

Санкт-Петербургский государственный университет информационных
технологий, механики и оптики

Кафедра «Компьютерные технологии»

А.В. Шестаков, К.А. Лиференко

**Отчет по курсовой работе
«Создание платформы для проведения виртуальных
лабораторных работ»**

Санкт-Петербург

2011 г.

Оглавление

Введение	3
1. Постановка задачи	5
2. Реализация	6
2.1. Сайт виртуальной лаборатории.....	6
2.1.1. Реализация сайта виртуальной лаборатории	6
2.1.2. Описание интерфейса пользователя.....	8
2.1.3. Описание интерфейса администратора	10
2.2. Лабораторная работа по рефакторингу автоматных программ ..	18
2.2.1. Реализация дополнения к UniMod	19
2.2.2. Интерфейс дополнения к UniMod	24
2.2.3. Пример рефакторинга	26
2.2.4. Пример отчета	28
2.3. Лабораторная работа по контрактному программированию.....	29
Заключение	31
Источники.....	32

Введение

Сегодня многие учебные заведения используют инновационные технологии в образовательной среде, в том числе виртуальные лабораторные работы. Этот подход позволяет студенту в удобной для него форме проходить обучение и тестирование, а преподавателю проверять знания студентов. Задачей данной работы является создание платформы (сайта), реализующей три основные функции, которые необходимо реализовать для проведения виртуальных лабораторных работ. Этими функциями являются: описание темы работы, проверочное тестирование по теме, выполнение задания лабораторной работы.

На разработанном сайте студент получает возможность ознакомиться с темой лабораторной работы, пройти проверочное тестирование по теме в форме вопросов и ответов и скачать задание. Для выполнения задания используется то или иное инструментальное средство, соответствующее теме лабораторной работы. Это инструментальное средство не относится к платформе.

После выполнения задания студент должен создать отчет в формате, определяемом платформой. Например, для лабораторной работы по рефакторингу автоматных программ этот отчет должен содержать результат проведения рефакторинга автомата в среде *UniMod* и текстовый документ со списком проведенных рефакторингов. Этот отчет необходимо отправить с помощью разработанного сайта на проверку преподавателю. Если работа выполнена правильно, то преподаватель письмом уведомляет студента о завершении работы. В противном случае преподаватель уведомляет студента о необходимости продолжения работы.

В рассматриваемой работе в качестве примеров на предлагаемой платформе реализуется две виртуальные лаборатории: по рефакторингу [1, 2] автоматных программ, построенных с помощью инструментального средства *UniMod*, и применению контрактов для автоматных программ, построенных в

среде *MPS* [3, 4]. В первом случае к базовой функциональности среды разработки *UniMod* необходимо добавить возможность рефакторинга автоматов, а во втором – к среде *MPS* ничего добавлять не требуется.

Автоматный подход [5] – один из способов программирования объектов со сложным поведением. Этот подход применим при решении широкого круга задач, его основным принципом является явное разделение программы на управляющий автомат и объекта управления. Управляющий детерминированный конечный автомат [5] описывает поведение программы.

Рефакторинг автоматных программ [2] – изменение программ, которое не изменяет их поведение, но улучшает их структуру. Так как логика программы вынесена в управляющий автомат, то рефакторинг автоматной программы сводится к рефакторингу управляющего автомата. Такими рефакторингами являются:

- группировка состояний;
- удаление группы состояний;
- слияние состояний;
- выделение автомата;
- встраивание вызываемого автомата;
- перемещение воздействия из состояния в переходы;
- перемещение воздействия из переходов в состояние.

Применение контрактов для автоматных программ [4] – это подход к проектированию программ, при использовании которого, спецификация к интерфейсу модулей программы задается с помощью контрактов. Под контрактами обычно понимают совокупность способов формализации и проверки требований к программе. В указанную совокупность входят инварианты, постусловия и предусловия.

1. Постановка задачи

Цель курсовой работы – разработка платформы (далее – виртуальной лаборатории), для проведения виртуальных лабораторных работ. Предполагается, что такая лабораторная работа должна включать:

- ознакомление с материалом;
- тестирование;
- выполнение лабораторной работы.

Студенты должны иметь возможность ознакомиться с материалом, пройти проверочное тестирование и выполнить лабораторную работу.

На этой платформе в качестве примеров в настоящей работе должны быть реализованы лабораторные работы по рефакторингу автоматных программ и по применению контрактов для автоматных программ. При этом, как отмечено выше, для лабораторной работы по рефакторингу автоматных программ необходимо реализовать плагин, позволяющий проводить рефакторинг автоматов в среде *UniMod*.

2. Реализация

Для решения поставленной задачи был создан сайт <http://sevirlab.appspot.com>. На нем студенты могут ознакомиться с темой лабораторной работы, найти ссылки для ознакомления с материалом, пройти тестирование, а также скачать условие лабораторной работы и отправить отчет. Администратор может редактировать существующие лабораторные работы и добавлять новые. Для реализации сайта была выбрана платформа *Google App Engine* [6].

Для реализации среды для рефакторинга было решено добавить требуемую функциональность в плагин для *Eclipse* [7] – *UniMod* [8]. *UniMod* предназначен для разработки управляющих конечных автоматов, в то время как управляемыми объектами являются объекты *Java*. *UniMod* обладает широкими возможностями для редактирования автоматов, однако в нем не предусмотрена возможность автоматического проведения рефакторинга.

Среда для контрактного программирования уже реализована в работе [4] на базе *MPS (Meta Programming System)* [9].

2.1. Сайт виртуальной лаборатории

Сайт виртуальной лаборатории предназначен для обучения и тестирования студентов. Он обладает необходимой функциональностью для добавления новых лабораторных работ и контроля существующих.

2.1.1. Реализация сайта виртуальной лаборатории

Хостинг сайта реализован на облачной платформе *Google App Engine* [6]. Исходный код написан на языке *Java* для *App Engine* версии 1.4.0. Выполнение приложений *Java* осуществляется с помощью виртуальной машины *Java 6 (JVM)*, а доступ к классам стандартной библиотеки *Java* ограничен «белым

списком» платформы. *Google App Engine* использует для веб-приложений стандарт *Java Servlet*. Описание структуры приложения приведено ниже.

Каждая страница имеет два различных представления: для студентов и администраторов. Отображаемые элементы и права доступа определяются по *Google Account*, поэтому для регистрации подходят только адреса, зарегистрированные в системе *Gmail*. Все администраторы должны быть добавлены в список владельцев приложения в панели администратора. Студенты имеют доступ только к тем лабораторным работам, в которых они зарегистрированы. Для этого в каждую лабораторную работу загружаются списки студентов, включающие их адреса электронной почты.

В стандартной структуре каталогов *WAR* представляются классы сервлетов приложения, статические файлы, дескриптор развертывания (файл `web.xml`) и другие файлы конфигурации. Все веб-страницы генерируются в сервлетах, которые находятся в пакете `pg.page`. При добавлении нового сервлета, необходимо зарегистрировать его в файле `war\web-inf\web.xml`. Класс `pg.util.Front` используется для генерации шаблона внешнего вида страниц. Файл `war\style.css` задает таблицу стилей.

Класс `pg.util.Persis` отвечает за работу с базой данных *Google BigTable*, через интерфейс *JDO*. Методы `save()`, `load()`, `delete()` позволяют сохранять, загружать и удалять объекты, наследуемые от класса `pg.util.SerializableMarker` по уникальному строковому идентификатору. В базе данных хранится список лабораторных работ (класс `pg.util.LabList`) и информация по каждой лабораторной работе (класс `pg.util.LabInfo`). Так же для всех лабораторных работ хранится тест (класс `pg.util.LabTest`), изображения для теста (класс `pg.util.PicHolder`) и информация о студентах (классы `pg.util.Student` и `pg.util.StudentLog`).

2.1.2. Описание интерфейса пользователя

При входе на сайт виртуальной лаборатории пользователю предлагается выбрать одну из существующих лабораторных работ (рис. 1).

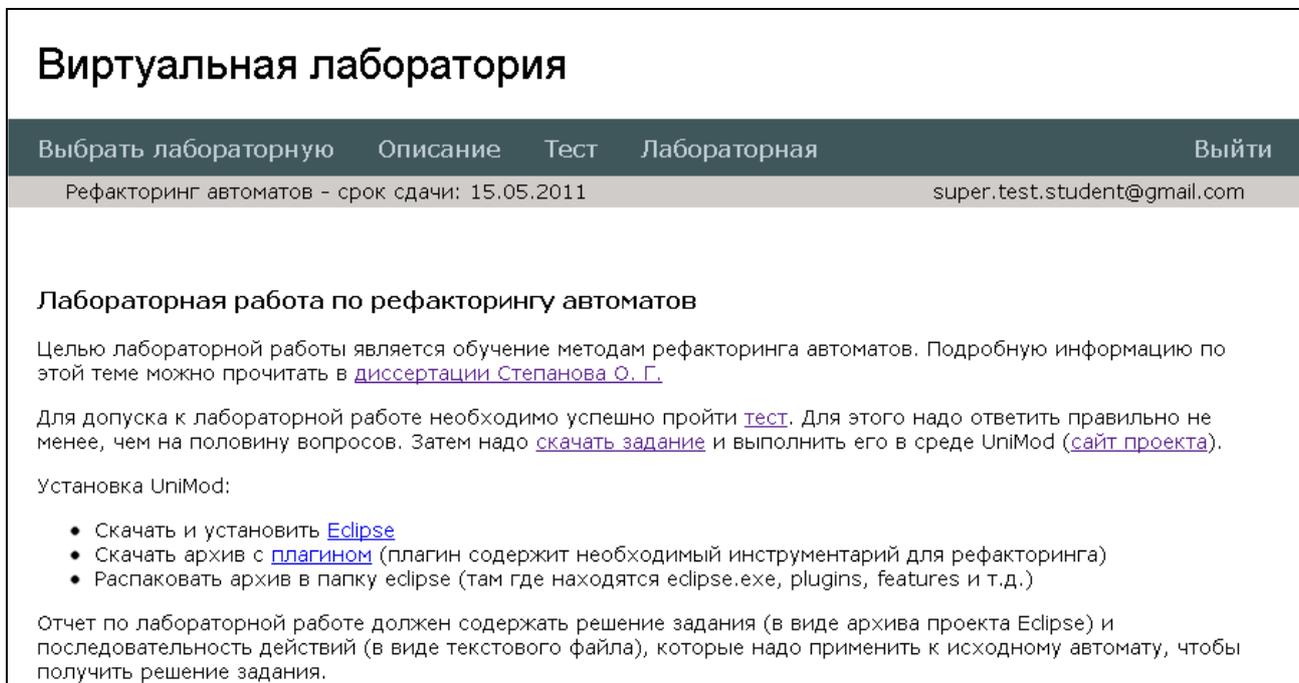


The screenshot shows a web interface for a virtual laboratory. At the top, there is a header with the text "Виртуальная лаборатория" and a "Выйти" button on the right. Below the header, there is a prompt "Выберите лабораторную работу". Underneath, there is a dropdown menu with "Рефакторинг автоматов" selected and a "Выбрать" button next to it.

Рис. 1. Страница выбора лабораторной работы

Каждая лабораторная работа состоит из следующих страниц:

- Страница с описанием лабораторной работы (рис. 2). Здесь пользователь может ознакомиться с целью работы, найти материалы по теме, узнать, как можно пройти тест, скачать задания и отправить отчет.



The screenshot shows a detailed page for a laboratory work. The header includes "Виртуальная лаборатория" and "Выйти". Below the header, there is a navigation bar with "Выбрать лабораторную", "Описание", "Тест", "Лабораторная", and "Выйти". The main content area displays "Рефакторинг автоматов - срок сдачи: 15.05.2011" and "super.test.student@gmail.com". The title is "Лабораторная работа по рефакторингу автоматов". The text describes the goal of the work and provides instructions for access, including a link to a dissertation and a test. It also lists installation steps for UniMod, such as downloading Eclipse and plugins. Finally, it states that the report should contain a solution to the task in the form of an Eclipse project archive and a sequence of actions in a text file.

Рис. 2. Описание лабораторной работы

- Страница с тестом (рис. 3). Здесь пользователь может пройти тестирование по теме. Прохождение теста обязательно и является допуском к лабораторной работе.

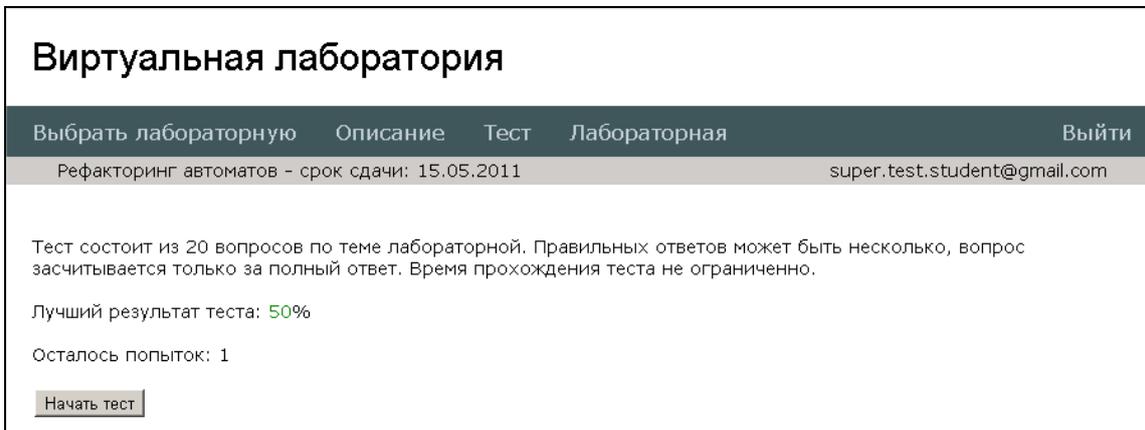


Рис. 3. Страница тестирования

При нажатии на кнопку «Начать тест» открывается страница с вопросами (рис. 4), ответ на вопрос теста необходимо отметить флажком. Для окончания теста необходимо нажать кнопку «Готово» внизу страницы. Результат выводятся сразу же, удовлетворительным считается результат, когда пользователь дал правильные ответы не менее чем на половину вопросов.

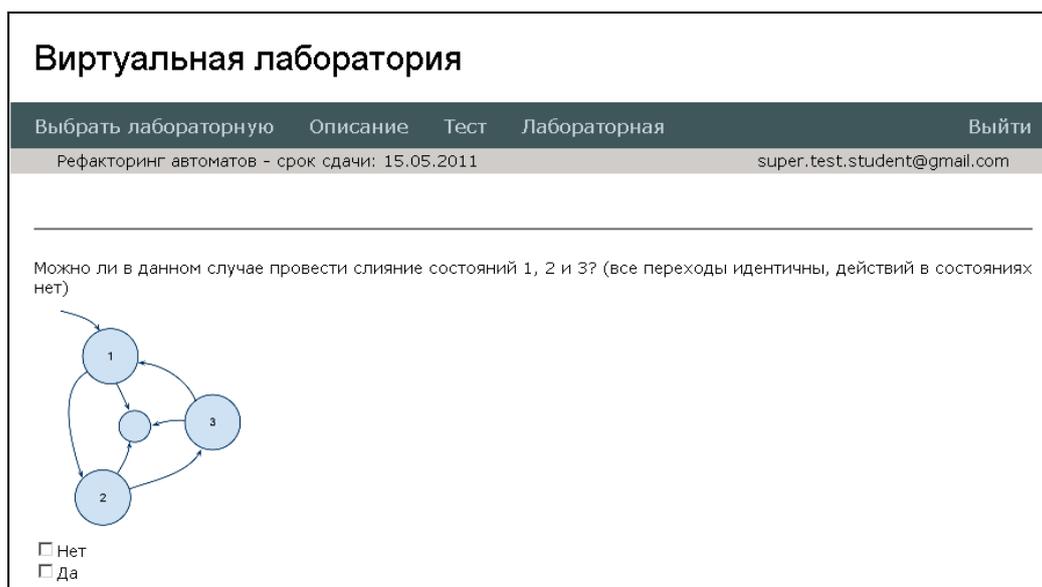


Рис. 4. Страница с тестом

- Страница с заданием (рис. 5). Здесь пользователь может скачать задание лабораторной работы (только в том случае, если он удовлетворительно прошел тестирование), а также отправить отчет на проверку.

Виртуальная лаборатория

Выбрать лабораторную Описание Тест Лабораторная Выйти

Рефакторинг автоматов - срок сдачи: 15.05.2011 super.test.student@gmail.com

[Скачать](#) персональное задание

Послать решение на проверку (это должен быть архив с решением и отчетом размером менее 1 мегабайта)

Выберите файл Файл не выбран Загрузить

Рис. 5. Страница с заданием

2.1.3. Описание интерфейса администратора

Администратор обладает правами на добавление новых лабораторных работ, изменения уже существующих и регистрацию студентов. При входе администратору предлагается выбрать лабораторную работу (рис. 6) для дальнейшего редактирования, или изменить список имеющихся работ (рис. 7). При удалении лабораторной работы безвозвратно удаляется вся связанная с ней информация.

Виртуальная лаборатория

Выйти

Выберите лабораторную работу

Рефакторинг автоматов Выбрать

[Настроить лабораторию](#)

Рис. 6. Страница выбора лабораторной работы, для изменения списка работ требуется перейти по ссылке «Настроить лабораторию»

Виртуальная лаборатория

Выйти

Удаление лабораторных (данные будут утеряны безвозвратно)

Рефакторинг автоматов (id = 1)

Удалить

Добавить новую лабораторную

Название лабораторной: Создать

[Готово](#)

Рис. 7. Страница изменения списка лабораторных работ, здесь можно создать новую или удалить существующую работу

После выбора лабораторной работы открывается страница с описанием (рис. 8), она почти идентична странице с описанием, которую видит пользователь, за исключением кнопки «Остановить лабораторию». При нажатии на эту кнопку выбранная лабораторная работа останавливается, и администратор может начать редактировать информацию (рис. 9). Когда лабораторная работа остановлена, студенты не имеют к ней доступа.

Виртуальная лаборатория

Выбрать лабораторную

Описание

Тест

Лабораторная

Выйти

Рефакторинг автоматов - срок сдачи: 15.05.2011

onemorealexey@gmail.com

Лабораторная работа по рефакторингу автоматов

Целью лабораторной работы является обучение методам рефакторинга автоматов. Подробную информацию по этой теме можно прочитать в [диссертации Степанова О. Г.](#)

Для допуска к лабораторной работе необходимо успешно пройти [тест](#). Для этого надо ответить правильно не менее, чем на половину вопросов. Затем надо [скачать задание](#) и выполнить его в среде UniMod ([сайт проекта](#)).

Установка UniMod:

- Скачать и установить [Eclipse](#)
- Скачать архив с [плагином](#) (плагин содержит необходимый инструментарий для рефакторинга)
- Распаковать архив в папку eclipse (там где находятся eclipse.exe, plugins, features и т.д.)

Отчет по лабораторной работе должен содержать решение задания (в виде архива проекта Eclipse) и последовательность действий (в виде текстового файла), которые надо применить к исходному автомату, чтобы получить решение задания. Например:

- Автомат A1, слияние состояний s1 s4 s9
- Автомат A1, удаление группы состояний s8
- Автомат A2, замена вызова A3.run на A1.run

Отправить отчет можно [здесь](#).

Литература по рефакторингу:

- Фаулер М. Рефакторинг: улучшение существующего кода. СПб.: Символ-Плюс, 2004. – 432 с.
- Поликарпова Н.И., Шалыто А. А. Автоматное программирование. СПб.: Питер, 2010. – 176 с.
- Степанов О.Г. Методы реализации автоматных объектно-ориентированных программ. Кафедра «Технологии программирования». 2009. [Источник](#)
- UniMod. [Источник](#)
- Кафедра «Технологии программирования»: UniMod-проекты. [Источник](#)
- Бек К. Экстремальное программирование: разработка через тестирование. СПб.: Питер, 2003. – 224 с.

Остановить лабораторную

Рис. 8. Страница с описанием лабораторной работы для администратора

Виртуальная лаборатория

Выбрать лабораторную Описание Тест Лабораторная Выйти

Рефакторинг автоматов – срок сдачи: 15.05.2011 onemorealexey@gmail.com

Для верной работы лаборатории необходимо загрузить список студентов, описание и срок сдачи.
[Формат](#) загружаемых файлов

Список студентов
 Файл не выбран

Описание лабораторной
 Файл не выбран

Срок сдачи

Рис. 9. Страница с редактированием данных лабораторной работы

На странице «Описание» (если работа остановлена) администратор может загрузить список студентов, описание лабораторной работы, установить время сдачи задания. Эти файлы должны быть в кодировке *UTF-8* и соблюдать необходимый формат. Файл со списком студентов должен быть следующего формата:

```
Имя студента 1<Tab>почта студента 1
Имя студента 2<Tab>почта студента 2
Имя студента 3<Tab>почта студента 3
...
```

Например:

```
Тестовый Студент test.student@gmail.com
Шестаков Алексей onemorealexey@gmail.com
```

Необходимо обратить внимание на то, что при загрузке нового списка студентов обнуляются результаты тестирования.

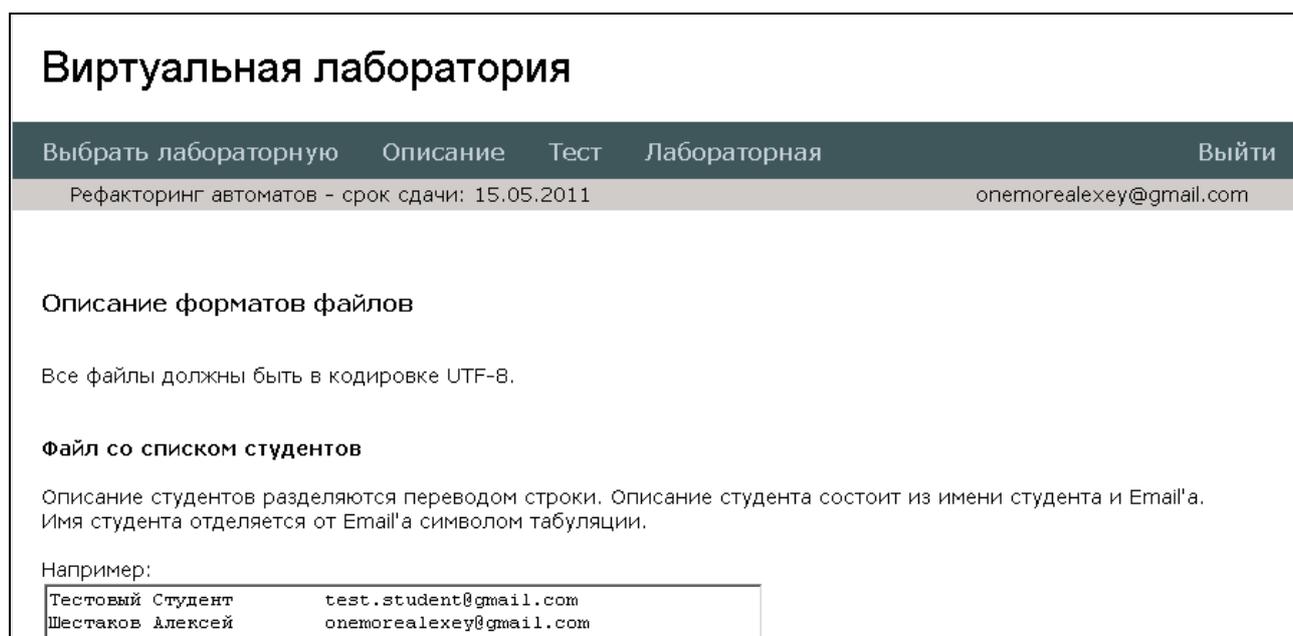
Файл с описанием лаборатории должен быть корректным файлом в формате *HTML*, например:

```
<h3>Лабораторная работа по рефакторингу  
автоматов</h3>
```

```
<p>Целью лабораторной работы является обучение  
методам рефакторинга автоматов. Подробную информацию по  
этой теме можно прочитать в <a href =  
"http://is.ifmo.ru/disser/stepanov_disser.pdf">  
диссертации Степанова О. Г.</a></p>
```

...

Полное описание форматов с примерами доступно по ссылке «Формат» (рис. 10).



The screenshot shows a web interface for a virtual laboratory. At the top, there is a navigation bar with links: 'Выбрать лабораторную', 'Описание', 'Тест', 'Лабораторная', and 'Выйти'. Below the navigation bar, the current page is identified as 'Рефакторинг автоматов - срок сдачи: 15.05.2011' and the user is logged in as 'onemorealexey@gmail.com'. The main content area is titled 'Описание форматов файлов' and contains the following text: 'Все файлы должны быть в кодировке UTF-8.' Below this, there is a section titled 'Файл со списком студентов' with the instruction: 'Описание студентов разделяются переводом строки. Описание студента состоит из имени студента и Email'a. Имя студента отделяется от Email'a символом табуляции.' An example is provided in a table:

Тестовый Студент	test.student@gmail.com
Шестаков Алексей	onemorealexey@gmail.com

Рис. 10. Страница с описанием форматов файлов

На странице «Тест» (рис. 11) администратор может изменить существующий тест и загрузить картинки, которые могут быть использованы в тесте.

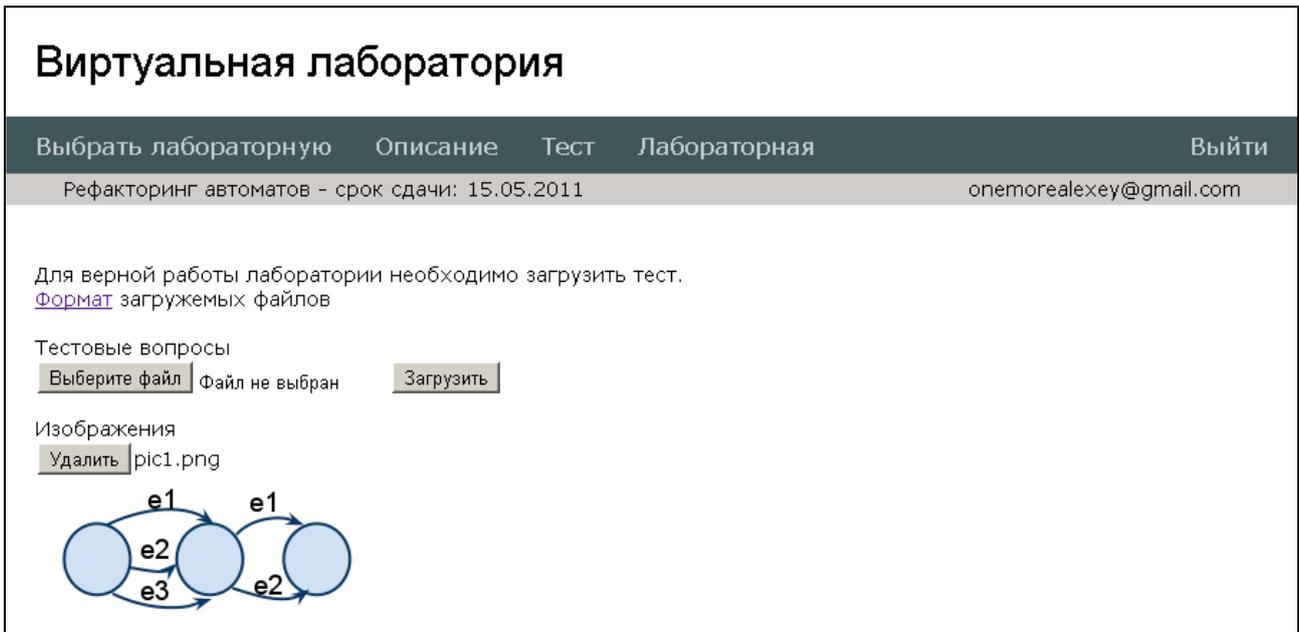


Рис. 11. Страница «Тест»

Файл с тестовыми вопросами должен быть в кодировке *UTF-8* и быть в формате, описанном на странице «Формат» (рис. 10):

Первый вопрос

@

+Правильный ответ

-Неправильный ответ

@

Второй вопрос

@

+Правильный ответ

+Правильный ответ

-Неправильный ответ

Например:

Как называется данный рефакторинг?

@

+Слияние состояний

- Группировка состояний
- Выделение автомата
- Перемещение действий в переходы

В тексте вопроса можно использовать форматирование с помощью *HTML*, в том числе и добавление картинок. Как показано на приведенном выше примере добавление картинки осуществляется с помощью тега ``. Если требуется добавить картинку, которая уже загружена на сайт, то вместо ссылки необходимо использовать имя файла.

Если перейти на страницу «Тест» в момент, когда лаборатории запущена, то администратор получает возможность просмотреть вопросы теста, правильные ответы отмечены галочками (рис. 12).

Выбрать лабораторную	Описание	Тест	Лабораторная	Выйти
Рефакторинг автоматов - срок сдачи: 15.05.2011		onemorealexey@gmail.com		
Тестовые вопросы:				
<hr/>				
Автоматное программирование это:				
<input type="checkbox"/> автоматическое создание программного обеспечения				
<input type="checkbox"/> программирование автоматов, роботов и т.д.				
<input checked="" type="checkbox"/> программирование с использованием модели конечных автоматов				

Рис. 12. Страница «Тест» при запущенной лабораторной работе

На странице «Лабораторная» (рис. 13) администратор может добавить задание студенту, отправить студенту письмо или получить отчет (в том случае, если студент выполнил задание и отправил отчет). Также выводится информация о том, как студент прошел тестирование.

Виртуальная лаборатория				
Выбрать лабораторную	Описание	Тест	Лабораторная	Выйти
Рефакторинг автоматов - срок сдачи: 15.05.2011			onemorealexey@gmail.com	
Студент	Информация	Тест	Задания	
Алексеев Сергей	подробнее	0 / 0	Задание	
Антон Сергеев	подробнее	0 / 0	Задание	
Бисенов Ерлан	подробнее	1 / 65	Задание	
Валентин Горбунов	подробнее	1 / 75	Задание	

Рис. 13. Страница «Лаборатория» со списком студентов

По ссылке «подробнее» администратор может узнать дополнительную информацию о студенте (рис. 14). На этой странице можно узнать электронную почту студента, написать ему сообщение от имени виртуальной лаборатории и посмотреть активность студента (информацию о том, когда студент скачал задание, прошел тест, сдал отчет).

Виртуальная лаборатория

Выбрать лабораторную | Описание | Тест | Лабораторная | Выйти

Рефакторинг автоматов - срок сдачи: 15.05.2011 onemorealexey@gmail.com

Заикин Александр
Email: **someEmail@gmail.com**

Результат теста: 65%, попыток осталось: 2

Задание [загружено](#)
Отчет [загружен](#)

Написать сообщение

Активность студента:

- 09:05:02 15.04.2011 - Задание скачано
- 19:42:40 28.04.2011 - Отчет загружен
- 18:00:22 02.05.2011 - Задание скачано
- 18:00:41 02.05.2011 - Отчет загружен

Рис. 14. Страница с подробной информацией о студенте

2.2. Лабораторная работа по рефакторингу автоматных программ

Одной из задач курсовой работы является создание лабораторной работы по рефакторингу автоматных программ. Для ее реализации требовалось создать новую лабораторную работу на сайте виртуальной лаборатории и загрузить файлы с описанием работы, тестом, списком студентов, заданиями лабораторной работы. Эти файлы можно найти в приложении. Также необходимо было создать среду, которая бы предоставляла возможность рефакторинга автоматов. *UniMod* [8] – проект с открытым исходным кодом, предназначенный для создания автоматных программ. Он позволяет

редактировать автоматы и проверять их корректность. Являясь плагином для *Eclipse* [7], этот проект предоставляет большие возможности для написания автоматных программ на *Java*.

Дополнение к *UniMod* добавляет возможность рефакторинга автоматов. Это позволяет автоматизировать работу пользователя и помогает избежать ошибок.

2.2.1. Реализация дополнения к *UniMod*

Для реализации дополнения был изменен пакет *UniMod* `unimod-modeller` (`eclipse/plugins/com.evelopers.unimod_1.3.39.1`). В него был добавлен пакет `refactor`, который содержит требуемую функциональность. Каждый рефакторинг состоит из списка простых команд, каждая из которых является обратимой, таким образом, весь рефакторинг тоже обратим. Рефакторинги находятся в пакете `refactor.actions`, составляющие их команды находятся в пакете `refactor.command`.

Для добавления новых возможностей необходимо создать класс, реализующий новое действие. Этот класс должен быть наследником класса `org.eclipse.jface.action.Action`. В случае если требуется работа с выделенными объектами, необходимо наследоваться от класса `org.eclipse.jface.action.SelectionAction`. Выделенные объекты можно получить с помощью метода `getSelectedObjects()`.

Для того чтобы действие было доступно, только если выполнено некоторое условие, надо переопределить метод `calculateEnabled()`. Также требуется переопределить метод `run()`. Все действия должны быть выполнены с помощью «команд». Класс команды должен наследоваться от класса `org.eclipse.gef.commands.Command`. В нем необходимо переопределить методы `undo()`, `redo()`, `execute()`, то есть каждая команда должна быть корректно обратимой. Несколько команд можно

объединить в сложную команду с помощью экземпляра класса `org.eclipse.gef.commands.CompoundCommand`. Таким образом, в методе `run()` требуется сформировать список команд, объединить их в `CompoundCommand` и передать на исполнение в редактор, для чего требуется вызвать метод `getUniModEditor()`.

Для примера более подробно рассмотрен код рефакторинга «Перенос действий из состояния в переходы».

```
package com.refactor.actions;  
import ...;
```

Для доступа к выделенным объектам класс наследуется от класса `SelectionAction`.

```
class MoveActionsFromStateToTransitions extends SelectionAction {
```

Идентификаторы действия.

```
ID = MoveActionsFromStateToTransitions.class.getName() + "_id";  
LABEL = "Export actions";  
TEXT = "Export actions";  
TOOL_TIP = "Export actions from selected states to incoming  
transitions";
```

Проверка возможности выполнения данного рефакторинга, а именно того, что текущий редактор является редактором автомата, есть выделенные объекты, эти объекты являются состояниями, пригодными для рефакторинга.

```
boolean calculateEnabled() {  
    return (getActiveEditor() instanceof StatechartPage)  
        && (getSelectedObjects().size() != 0)  
        && stateHasIncomingTransitionAndActions();  
}
```

Проверка того, что выделенные объекты являются обычными состояниями, у которых есть действия на входе и входящие переходы.

```
boolean stateHasIncomingTransitionAndActions() {
    for (Object o : getSelectedObjects()) {
        if (o instanceof StateEditPart) {
            StateEditPart ep = (StateEditPart) o;
            GState state = (GState) ep.getModel();
            if (!state.getType().equals(StateType.NORMAL)) {
                return false;
            }
            if (state.getIncomingTransitions().size() == 0 ||
                state.getOnEnterActions().size() == 0) {
                return false;
            }
        } else {
            return false;
        }
    }
    return true;
}
```

Создание списка команд, выполняющих рефакторинг и передача этого списка в редактор автомата.

```
void run() {
    ...
    getUniModEditor().execute(moveAllActions());
    ...
}
```

Создание команд для рефакторинга каждого выделенного состояния и объединение этих команд.

```
Command moveAllActions() {
    ...
```

```

CompoundCommand c = new CompoundCommand();
for (Object o : getSelectedObjects()) {
    StateEditPart ep = (StateEditPart) o;
    c.add(moveActions(ep));
}
return c;
}

```

Создание команды для рефакторинга конкретного состояния.

```

Command moveActions(StateEditPart ep) {
    CompoundCommand c = new CompoundCommand();

```

Выделенное состояние.

```

GState state = (GState) ep.getModel();

```

Действие, которое необходимо переместить.

```

Action toMove = (Action) state.getOnEnterActions().get(0);

```

Для каждого входящего перехода в список команд добавляется команда добавления нового действия.

```

for (Object o : state.getIncomingTransitions()) {
    GTransition tr = (GTransition) o;
    c.add(ref.addOutputActionToTransition(tr, toMove));
}

```

В список команд добавляется команда удаления действия из состояния, список команд завершен.

```

c.add(ref.removeOnEnterAction(state, toMove));
return c;
}

```

В качестве примера команды приведена реализация команды добавления состояния. Реализующий класс наследуется от класса `LoggedCommand`, который в свою очередь наследуется от класса `org.eclipse.gef.commands.Command`. В данном случае это было сделано для автоматического логирования выполнения команд.

```
package com.refactor.command;
import ...;
class AddStateCommand extends LoggedCommand {
```

В конструктор команды передается активный редактор автомата, имя автомата, в который необходимо добавить состояние, и параметры состояния.

```
AddStateCommand(UniModEditor editor, String m, String
stateName, StateType type, String superState) {
    ...
}
```

Выполнение команды: поиск необходимого автомата, создание требуемого состояния.

```
void exec() {
    StateMachine m = editor.getGModel().
getStateMachine(mName);
    State state = m.createState(name, type);
    state.setSuperstate(m.findState(superState));
}
```

Отмена команды: удаление состояния из автомата.

```
void und() {
    StateMachine m = editor.getGModel().
getStateMachine(mName);
    State state = m.findState(name);
    state.getSuperstate().removeSubstate(state);
```

```
}  
}
```

Для того чтобы добавить новое действие в главное или контекстное меню необходимо отредактировать файлы пакета `com.evelopers.unimod.plugin.eclipse.ui.base`:

- в классе `AbstractMultiPageEditor` модифицировать метод `createActions` (например, `addEditPartAction(new StateGrouping(this));`);
- в классе `DefaultActionBarContributor` модифицировать метод `buildActions` (например, `addRetargetAction(new RetargetAction(StateGrouping.ID, StateGrouping.LABEL));`);
- в классе `DefaultActionBarContributor` модифицировать метод `contributeToMenu` (для добавления пункта или подменю в главное меню);
- в классе `DefaultContextMenuProvider` модифицировать метод `buildContextMenu` (для добавления пункта или подменю в контекстное меню).

2.2.2. Интерфейс дополнения к *UniMod*

Разработанное дополнение к плагину *UniMod* добавляет в редактор необходимую функциональность для проведения рефакторинга автомата. В связи с увеличенной функциональностью интерфейс пользователя был расширен.

При открытии файла с автоматом в главном меню появляется подменю «Machine refactor» (рис. 15), в котором можно выбрать один из семи рефакторингов. Активированы только те рефакторинги, которые применимы к выделенным состояниям. Также, если к выделенным состояниям применим

хотя бы один из рефакторингов, то в контекстном меню появляется подменю «Machine refactor» (рис. 16). В нем можно выбрать один из возможных рефакторингов.

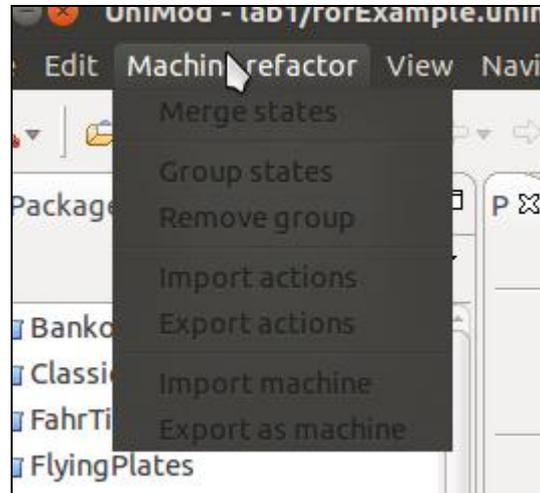


Рис. 15. Главное меню

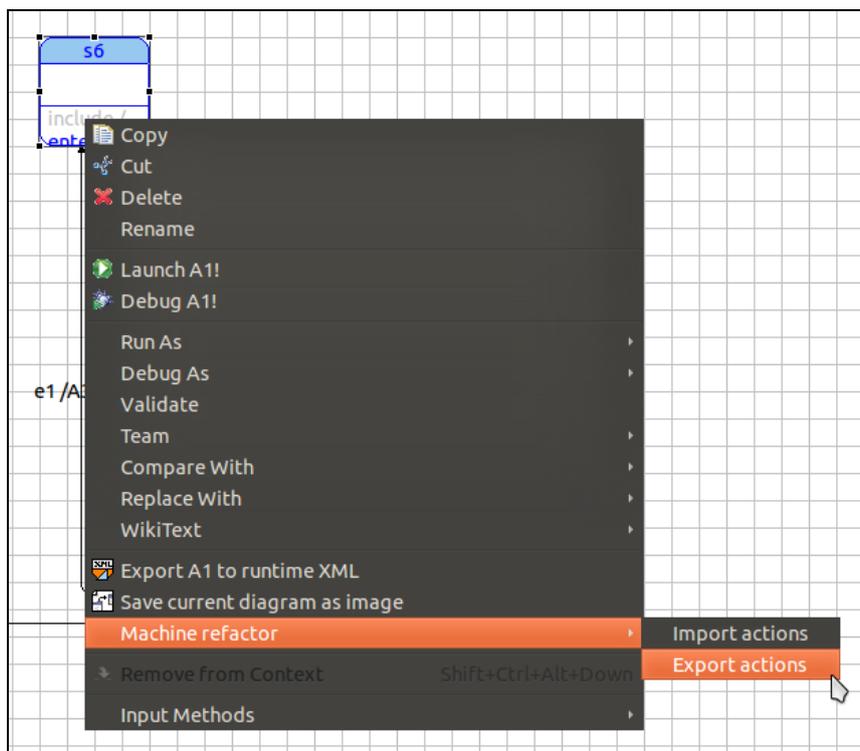


Рис. 16. Контекстное меню

Таким образом, пользователю предлагается удобный интерфейс для проведения рефакторинга автоматных программ.

2.2.3. Пример рефакторинга

В качестве примера работы дополнения приведен рефакторинг «Слияние состояний». Его можно провести, если выделенные состояния обладают одинаковыми исходящими переходами и содержат одинаковый набор действий на входе. Этот и другие рефакторинги, а также методика их выполнения подробно описаны в работе [2].

В исходном автомате присутствуют состояния s_3 , s_4 , s_5 (рис. 17). Они подходят для рефакторинга «Слияние состояний». Для проведения рефакторинга требуется выделить их и в контекстном меню выбрать пункт «Merge states» (рис. 18). Рефакторинг завершен (рис. 19). Три состояния объединены в одно новое s_7 .

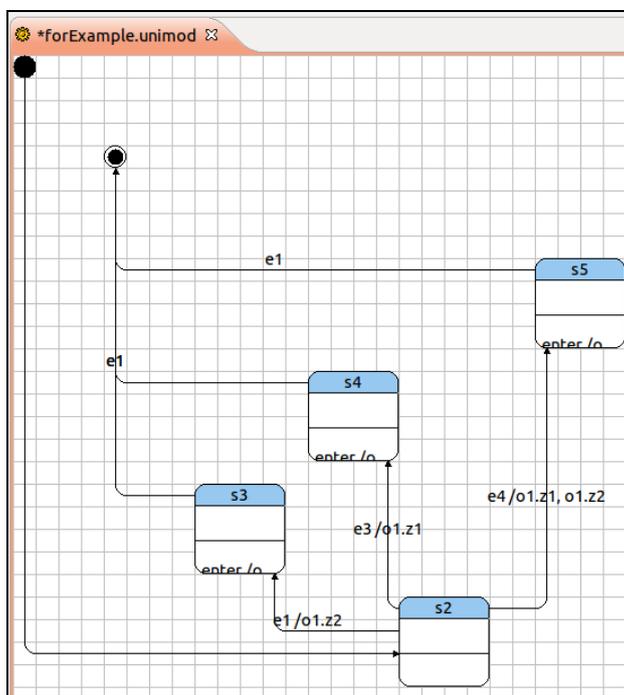


Рис. 17. До рефакторинга «Слияние состояний»

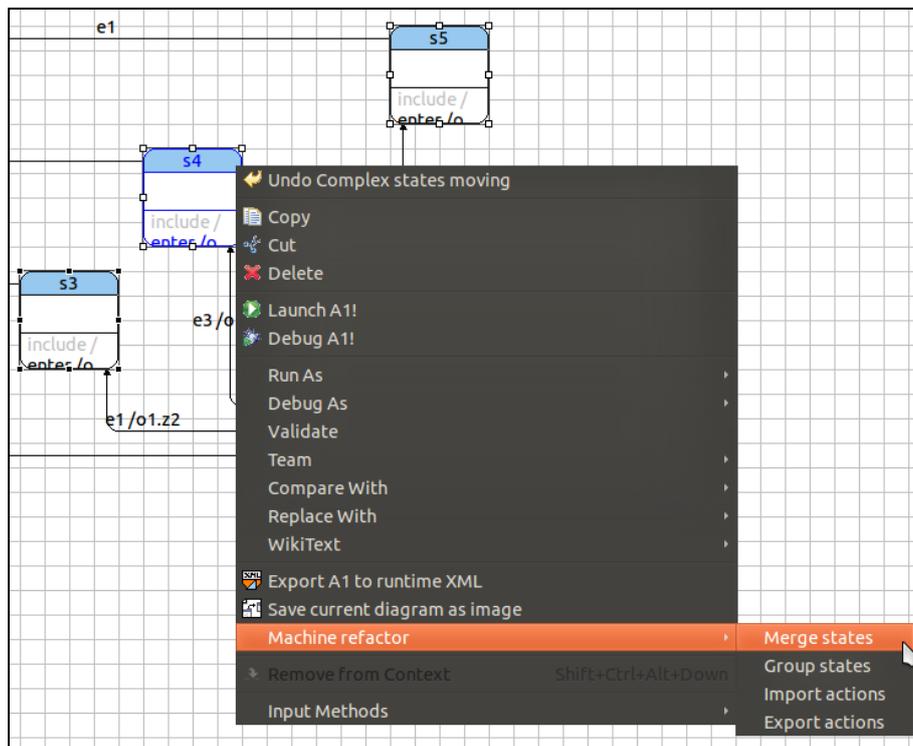


Рис. 18. Выбор пункта «Merge states»

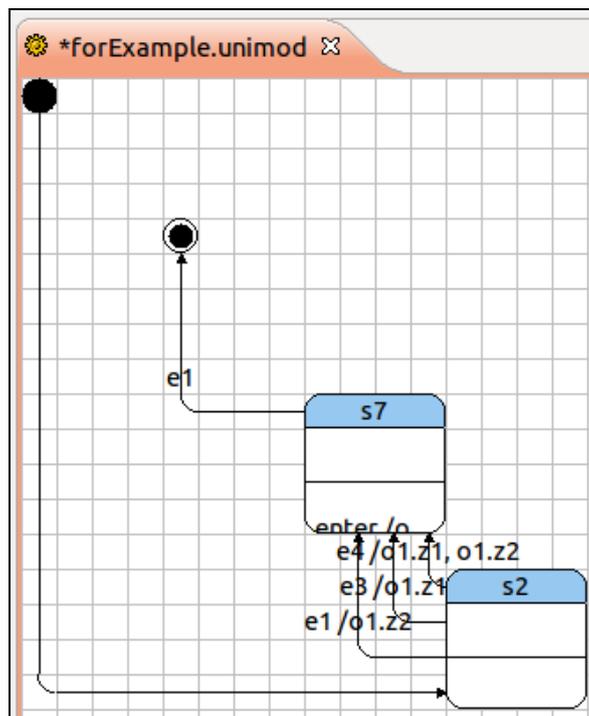


Рис. 19. После проведения рефакторинга «Слияние состояний»

2.2.4. Пример отчета

В случае виртуальной лаборатории по рефакторингу автоматных программ, отчет должен содержать результат рефакторинга (в виде проекта *Eclipse*) и последовательность действий, которые необходимо совершить, чтобы прийти к этому результату. Предполагается, что студенты должны:

- скачать задание;
- открыть проект (входящий в задание) в среде *Eclipse*;
- провести рефакторинг предложенного автомата;
- записать произведенные в процессе рефакторинга действия;
- отправить полученный проект в виде архива, приложив к нему файл со списком выполненных действий.

Пример задания:

Действие `o1.inMarkedState` должно вызываться при входе во все помеченные состояния (имя состояния начинается с `M`). Проведите рефакторинг автомата таким образом, чтобы для выполнения задания потребовалось модифицировать лишь одно состояние.

В данном задании предполагается, что возможно провести рефакторинг автомата таким образом, что после него останется одно состояние, логически эквивалентное всем помеченным состояниям.

Пример списка рефакторингов:

- 1) Автомат A2, remove group s3
- 2) Автомат A2, import actions in Ms3
- 3) Автомат A2, merge states Ms2 and Ms3
- 4) Автомат A2, rename merged states to Marker
- 5) Автомат A2, import actions in s6
- 6) Автомат A3, remove group s3
- 7) Автомат A3, import actions in Ms2
- 8) Автомат A3, import actions in Ms3

- 9) Автомат A3, merge states Ms1, Ms2 and Ms3
- 10) Автомат A3, rename merged states to Marker
- 11) Автомат A3, import actions in s7

12) Автомат A3, merge states s6 and s7

13) Так как теперь автоматы A2 и A3 эквивалентны, то делаю замену вызова A3.run на A2.run. Для выполнения задания остаётся модифицировать только одно состояние, это состояние Marked в автомате A2.

2.3. Лабораторная работа по контрактному программированию

Для реализации лабораторной работы по контрактному программированию потребовалось создать новую лабораторную работу на сайте виртуальной лаборатории и загрузить файлы с описанием работы, тестом, списком студентов, заданиями лабораторной работы. Эти файлы можно найти в приложении.

Среда для разработки приложений с использованием указанного подхода уже реализована в работе [4]. Для создания нового проекта необходимо правильно настроить среду *MPS*. Установка и настройка описаны на странице с описанием лаборатории.

1. Установите *MPS 1.0* (источник <http://download-in.jetbrains.com/mps/MPS-1.0.1.exe>).
2. Запустите среду *MPS*, создайте новый проект, не создавайте новый язык.
3. Скачайте архив с языком *State Machine* (источник <http://confluence.jetbrains.com/download/attachments/13697578/stateMachine-4120.zip?version=1&modificationDate=1250266131000>).
4. Распакуйте архив в любую папку (например, **smlib**).
5. Запустите *MPS*, перейдите в **Settings->MPS Library Manager**.

6. Добавьте новую библиотеку (кнопка **add**), введите путь до папки с jar-файлами (например, **smlib**).
7. Установите *NuSMV 2.4.3*. Для работы плагина требуется, чтобы *NuSMV* был установлен в **C:\Program Files\NuSMV\2.4.3**. (источник <http://nusmv.fbk.eu/distrib/NuSMV-2.4.3-i586-pc-mingw32msvc.exe>).
8. Скачайте архив с языком темпоральных спецификаций.
9. Распакуйте архив в любую папку (например, **temporalLogic**).
10. Запустите *MPS*, перейдите в **Settings->Project Settings**.
11. Добавьте в *Languages* *MPL*-файлы из папки **temporalLogic\languages**.
12. Добавьте в *Solutions* *MSD*-файлы из папки **temporalLogic\solutions**.
13. Возможно, для продолжения работы потребуется перезапустить *MPS*.
14. В `jetbrains.mps.temporalLogicLanguage.sandbox` находятся примеры программ, подробнее о работе в *MPS* можно прочитать в работе [2].

После установки и настройки студентам предлагается пройти тест, содержащий вопросы по теме «Проектирование программного обеспечения», и реализовать какой-нибудь объект. Для этого объекта требуется составить спецификацию на языке *temporalLogic* [4]. Язык темпоральных спецификаций является расширением языка *stateMachine* [4], таким образом, студенту необходимо написать класс *Java* реализующий объект управления, написать автомат, реализующий логику программы, и расширить этот автомат спецификацией.

Заключение

Результатом настоящей курсовой работы является платформа для проведения виртуальных лабораторных работ. Виртуальная лаборатория реализована в виде сайта и обладает требуемой функциональностью. На этой платформе были реализованы следующие лабораторные работы:

- лабораторная работа по рефакторингу автоматных программ;
- лабораторная работа по контрактному программированию.

Для лабораторной работы по рефакторингу автоматных программ было реализовано дополнение, позволяющее использовать *UniMod* [8] для рефакторинга автоматов.

Источники

1. Фаулер М. Рефакторинг: улучшение существующего кода. СПб.: Символ-Плюс, 2004. – 432 с.
2. Степанов О. Г. Методы реализации автоматных объектно-ориентированных программ. Диссертация на соискание ученой степени кандидата технических наук. СПб.: 2009. http://is.ifmo.ru/disser/stepanov_disser.pdf
3. Кулямин В. В. Методы верификации программного обеспечения — Институт системного программирования РАН. <http://www.viva64.com/go.php?url=282>
4. Борисенко А. А. Интеграция верификации в интерактивный процесс разработки программ с явным выделением состояний. Бакалаврская работа. СПб.: 2010. http://is.ifmo.ru/papers/_borisenkobak.pdf
5. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. СПб.: Питер, 2010. – 176 с.
6. *Google App Engine*. <http://code.google.com/intl/ru-RU/appengine/>
7. Среда разработки *Eclipse*. <http://www.eclipse.org/>
8. Плагин *UniMod*. <http://unimod.sourceforge.net/>
9. Среда разработки *MPS* (*Meta Programming System*). <http://www.jetbrains.com/mps/>