обращения 17.05.2015)

[10]Библиотека Jackson, URL: `http://wiki.fasterxml.com/JacksonHome` (дата обращения 17.05.2015)

[11]Библиотека Hibernate, URL: `http://hibernate.org/` (дата обращения 17.05.2015)

Для управления базой данных используется СУБД MySQL[12].

В качестве веб-сервера был взят Apache Tomcat[13].

3.1.2. Клиентская часть

Для реализации клиентской части приложения были выбраны технологии, описанные ниже.

В качестве основного языка программирования выбран язык TypeScript[14][5]. TypeScript обратно совместим с JavaScript[15] и компилируется в него. Кроме того, TypeScript позволяет явно определять типы, поддерживает полноценные классы, добавляет поддержку модулей.

Для компиляции языка Typescript в JavaScript был выбран инструмент сборки Grunt[16]. Он позволяет следить за изменениями в TypeScript файлах и автоматически компилировать их.

Для удобного разделения пользовательского интерфейса и логики приложения был взят фреймворк AngularJS[17].

Для облегчения взаимодействия между JavaScript и HTML была выбрана библиотека jQuery[18][2].

Чтобы реализовать диаграмму поведения робота, была выбрана библиотека JointJs[19][6], позволяющая создавать интерактивные диаграммы.

Для реализации двумерной модели поведения решено использовать библиотеку Raphaël[20][1], которая упрощает работу с векторной графикой.

---

[12]Система управления базами данных MySQL, URL: `https://www.mysql.com/` (дата обращения 17.05.2015)

[13]Apache Tomcat, URL: `http://tomcat.apache.org/` (дата обращения 17.05.2015)

[14]Язык программирования TypeScript, URL: `http://www.typescriptlang.org/` (дата обращения 17.05.2015)

[15]Язык программирования JavaScript, URL: `http://javascript.ru/` (дата обращения 17.05.2015)

[16]Инструмент сборки Grunt, URL: `http://gruntjs.com/` (дата обращения 17.05.2015)

[17]Фреймворк AngularJS, URL: `https://angularjs.org/` (дата обращения 17.05.2015)

[18]Библиотека jQuery, URL: `http://jquery.com/` (дата обращения 17.05.2015)

[19]Библиотека JointJs, URL: `http://www.jointjs.com/` (дата обращения 17.05.2015)

[20]Библиотека Raphaël, URL: `http://raphaeljs.com/` (дата обращения 17.05.2015)

3.2. Архитектура приложения

Веб-приложение состоит из серверной и клиентской части, как показано на рис. 9. В зависимости от запроса серверная часть либо формирует веб-страницу и отправляет ее пользователю, либо обращается к базе данных для получения/занесения данных и возвращает результат пользователю, сериализованный в формат JSON.

Редактор диаграмм и двумерная модель поведения робота являются программами, выполняемыми на клиентской стороне. Для сохранения/загрузки состояния диаграммы посылается соответствующий Ajax запрос на сервер с сериализованными в формат JSON данными.

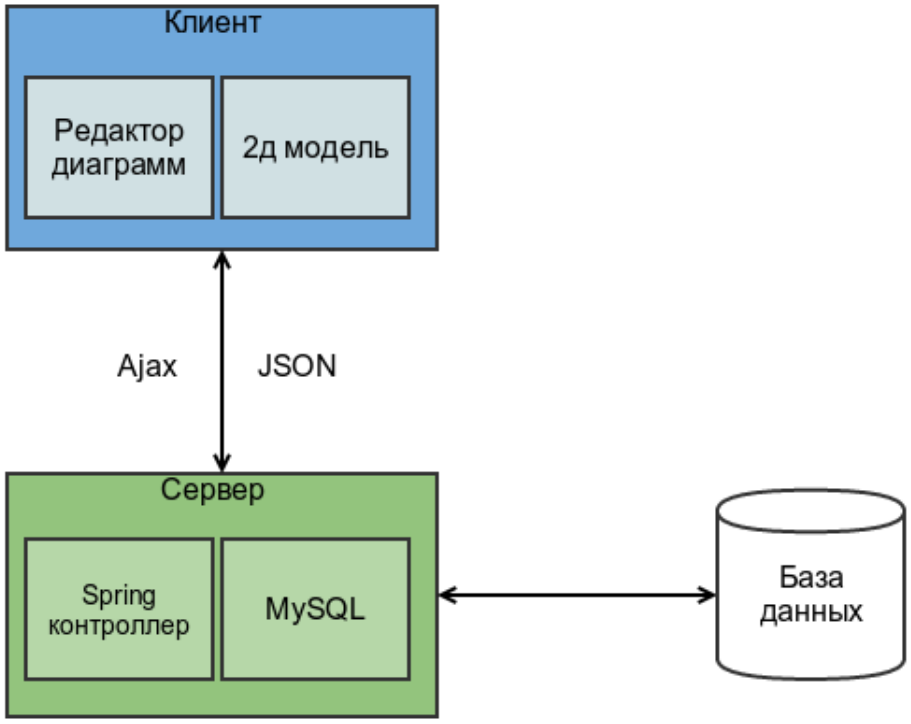

Рис. 9: Общая архитектура системы

Редактор диаграмм и двумерная модель расположены на одной веб-странице для обеспечения возможности передачи данных между друг другом, что необходимо для отображения в двумерной модели данных, получаемых при интерпретации диаграммы поведения робота.

Для разбиения функциональности, отвечающей за управление редактором диаграмм и двумерной моделью, было решено создать два

контроллера AngularJS. Проблема в том, что в AngularJS контроллеры имеют иерархическую структуру, и контроллеры на одном уровне не могут передавать данные друг другу. Для каждого контроллера создается объект $scope. Эти объекты связаны через механизм наследования прототипа. Таким образом, любой дочерний $scope может использовать методы и свойства его родителя. Поэтому было решено создать общий родительский контроллер, в его объекте $scope в свойстве root сделать ссылку на него, а это позволило в дочерних объектах $scope обращаться к контроллеру через свойство root и получать или заносить данные. Схема этого решения изображена на рис. 10.

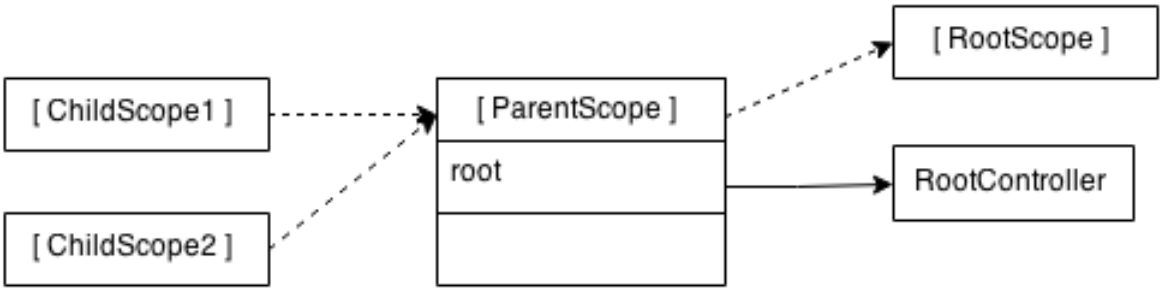

Рис. 10: Связь объектов scope

Здесь ChildScope1 и ChildScope2 обозначают объекты $scope, наследуемые через прототипное наследование от объекта ParentScope. Объект ParentScope имеет свойство root, которое является ссылкой на контроллер RootController.

3.3. Функционал созданного решения

3.3.1. Функционал серверной части

Для обработки запросов пользователей реализован Spring-контроллер. По запросу пользователя контроллер формирует веб-страницу и возвращает ее, либо сообщает об ошибке.

Для удобства работы с состоянием диаграммы реализована Java-модель. При помощи библиотеки Jackson она может быть конвертирована в формат JSON, а при помощи библиотеки Hibernate отображена

на структуры базы данных.

Для создания и заполнения базы данных был написан скрипт.

Для обработки запросов на сохранение и загрузку состояния диаграмм написан специальный сервис. Он обращается к объекту доступа к данным (Data access object[21]), который при помощи библиотеки Hibernate заносит состояние диаграммы в базу данных или извлекает его в зависимости от запроса. После этого сервис возвращает результат операции.

3.3.2. Функционал редактора диаграмм

Редактор диаграмм состоит из рабочей области — сцены, на которой размещаются элементы, палитры элементов, редактора свойств, верхней панели, на которой размещены элементы навигации и кнопки для сохранения/загрузки состояния диаграмм. Элементы помещаются на сцену из палитры при помощи метода "drag-and-drop", т.е. перетягиваются при помощи мыши. При нажатии на элемент появляются свойства этого элемента, которые можно изменить. Редактор диаграмм изображен на рис. 11

[21]Data access object, URL: `https://ru.wikipedia.org/?oldid=55257386` (дата обращения 18.05.2015)

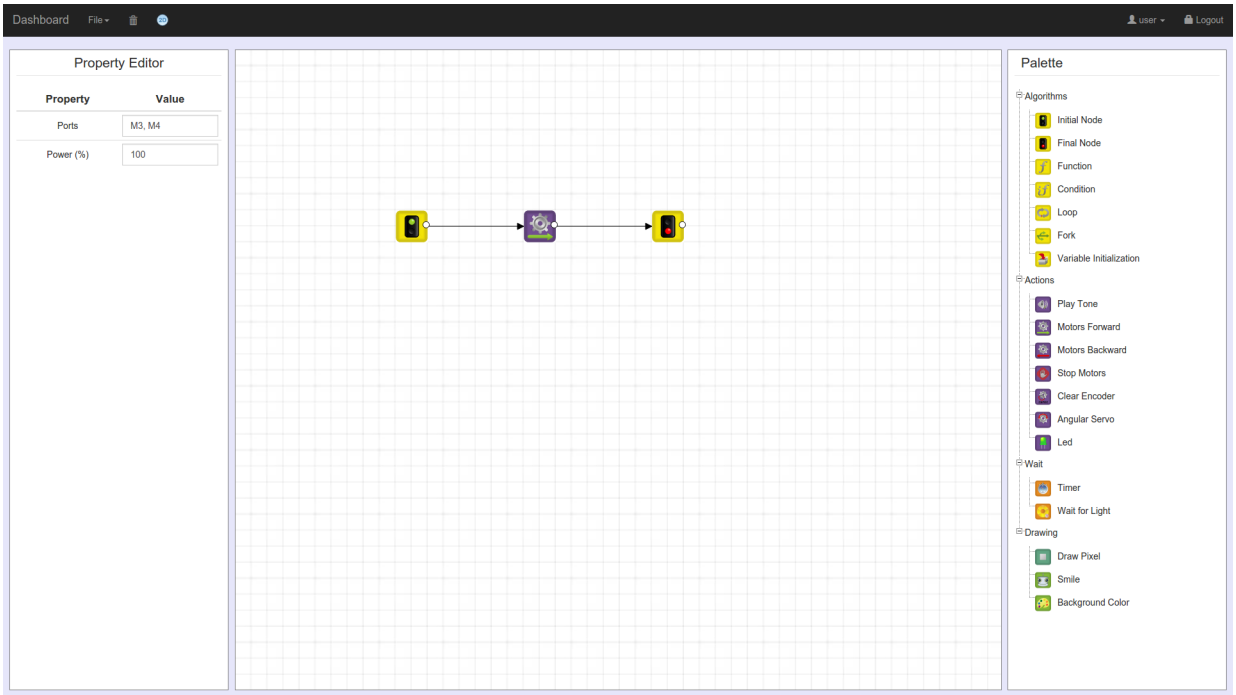

Рис. 11: Редактор диаграмм

Набор элементов палитры задается при помощи XML файла, находящегося на сервере. Когда пользователь открывает страницу с редактором, из этого файла берутся данные об элементах палитры. Далее опишем формат XML файла.

Пример части файла:

```
<Elements>
   <Category name="Algorithms">
      ...
      <Element name="Motors Forward" type="TrikV6EnginesForward">
         <Property key="Ports" name="Ports" type="string">
            <value>M3, M4</value>
         </Property>
         <Property key="Power" name="Power (%)" type="string">
            <value>100</value>
         </Property>
         <Image src="images/enginesForwardBlock.svg"/>
      </Element>
      ...
   </Category>
</Elements>
```

Элементы разделяются на категории, каждый элемент имеет имя, которое отображается пользователю, и тип. У элемента имеется список свойств и путь до соответствующего ему изображения.

Возможные типы Property: string, checkbox, dropdown. В зависимости от типа свойство будет отображаться как строка, флажок или выпадающий список соответственно.

Если тип свойства - dropdown, то должен быть список вариантов выбора в следующем виде:

```
<Property key="Sign" name="Sign" type="dropdown">
   <value>greater</value>
   <Variants>
      <variant>greater</variant>
      <variant>less</variant>
      <variant>not greater</variant>
      <variant>not less</variant>
   </Variants>
</Property>
```

Variants - список возможных для выбора вариантов. При этом значение value должно совпадать с одним из вариантов выбора.

Сцена редактора диаграмм реализована при помощи библиотеки JointJs. Для создания диаграммы необходимо иметь элементы с некоторым изображением и портом для создания линий соединения. В библиотеке есть стандартная возможность создавать блоки с картинкой или с портом для создания линий, но не одновременно. В связи с этим был создан свой элемент, имеющий возможности обоих этих блоков. Кроме того, линия, созданная при помощи порта, должна соединять 2 элемента диаграммы, но она соединяла порт и элемент. Также у линии соединения имеются свойства, поэтому при создании линии было необходимо создавать объект, в котором хранятся текущие значения свойств. В связи с этим для блоков диаграммы было изменено поведение создания линии при помощи порта на необходимое.

Для сохранения состояния диаграммы данные об элементах, расположенных на сцене, сериализуются в формат JSON. При сохранении у пользователя запрашивается имя диаграммы, которое помещается в

свойство name. Элементы диаграммы помещаются в массив nodes. Каждый элемент имеет идентификатор элемента диаграммы, тип, позицию на сцене, массив свойств properties. Свойства имеют имя, текущее значение, тип и позицию в отображаемом списке свойств. Линии, соединяющие элементы, помещаются в массив links. Каждая линия имеет идентификатор элемента диаграммы, свойства source и target, набор вершин, массив свойств properties. В свойстве source находится имя элемента, из которого выходит линия, а в свойстве target — имя элемента, в который входит линия. Каждая вершина имеет позицию на сцене и порядковый номер. Свойства имеют такой же формат, как и у элементов диаграммы. Пример формата изображен ниже.

```
{
   "name":"diagram",
   "nodes":[
      ...
      {
         "jointObjectId":"2b97e2ab-1aaf-4ec4-8af7-46bb59eca040"
         "type":"Motors Forward",
         "x":475,
         "y":175,
         "properties":[
            {
               "name":"Ports",
               "value":"M3, M4",
               "type":"string",
               "position":1
            },
            ...
         ]
      },
      ...
   ],
   "links":[
      ...
      {
         "jointObjectId":"708450a7-ede6-403f-ad3e-ce0f53704878",
         "source":"2b97e2ab-1aaf-4ec4-8af7-46bb59eca040",
```

```
            "target":"9836f46a-e663-44fc-8b04-e2c546f92c82",
            "vertices":[
               {
                  "x":300,
                  "y":225,
                  "number":1
               }
            ],
            "properties":[
               {
                  "name":"Guard",
                  "value":"",
                  "type":"dropdown",
                  "position":1
               }
            ]
         },
         ...
      ]
}
```


Далее эти данные отправляются на сервер при помощи Ajax запроса. На сервере они автоматически переводятся в Java-модель при помощи библиотеки Jackson. После этого при помощи библиотеки Hibernate и MySQL данные заносятся в базу данных.

Чтобы загрузить диаграмму, пользователь вводит имя необходимой ему диаграммы. Далее на сервер посылается Ajax запрос, с указанным именем. Если диаграмма с таким именем существует, то данные извлекаются из базы данных, преобразуются в Java-модель, сериализуются в формат JSON и отправляются обратно пользователю. На клиентской стороне данные разбираются и отображаются пользователю на сцене. Если диаграммы с заданным именем не существует, выводится ошибка.

3.3.3. Функционал двумерной модели

Двумерная модель состоит из сцены, левой панели с кнопками переключения между палитрой, конфигурацией портов, настройками, верх-

ней панели, на которой размещены элементы навигации. На сцене помещаются модель робота с сенсорами, стены, цветные фигуры. Стены и цветные фигуры рисуются на сцене при помощи движения мыши. Все элементы, расположенные на сцене, можно перемещать, вращать. Стены и цветные фигуры можно также изменять в размерах. Имеется возможность менять цвет и ширину линии цветных фигур. Двумерная модель изображена на рис. 12.

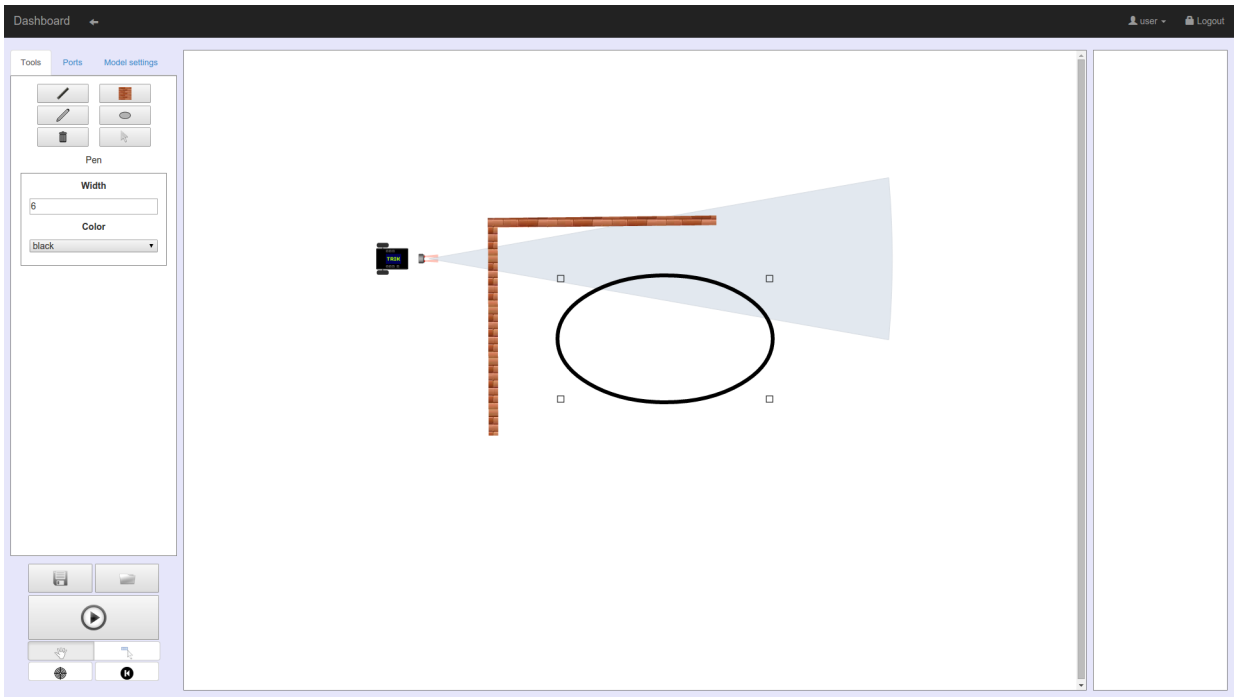

Рис. 12: Двумерная модель

Стояла задача разработать довольно гибкую и удовлетворяющую всем потребностям архитектуру приложения для двумерной модели. Ниже описан полученный результат, изображенный на рис. 13.

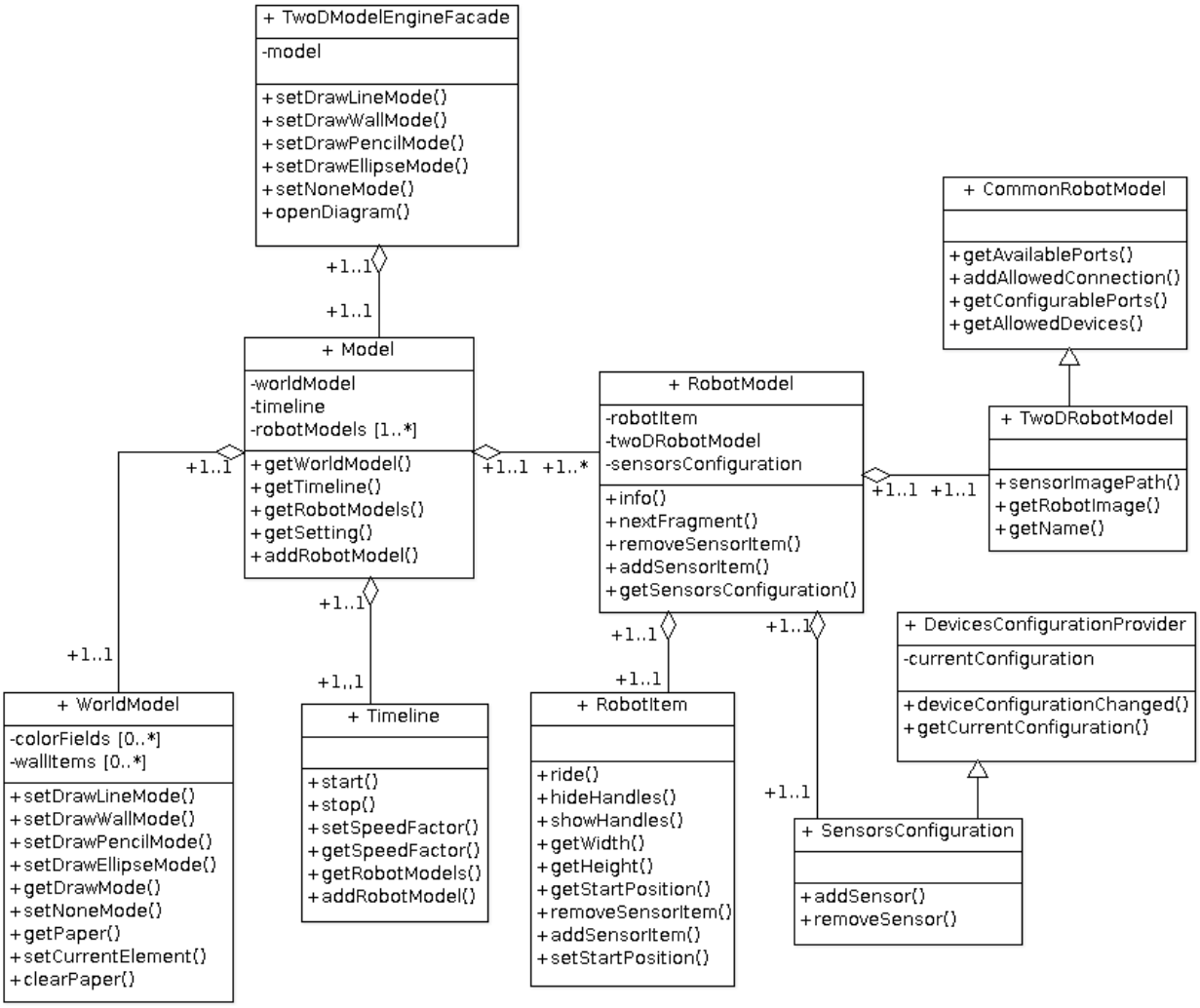

Рис. 13: Архитектура двумерной модели

Чтобы скрыть сложность системы, используется шаблон проектирования Фасад[10]. Все внешние взаимодействия с двумерной моделью проходят через него. Класс Model отвечает за коммуникацию между частями приложения. WorldModel содержит сцену, цветные фигуры, стены. Он отвечает за рисование и изменение этих фигур. Класс Timeline служит для контролирования скорости отрисовки движения робота. Класс RobotModel необходим для хранения информации о модели робота. В объекте этого класса содержится экземпляр класса SensorConfiguration, где находится информация о конфигурации сенсоров, а также имеется ссылка на двумерную модель робота. Класс RobotItem служит для графического представления модели робота. Двумерная модель TwoDRobotModel, наследуемая от класса

CommonRobotModel, содержит информацию о портах, которые можно конфигурировать, устройствах, которые можно подключать к этим портам, а также имя модели и пути к изображениям для данной модели робота.

Такая архитектура позволяет легко изменить поведение модели робота, добавлять новые возможности. Например, легко изменить набор портов и устройств, заменив двумерную модель робота на другую, или просто добавить новые устройства в список доступных устройств, графическое представление робота нетрудно изменить заменой класса RobotItem.

Иерархия объектов, которые могут быть размещены на сцене, изображена на рис. 14.

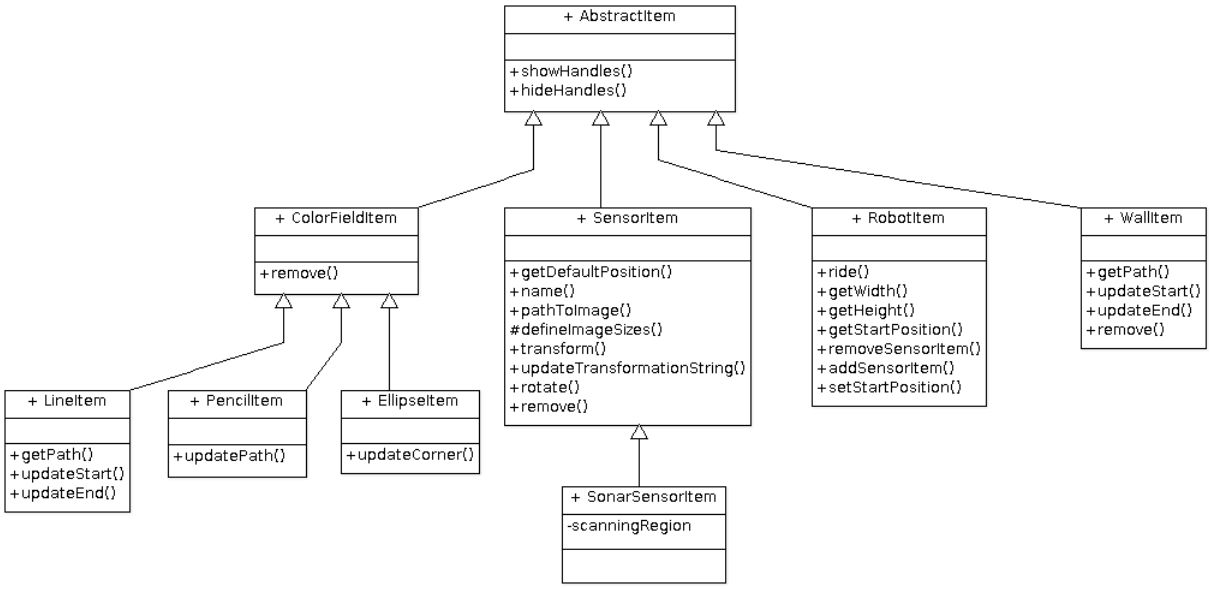

Рис. 14: Иерархия объектов двумерной модели

Чтобы вращать и изменять объекты, размещенные на сцене, для них были созданы обработчики, при перетаскивании которых объекты изменяются. Обработчики становятся видимыми, когда пользователь выделяет объект, и скрываются, когда выделяется другой элемент или пользователь кликает на пустое место сцены.

4. Внедрение

Летом 2015 года планируется запустить онлайн-курс по основам робототехники на базе образовательной платформы Stepic[22]. Для некоторых задач из этого курса возможно применить реализуемую в данной работе среду программирования роботов с некоторыми изменениями. Тогда пользователь сможет выполнять задания, не выходя из браузера, что, несомненно, удобнее, чем решать их в отдельной программе.

Рассмотрим последовательность действий, совершаемых пользователем, и ответы среды на эти действия. Диаграмма последовательности приведена на рис. 15.

Рис. 15: Диаграмма последовательности действий

Пользователь выбирает задание, по идентификатору задания сервер отвечает частично готовой диаграммой поведения робота, палитрой элементов и полем двумерной модели. После этого пользователь изменяет диаграмму и свойства элементов. Закончив изменение, он подтверждает окончание работы. Далее созданная диаграмма отправляется на

[22]Образовательная платформа Stepic, URL: `https://stepic.org/` (дата обращения 17.05.2015)

сервер, переводится в формат сохранений TRIK Studio и проверяется специальной программой. Эта программа возвращает результат проверки и набор положений робота с временными штампами, которые он должен пройти в двумерной модели. Эти данные возвращаются в онлайн-среду, где отображается движение робота и сообщается результат проверки.

Диаграмма поведения робота и поле двумерной модели хранятся в формате сохранений TRIK Studio. Это сделано для удобства формирования заданий. Диаграмма поведения хранится в запакованном виде. Для упаковки и распаковки используется инструмент, написанный для проекта QReal[23].

Палитра элементов для задания определяется при помощи xml файла, формат которого совпадает с форматом набора элементов в полноценной среде.

Двумерная модель является нередактируемой, она служит лишь для отображения результата проверки. В связи с этим были убраны панели, предназначенные для рисования объектов и конфигурации портов.

Без лишних панелей двумерная модель стала занимать меньше места, а это дало возможность разместить редактор диаграмм и двумерную модель на одной странце, что придает большую наглядность. Полученный результат среды для онлайн-курса изображен на рис. 16.

[23]Утилита compressor, URL: `https://github.com/qreal/tools/tree/master/compressor` (дата обращения 17.05.2015)

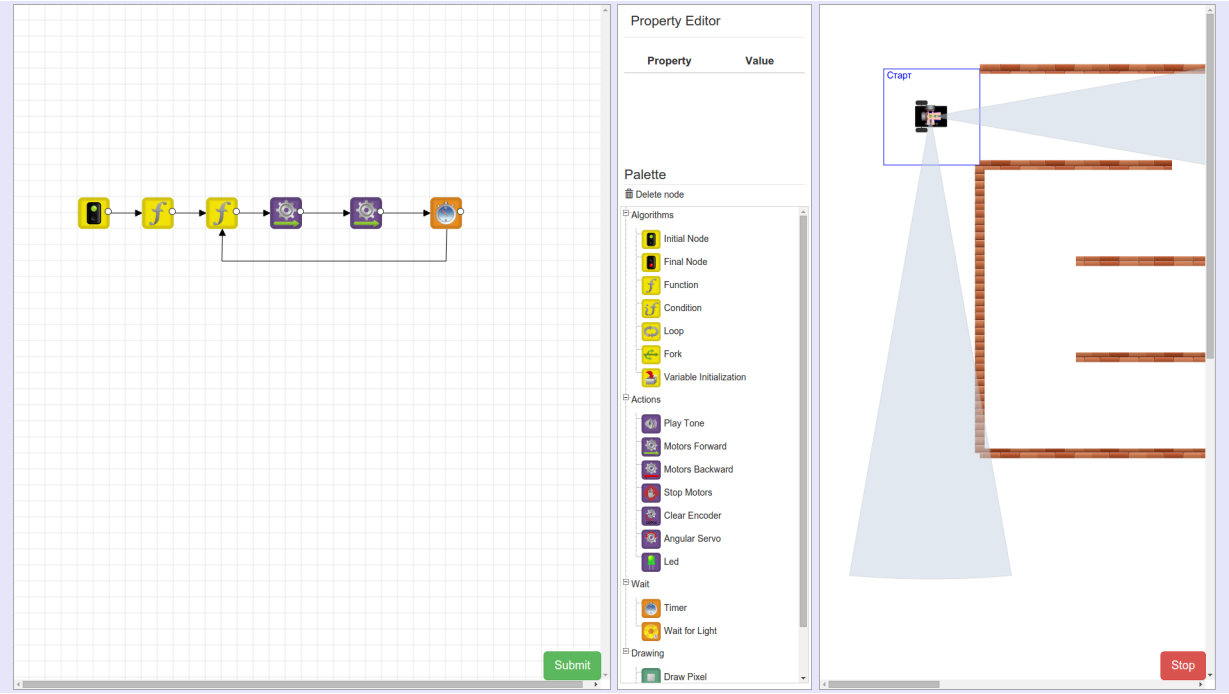

Рис. 16: Среда для онлайн-курса

Результаты

В ходе работы были проанализированы существующие средства графического программирования роботов. Была реализована онлайн среда, позволяющая задавать поведение робота при помощи диаграмм. Разработан редактор диаграмм и двумерная модель поведения робота.

Ниже перечислены результаты данной работы.

- Проанализированы существующие решения и сделан вывод о недостатках этих решений.
- Выбраны подходящие для реализации инструменты.
- Разработана общая архитектура онлайн среды графического программирования роботов.
- Реализована серверная часть приложения.
- Реализован редактор диаграмм поведения робота.
- Реализована двумерная модель поведения робота.
- Создан инструмент для проверки заданий для онлайн-курса на базе платформы Stepic.

Список литературы

[1] Baranovskiy Dmitry. Raphaël documentation. — URL: http://raphaeljs.com/reference.html (online; accessed: 08.05.2015).

[2] Chaffer Jonathan, Swedberg Karl. Learning jQuery. — 4 edition. — Packt Publishing, 2013. — P. 444.

[3] Clooca : Web based tool for Domain Specific Modeling / Shuhei Hiya, Kenji Hisazumi, Akira Fukuda, Tsuneo Nakanishi // CEUR Workshop Proceedings. — 2013. — Vol. 1115. — P. 31–35.

[4] Elliott James, O'Brien Timothy M., Fowler Ryan. Harnessing Hibernate. — O'Reilly Media, 2008. — P. 382.

[5] Fenton Steve. TypeScript For JavaScript Programmers [Kindle Edition]. — 4 edition. — Swift Point Press, 2012. — P. 74.

[6] JointJs API. — URL: http://www.jointjs.com/api (online; accessed: 11.05.2015).

[7] O'Brien Timothy, Casey John, et al. Maven: The Complete Reference. — Sonatype Inc., 2011. — P. 315.

[8] Wheeler Willie, White Joshua. Spring in Practice. — Manning Publications Co, 2013. — P. 560.

[9] А.Н. Терехов, Ю.В. Литвинов, Т.А. Брыксин. Среда визуального программирования роботов QReal:Robots // III Всероссийская конференция «Современное технологическое обучение:от компьютера к роботу» (сборник тезисов). — СПб., 2013. — С. 1–4.

[10] Приемы объектно-ориентированного проектирования. Паттерны проектирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. — Питер, 2015. — С. 366.

[11] Ю.В. Литвинов. Применение DSM-платформы QReal при разработке среды программирования роботов QReal:Robots // Системное программирование, Вып. 7. — СПб., 2012. — С. 161–186.

[12] Ю.В. Литвинов, Я.А. Кириленко. TRIK Studio: среда обучения программированию с применением роботов // V Всероссийская конференция «Современное технологическое обучение: от компьютера к роботу» (сборник тезисов). — СПб., 2015. — С. 5–7.