МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”
им. В.И. Ульянова (Ленина)” (СПбГЭТУ)**

Направление *Информационные системы*

Кафедра АСОИУ

К защите допустить:
Руководитель направления **Советов. Б. Я.**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: «Информационная система управления вариантами использования ПО».

Студент	*Подольская А. Н.*	/ /
Руководитель	*Выговский Л. С.*	/ /
Зав. кафедрой	*Советов Б. Я.*	/ /

Санкт-Петербург

2012 г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”
им. В.И. Ульянова (Ленина)” (СПбГЭТУ)**

УТВЕРЖДАЮ

Факультет КТИ

Руководитель направления

Кафедра АСОИУ

Советов Б. Я. / /

«_____» _____ 201__г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра

Студент **Подольская А. Н.** Группа № **8372**

1. Тема работы **«Информационная система управления вариантами использования ПО»**

(утверждена приказом № _____ от _____)

2. Исходные данные (технические требования) Описание работы с вариантами использования.
Общая постановка задачи.

3. Содержание работы Разработка модели предметной области вариантов использования программного обеспечения.
Разработка проектного решения для информационной системы управления вариантами использования программного обеспечения.

4. Перечень отчетных материалов Пояснительная записка к ВКР. Аннотация к ВКР. Презентация.

Дата выдачи задания
«__» _____2012г.

Дата представления работы к защите
«15» июня 2012 г.

Заведующий кафедрой Советов Б. Я. / /

Руководитель Выговский Л.С. / /

Задание к исполнению принял Подольская А.Н. / /

«_____» _____ 2012г.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”
им. В.И. Ульянова (Ленина)” (СПбГЭТУ)**

УТВЕРЖДАЮ

Факультет КТИ — Заведующий кафедрой

Советов Б. Я.

Кафедра АСОИУ — / /

«_____» _____ 201__г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы

Студент **Подольская А.Н.** Группа № **8372**

Тема ВКР **«Информационная система управления вариантами использования ПО»**

Сроки выполнения ВКР **5 недель**

№ п/п	Наименование работ	Срок выполнения	Отметка о выполнении
1.	Выбор темы ВКР.	06.05.2012	Выполнено
2.	Разработка плана написания ВКР.	07.05.2012	Выполнено
3.	Установка необходимых средств.	06.05.2012 – 09.05.2012	Выполнено
4.	Изучение необходимой информации.	06.05.2012 – 09.05.2012	Выполнено
5.	Написание введения.	10.05.2012 – 14.05.2012	Выполнено
6.	Разработка модели предметной области вариантов использования ПО.	15.05.2012 – 19.05.2012	Выполнено
7.	Написание 1 главы.	20.05.2012 – 24.05.2012	Выполнено
8.	Разработка проектного решения для информационной системы управления вариантами использования ПО.	25.05.2012 – 29.05.2012	Выполнено
9.	Написание 2 главы.	30.05.2012 – 03.06.2012	Выполнено
10.	Написание заключения.	04.06.2012 – 07.06.2012	Выполнено
11	Полное оформление пояснительной записки.	08.06.2012 – 12.06.2012	Выполнено
12.	Создание презентации.	08.06.2012 – 13.06.2012	Выполнено
13.	Подготовка к выступлению.	10.06.2012 – 14.06.2012	Выполнено

Руководитель Выговский Л.С. / /

Задание к исполнению принял Подольская А.Н. / /

«_____» _____ 2012г.

Аннотация

Пояснительная записка 46 с., 16 рис., 14 источников. В данной дипломной работе разработано проектное решение, реализующее web-приложение для работы с вариантами использования – создания, редактирования и просмотра. В дипломном проекте разработаны и представлены модель предметной области и модель самого приложения; наглядно представлена будущая работа приложения; а также объяснен выбор всех используемых средств и технологий. Эта работа будет использоваться в дальнейшем для разработанного программного продукта Use Case Organizer.

Abstract

Explanatory note 46 p., 16 pic., 14 sources. In this diploma has been created design decision realizing the web-application for work with use case – creating, editing and viewing. In the diploma project a domain model and the model of the application are developed and presented; visually future work of the application is presented; as well as explained the choice of one used by tools and technologies. This work will be used in future for the developed software product Use Case Organizer.

Ключевые слова

- Программное обеспечение – совокупность программ системы обработки данных и программных документов, необходимых для эксплуатации этих программ.
- Web-приложение – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – web-сервер.
- Управление требованиями – процесс, включающий идентификацию, выявление, документирование, анализ, отслеживание, приоритезацию требований, достижение соглашения по требованиям и затем управление изменениями и уведомление соответствующих заинтересованных лиц.
- Вариант использования – совокупность сценариев, описывающих поведение системы при ее ответах на запрос от одного из участников и заканчивающихся успешно.
- Методология – совокупность методов, применяемых в некоей науке, области знания.
- Водопадная модель – модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.
- Agile – серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки и динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самооргани-

зующихся рабочих групп, состоящих из специалистов различного профиля.

- Java – объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems.
- ООП – парадигма программирования, в которой основными концепциями являются понятия объектов и классов.
- Инкапсуляция – свойство языка программирования, позволяющее объединить и защитить данные и код в объекте и скрыть реализацию объекта от пользователя.
- Принцип открытости-закрытости – принцип языка программирования, при котором программные сущности должны быть открыты для расширения, но закрыты для изменения.
- SQL – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.
- Maven – фреймворк для автоматизации сборки проектов, специфицированных на XML-языке POM.
- Mybatis – фреймворк для маппинга объектов СУБД к Java объектам.
- Wicket – фреймворк с открытым исходным кодом и свободной лицензией для создания web-приложений, который основан на использовании компонентов.

Содержание

Введение

В современном информационном мире программное обеспечение играет очень важную роль. Программное обеспечение подразделяется на системное, прикладное и инструментальное. Без программного обеспечения невозможна работа на компьютере; в первую очередь это относится к системному программному обеспечению. Прикладное и инструментальное программное обеспечение так же необходимы для всех пользователей: различные редакторы, мультимедиа, корпоративные информационные системы, программные средства защиты, системы управления базами данных и средства разработки самого программного обеспечения. Разрабатывается новое программное обеспечение, которое более эффективно, качественно и продуктивно решает особые цели пользователей.

Проблемы, возникающие при разработке нового программного обеспечения [1]:

1. Недостаток прозрачности: в любой момент времени сложно определить, каков процент завершения проекта и какого его состояние, данная проблема возникает из-за отсутствия достаточного финансирования проекта – разработчики сокращают этап проектирования, а в реезультате в дальнейшем им приходится решать эту проблему;

2. недостаток контроля: без точной оценки процесса разработки изменяются (увеличиваются) сроки выполнения проека, а также превышаются установленные бюджеты, данная проблема чаще всего возникает тогда, когда после завершения больше половины проекта вкладываются дополнительные финансковые средства и разработка продолжается без оценки степени завершенности проекта;

3. неконтролируемые изменения: у потребителей постоянно возникают новые идеи относительно разрабатываемого программного обеспечения, что приводит к тому, что необходимо постоянно корректирования требования к проекту; эта задача требует дополнительных средств, как человеческих и финансовых, так и программных;

4. недостаточная надежность: поиск и исправление ошибок в программе; число ошибок заранее неизвестно, поэтому неизвестно продолжительность отладки программы, из-за этого тяжело точно определить сроки выполнения проекта.

Все эти проблемы в целом приводят к проблемам качества, стоимости и надежности, для их решения требуется больше запланированного времени. В следствии этого на данный момент к поставленному сроку, не превышая бюджет, выполняется не более 20% [2] всех проектов по разработке программного обеспечения.

Для решения этих проблем необходимо использовать эффективную методологию, что позволит реализовать: управление проектом на основе актуальных данных; планирование действий сотрудников; документирование требований и отслеживание изменений; распределение обязанностей и ответственности всех участников; удобное и эффективное взаимодействие между службой технической поддержки, тестировщиками, разработчиками; составление отчетов по проекту. Модель жизненного цикла программного обеспечения – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Для решения проблем, описанных выше, было предложено применять водопадную модель или waterfall model. Эта модель предполагает строго последовательное и однократное выполнение фаз проекта с жестким предварительным планированием. Фазы

выполняются в порядке:

1. Определение требований;
2. проектирование;
3. реализация (кодирование);
4. интеграция;
5. тестирование и отладка;
6. инсталляция;
7. поддержка.

Модель подразумевает, что переход от одной фазы к последующей происходит только после полного и успешного завершения предыдущей фазы. К преимуществам можно отнести наличие полной и согласованной документации на каждом этапе, а также легкость определения сроков и затрат на проект. Из-за неточности требований или некорректной их интерпретации, а также большой вероятности изменений пожеланий клиента, на практике легкость определения сроков и затрат на проект выполняется редко. Приходится начинать всю разработку проекта заново, что сильно увеличивает сроки, а также бюджет проекта. К недостаткам данного метода часто относят недостаточную гибкость, а также объявление самоцелью формальное управление проектом в ущерб срокам, стоимости и качеству.

Многие разработчики используют гибкую методологию разработки (Agile software development)[3] - серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки и динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри рабочих групп,

состоящих из специалистов различного профиля. Большинство гибких методологий нацелены на минимизацию рисков, путем сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся две-три недели. Каждая итерация включает все или часть этапов водопадной модели. Главный упор в agile делается на работу с клиентом, на частое общение с ним, а также на необходимую корректировку требований. К недостаткам данной методологии относят отсутствие далеко идущих планов (заказчик может в любой момент выставлять новые требования, противоречащие тому, что уже сделано, что иногда приводит к значительным переделкам итерации), а также предпочтение решения задач простейшим и быстрейшим способом, что часто приводит к дефектам.

При выборе любой методологии для разработки программного обеспечения управление требованиями [4](процесс, включающий идентификацию, выявление, документирование, анализ, отслеживание, приоритеризацию требований, достижение соглашения по требованиям и затем управление изменениями и уведомление соответствующих заинтересованных лиц) является неотъемлемой частью разработки. Все требования необходимо отслеживать – документировать весь жизненный цикл требования. В частности, это может быть использовано для определения первоисточника требования, чтобы проследить интересы каждого пользователя и т.д. Многие разработчики прибегают к помощи вариантов использования (use cases) для описания поведения программных систем и требований к ним.

Вариант использования фиксирует соглашение между участниками системы о ее поведении. Вариант использования описывает поведение системы при ее ответах на запрос от одного участников, называемого основным действующим лицом, в различных условиях. Основное дей-

ствующее лицо инициирует взаимодействие с системой, чтобы добиться некоторой цели. Система отвечает, соблюдая интересы всех участников. Различные модели поведения, или сценарии, развертываются в зависимости от определенных запросов и условий, при которых делались эти запросы. Вариант использования собирает вместе эти сценарии. Варианты использования представлены большей частью в текстовой форме, так как они служат средством связи между лицами, часто не имеющими специальной подготовки. [5]

Вариант использования как форма описания стимулирует обсуждение проектируемой системы в группе разработчиков. Одна команда может с помощью варианта использования документировать действительные требования, другая – окончательный проект. Если варианты использования описывают требования к поведению части программного обеспечения, рассматриваемая система (SuD – system under discussion) – это компьютерная программа. Участники – пользователи программы, ее владельцы, провайдер и другие компьютерные программы. Основное действующее лицо – сидящий за компьютером пользователь.

Следует отметить, что варианты использования – это только часть требований. В общей схеме требований варианты использования занимают всего один раздел из шести (цель и область действия; используемые термины; варианты использования; используемая технология; другие требования; людские резервы, правовые, политические, организационные вопросы). Это требования к поведению. Варианты использования связаны со многими другими деталями требований. Они помогают связать информацию в различных частях требований, включая требования о профиле пользователя, правила и требования к форматам данных. Вне документа о требованиях варианты использования помогают систематизировать информацию о планировании проекта, такую, как даты выпус-

ка, команды разработчиков, приоритеты и состояние разработки. Тем самым частично решается проблема прозрачности. Кроме того, они служат для отслеживания определенных результатов деятельности команды разработчиков, в частности интерфейса пользователя и системных тестов. Хотя эти требования и не входят в описание вариантов использования, все они с ними связаны. Как следствие варианты использования представляются людям центральным узлом требований или даже центральным узлом процесса разработки системы.

Варианты использования популярны прежде всего потому, что связно рассказывают о поведении системы при ее использовании. Пользователи системы могут увидеть, что это будет за система. Они получают возможность на ранней стадии влиять на точную настройку. Варианты использования являются особенно ценными. когда именуются целями пользователя, которые будет реализовывать система, и собираются в список. Этот список объявляет, что будет делать система, раскрывая ее область применения и цели. Список – это каркас, на который нанизываются сложность, стоимость, сроки и состояние проекта, он накапливает разнообразную информацию в течение всего периода разработки. Также варианты использования важны тем, что в них определяются и записываются по возможности все факты, которые могли бы вызвать сбой в успешном сценарии, а также реакции системы на них.

На рынке программного обеспечения существуют продукты, реализующие работу с управлениями требований. Несколько из них:

1. CaliberRM [6]. Корпоративная система управления требованиями на этапе процесса создания программного обеспечения. Разработана в целях повышения качества создаваемых продуктов и предназначена для улучшения взаимодействия между участниками проек-

та, упрощения анализа влияний и процесса передачи информации в сфере управления изменениями исходных требований. Группы разработчиков, используя CaliberRM для определения и отслеживания требований, а также назначения приоритетов на протяжении всего жизненного цикла проекта, могут оперативно реагировать на постоянные изменения требований, не подвергая риску успешность проекта. К возможностям данного продукта относится: наличие централизованного репозитория с защищенным доступом к данным, это позволяет всем участникам разработки быть в курсе последних изменений; возможность интеграции с другими технологиями жизненного цикла приложений, что удобно, так как разработчики могут работать в различных приложениях, а затем интегрировать все данные о требованиях и т.д. В левой части пользовательского интерфейса отображается дерево требований, а в правой - вкладки, содержащие различную информацию по этим требованиям. У CaliberRM есть свои недостатки: этот продукт является платным, год эксплуатации стоит от 90 000 рублей – не все разработчики могут позволить себе такую цену; данный продукт требует установки, следовательно, пользователю необходимо периодически обновлять версию; это система охватывает весь процесс управления данными целиком, в то время, как более качественным был бы подход разбиения всего процесса по частям – разделам управления требования.

2. Use Case Tool [7]. Инструмент для написания вариантов использования. Вариант использования пишется в XML формате, затем Use Case Tool генерирует его в HTML формат, создавая ссылки между страницами. К достоинствам разработчики относят: создание

ссылок, которые упрощают и структурируют вид варианта использования; автоматическая нумерация шагов основного сценария и расширения; представление варианта использования по созданному шаблону Алистера Коберна; использование Maven; бесплатное, открытое приложение. Но по сути, данный способ работы не очень сильно отличается от способа работы в Microsoft Office Word. Данный продукт появился совсем недавно – последняя дата изменений информации на сайте – 12 апреля 2012 года, что показывает, что проблема создания эффективного инструмента по работе с вариантами использования является важной и ищутся способы её решения.

3. Microsoft Office Word. Текстовый редактор, стандартный для ОС Windows, часто используется для написания вариантов использования, так как является доступным, нетребующим изучения и финансовых затрат. В таком виде вариант использования выглядит как список с наименованием полей. Для взаимодействия между разработчиками файл должен либо постоянно передаваться между ними, либо хранится в доступном для всех месте. Данный инструмент является элементарным средством для записи, без каких-либо дополнительных функций.

4. Wiki Confluence [8]. Wiki – web-сайт, структуру и содержимое которого пользователи могут самостоятельно изменять с помощью инструментов, предоставляемых самим сайтом. Wiki Confluence – сайт, на котором все сотрудники фирмы могут выкладывать свои мысли, идеи, обмениваться файлами, сообщать обо всех важных событиях и так далее. В общем, этот продукт нельзя отнести к средствам, работающим с управлениями требований, так как он

предназначен больше для взаимодействия и поддержания контакта между всеми сотрудниками и решение проблемы управления требованиями не выносится на первый план.

Для эффективной работы с вариантами использования нужен продукт, который будет отвечать следующим требованиям:

1. web-приложение, несложное в освоении, доступное представителям малого и среднего бизнеса, не требующее никаких дополнительных средств;
2. возможность пользователя работать над несколькими проектами и возможность работы несколькими пользователями над одним проектом;
3. сохранение в проекте участников всех вариантов использования, а также области действия;
4. возможность просмотра истории изменения вносимых изменений;
5. безопасность работы с данными;
6. интеграция различных вариантов использования и их взаимодействие.

Небольшие фирмы, пишущие лишь несколько вариантов использования в течении длительного времени, чаще всего используют Microsoft Office Word, так как он всегда доступен и в нем можно описать вариант использования. Очень крупные компании используют такие продукты, как CaliberRM, которые включают в себя большое количество функции и интегрируются с другими программными продуктами. Остальным же фирмам – малого и среднего бизнеса – составляющим наибольший

процент среди пользователей, работающих с управлениями требования, необходим отдельный продукт, работающий только с вариантами требования и отвечающий приведенным выше требованиям. Написание грамотного варианта использования – один из самых главных шагов в управлении требованиями и на нем можно остановиться.

Проведенный анализ показал наличие противоречия между существующими средствами для управления вариантами использования и предъявленными к ним требованиями. Для разрешения выявленного противоречия была поставлена цель данного дипломного проекта: проектирование информационной системы управления вариантами использования программного обеспечения. Для достижения сформулированной цели были поставлены следующие задачи:

1. Разработка модели предметной области варианта использования программного обеспечения;

2. Разработка проектного решения для информационной системы управления вариантами использования программного обеспечения.

Глава 1

Модель предметной области

1.1. Описание предметной области

Вариант использования – совокупность сценариев, описывающих поведение системы при её ответах на запрос от одного из участников и заканчивающихся успешно; при этом система должна соблюдать интересы всех участников процесса.

Понятия, возникающие при работе с вариантами использования:

1. Действующее лицо (Actor) – кто-то (или что-то), обладающий поведением;

2. заинтересованное лицо (Stakeholder) – кто-то (или что-то), проявляющий интерес к поведению рассматриваемой системы, один из тех, кто заключает контракт;

3. основное действующее лицо (Primary actor) – участник, инициирующий взаимодействие с рассматриваемой системой для достижения некоторой цели;

4. вариант использования (Use case) – соглашение относительно поведения рассматриваемой системы;

5. область действия (Scope) – идентифицирует рассматриваемую систему;

6. уровень (Level) – обозначает за сколько сеансов можно добиться цели (один - цель пользователя, много - обобщенный уровень);

7. предусловия и гарантии (Preconditions and guarantees) – то, что должно быть истинным до и после реализации варианта использования;

8. основной сценарий (Main success scenario) – вариант, в котором не возникает никаких ошибок;

9. расширения (Extensios) – различные отклонения от основного сценария.

1.2. Модель предметной области

На рис. 1.1 приведена модель варианта использования.

Вариант использования выполняется на одном из уровней: обобщенный уровень, уровень цели пользователя и уровень подфункции. Обобщенный уровень включает в себя задачи, которые невозможно выполнить за один сеанс и которые выполняют цели нескольких участников. Задачи уровня цели пользователя решаются за один сеанс, выполняя цель основного действующего лица. Подфункции – это задачи, выполнение которых необходимо для реализации целей пользователя. Желательно избегать подфункций, объединять их между собой в задачи уровня цели пользователя.

Для каждого варианта использования существует своя область действия – определение границ проекта. Это может быть компьютерная программа, предприятие или отдельный отдел. Для верного определения области действия варианта использования необходимо определить, где находятся рассматриваемые объекты – вне или внутри области обсуждения. Действенным инструментом является создание таблицы Действующее лицо/Цель. В этой таблице содержатся только те услуги, которые

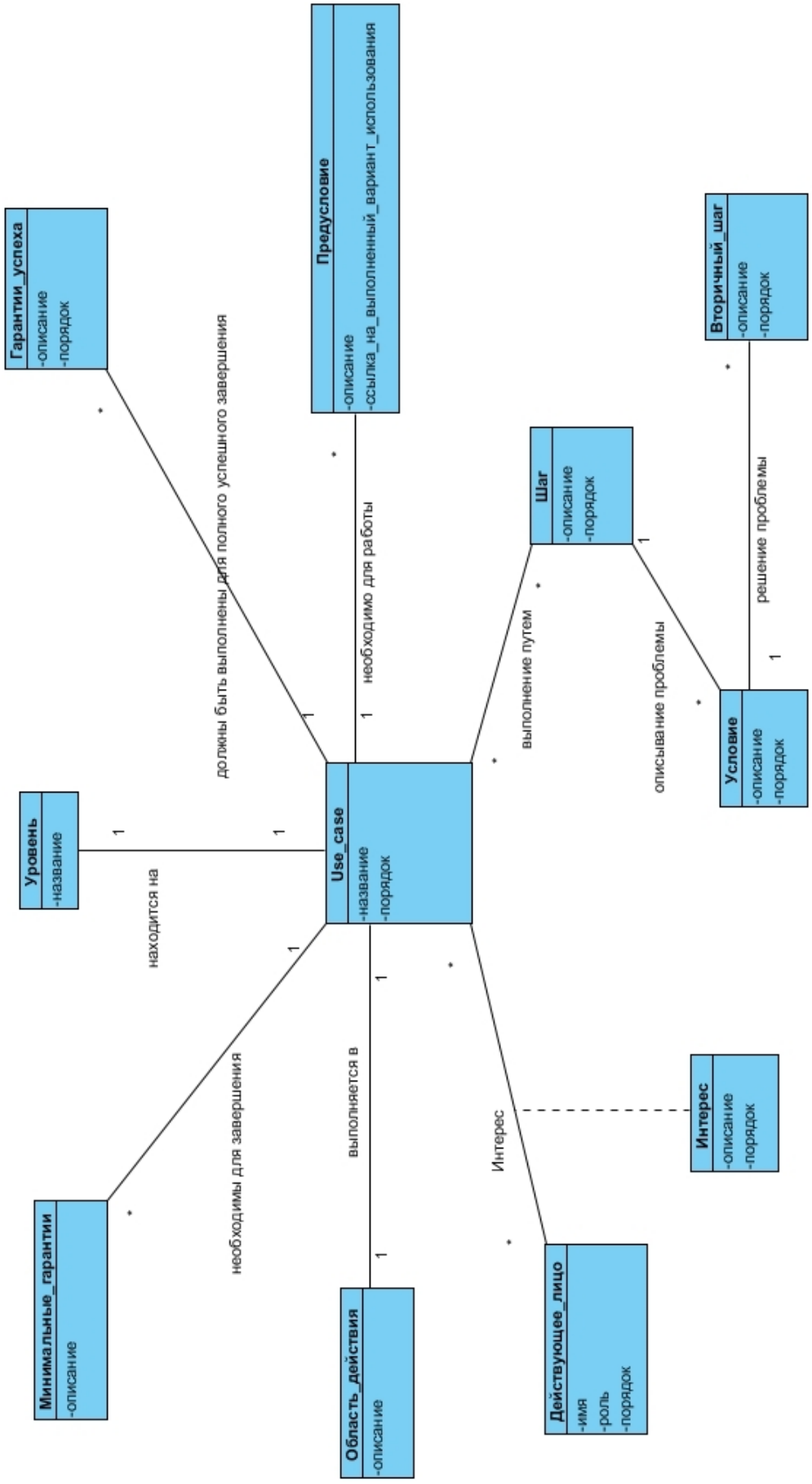

Рис. 1.1. Модель Use Case

реально будут поддерживаться рассматриваемой системой, данный список должен обновляться на протяжении всего процесса разработки, так как фокусирует внимание на планировании и содержании проекта.

Предусловие варианта использования объявляет, выполнение какого условия гарантирует система перед тем, как разрешить запуск этого варианта использования. Данное условие не проверяется в период выполнения. Обычно предусловие указывает на то, что был успешно выполнен предыдущий вариант использования. Так же оно может отсутствовать. Примером может служить предусловие "клиент зашёл в систему" которое показывает, что процесс "Вход в систему"прошел успешно.

Минимальные гарантии – наименьшие обещания системы участникам, если цель основного действующего лица не может быть достигнута. Минимальные гарантии необходимы, чтобы даже при отказе системы в выполнении основной цели, интересы участников хоть частично выполнялись и был виден результат работы системы.

Гарантии успеха устанавливают, что интересы участников удовлетворяются при успешном завершении варианта использования в конце основного сценария или в конце успешного альтернативного пути. Они пишутся в добавление к минимальным гарантиям, так как выполняются дополнительные условия. В первую очередь к этим дополнительным условиям относится цель, вынесенная в название варианта использования (цель основного действующего лица). Для верного определения гарантий успеха задается вопрос: "что сделало бы этого участника неудовлетворенным по окончании успешного выполнения варианта использования?"и записывается обратное утверждение.

В каждом варианте использования участвуют различные действующие лица, имеющие разные цели и выполняющие разные функции. На рис. 1.2 показаны отношения классов, связанных в данном проекте с дей-

ствующими лицами.

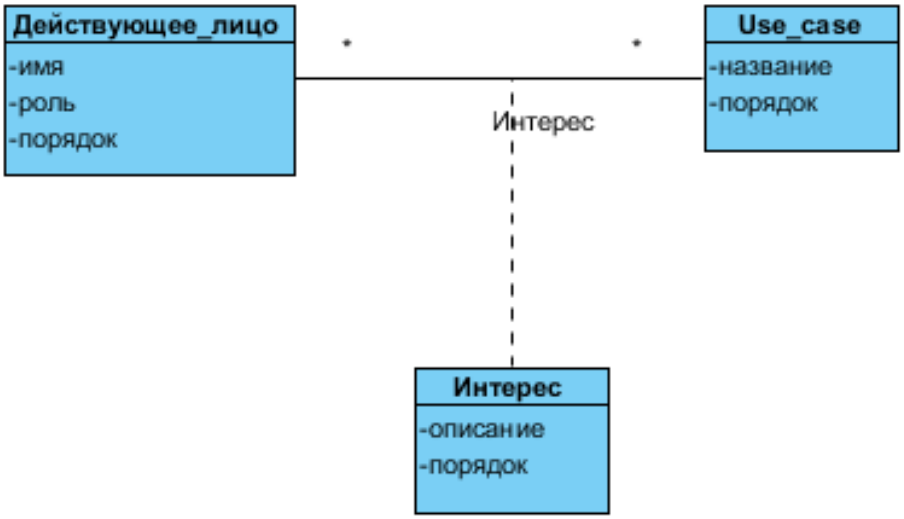

Рис. 1.2. Отношения таблиц, связанных с действующими лицами

У каждого действующего лица есть своя роль. Заинтересованное лицо – один их тех, кто заключает контракт и имеющий законный интерес в поведении системы, реализуемом в варианте использования. Основное действующее лицо – участник, который обращается к системе за одной из ее услуг. Обычно он запускает вариант использования, а цель выносится в его название. Действующее лицо – некто (нечто), обладающий поведением и выполняющий какую-либо функцию в процессе. Бывают вспомогательные действующие лица (они не выносятся в отдельную роль) – действующее лицо, предоставляющее некоторую услугу для разрабатываемой системы, например, интернет. Все участники варианта использования являются действующим лицом, только один, как правило, является основным действующим лицом.

Основной частью в варианте использования является основной сценарий, а также расширение, описывающее альтернативные пути в случае как-либо отказов системы. На рис. 1.3 показаны отношения классов, связанных в данном проекте со сценарием.

Каждый шаг сценария должен описывать действие, которое продви-

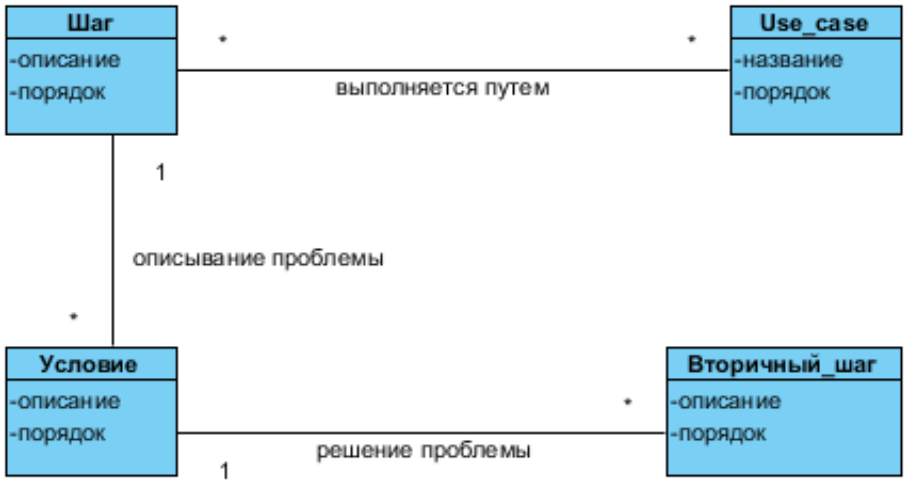

Рис. 1.3. Составляющие сценария

нет процесс на один пункт вперед. Последовательность этих шагов приводит к достижению цели основного действующего лица и удовлетворению интересов всех участников. Если же на каком-либо шаге возникает возможность отказа системы, его условие записывается отдельно в расширение вместе с шагами, необходимыми для успешного выхода из данной ситуации.

1.3. Пример варианта использования

Далее приведен пример варианта использования "Вход в систему".

Основное действующее лицо: Пользователь.

Область действия: Данная система.

Уровень: Цель пользователя.

Участники и интересы: Пользователь – хочет успешно войти в систему для работы. Владелец системы - хочет, чтобы его систему не сломали.

Предусловие: Пользователь зарегистрирован.

Минимальные гарантии: Отсутствуют.

Гарантии успеха: Система подтвердила пользователя, он успешно

начинает работу.

Основной сценарий:

1. Пользователь вводит логин и пароль.
2. Система подтверждает данные.
3. Пользователь переходит на страницу проектов.

Расширения:

1. Неверная пара логин + пароль.
    a. Система уведомляет пользователя.
    б. Пользователь заново вводит данные.
2. Превышен лимит ввода пароля.
    a. Система уведомляет пользователя и прекращает сеанс.
3. Сеть не работает.
    a. Пользователь повторяет попытку позже.

В данной главе была разработана модель предметной области - варианта использования, рассмотрены и описаны основные понятия, с ним связанные, а также приведен пример простого варианта использования. Вариант использования является эффективным средством для работы с управлениями требований и необходим инструмент, работающий с ним.

Глава 2

Проектное решение

Для эффективной работы с вариантами использования необходимо создать удобный инструмент. Было принято решение разрабатывать web-приложение, так как для данной задачи это удобно. К преимуществам web-разработки относятся:

1. пользователю нет необходимости устанавливать программу для работы. Необходимые инструменты для web-приложения это браузер (стандартная программа при установке операционной системы) и доступ в сеть Интернет. При этом, пользователю не обязательно находиться в одном месте, он может зайти в web-приложение в любом месте, где выполняются эти два условия;

2. вся программная логика приложения находится в одном месте - на сервере, то есть существует только одна его рабочая копия, что помогает легко распространять его между пользователями, обновлять версии без необходимости обновления каждому пользователю;

3. архитектура приложения не видна пользователю. Программисты могут выделить отдельный сервер под базу данных, заменить компьютеры на более мощные – пользователь этого никак не заметит и на его работе это не отразится.

Для данного web-приложения была разработана модель, представленная на рис. 2.1.

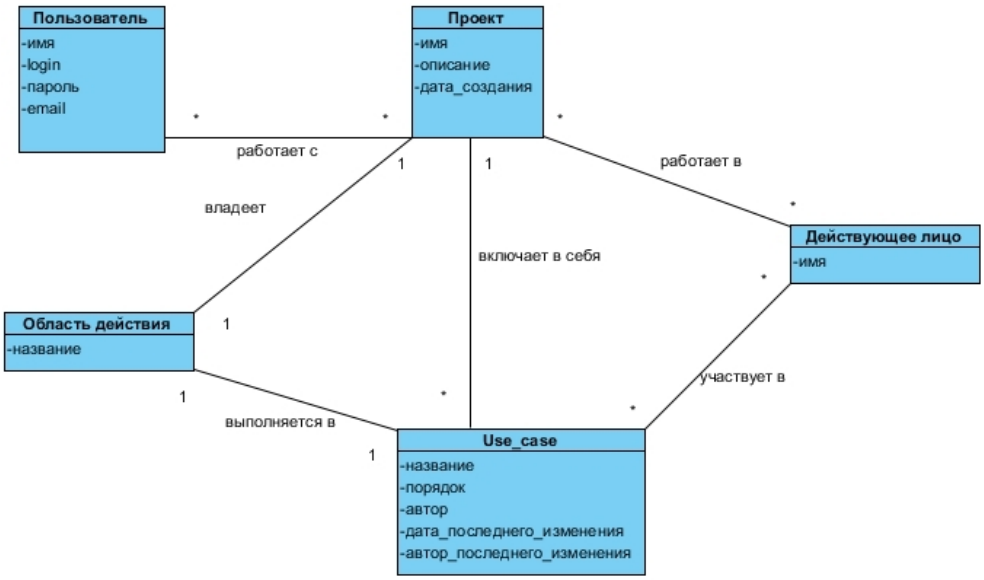

Рис. 2.1. Модель разрабатываемого проектного решения

2.1. Графический интерфейс проектируемой информационной системы управления

На рис. 2.2 приведена диаграмма переходов между страницами.

Изначально пользователь попадает на начальную страницу приложения, представленную на рис. 2.3.

Если пользователь уже зарегистрирован – ему необходимо выполнить авторизацию (рис. 2.4), если нет – пройти регистрацию (рис. 2.5). Обязательными для заполнения являются поля имени, логина и пароля.

Успешно войдя в систему, пользователь попадает на страницу проектов (рис. 2.6), где он может выбрать один из тех проектов, которые ему доступны, или создать новый.

С этой страницы пользователь переходит на страницу вариантов использования выбранного проекта (рис. 2.7) для дальнейшей работы с ними. Также пользователь может перейти на страницу управления проектом (рис. 2.8), посмотреть и изменить список пользователей, работающих в этом проекте.

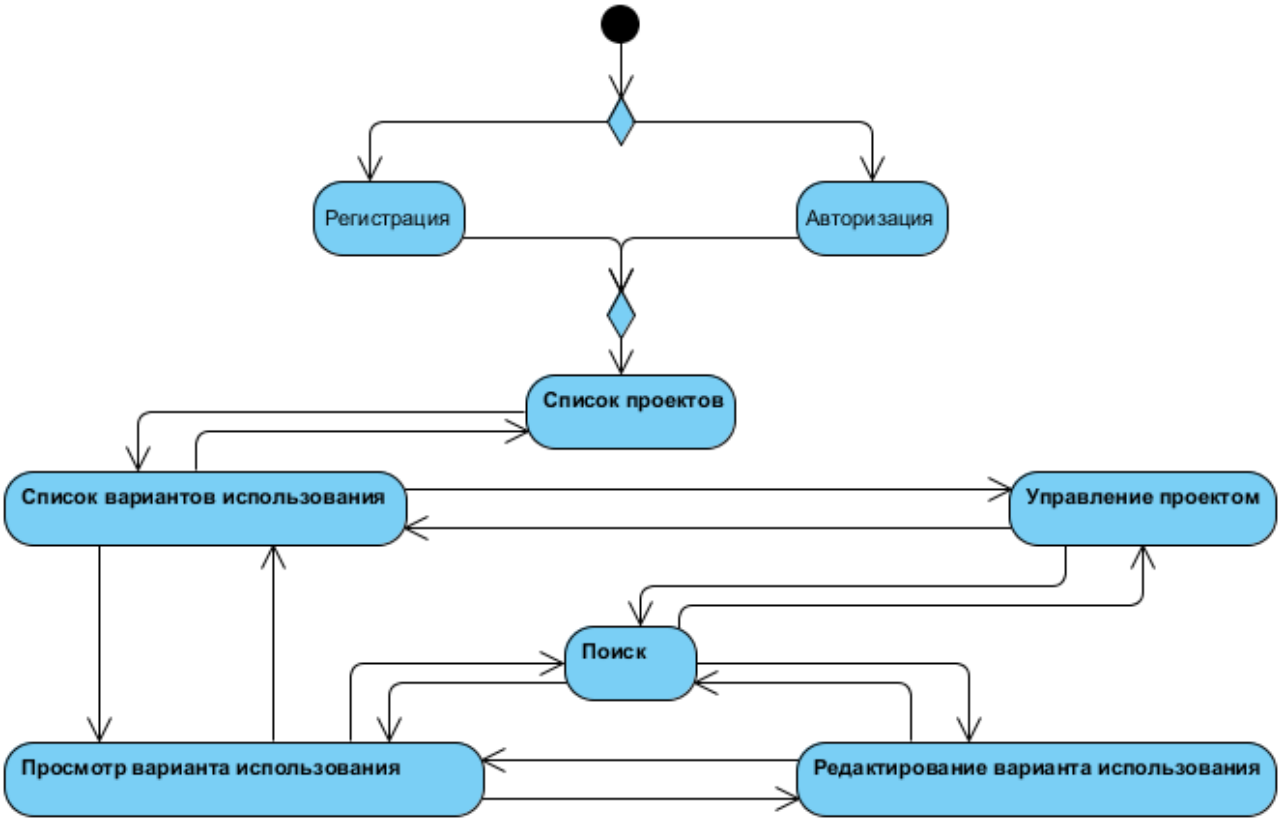

Рис. 2.2. Диаграмма перехода страниц

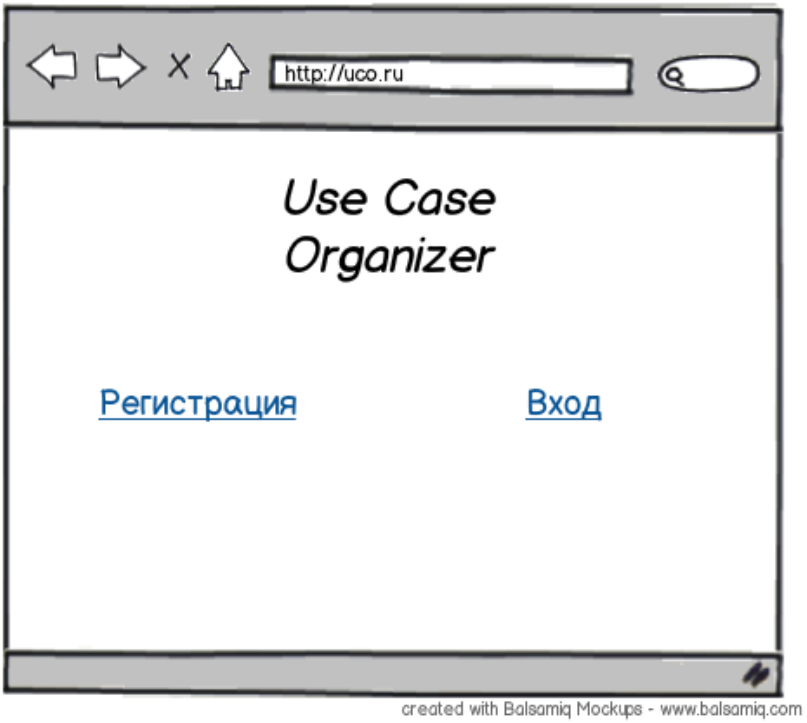

Рис. 2.3. Начальная страница приложения

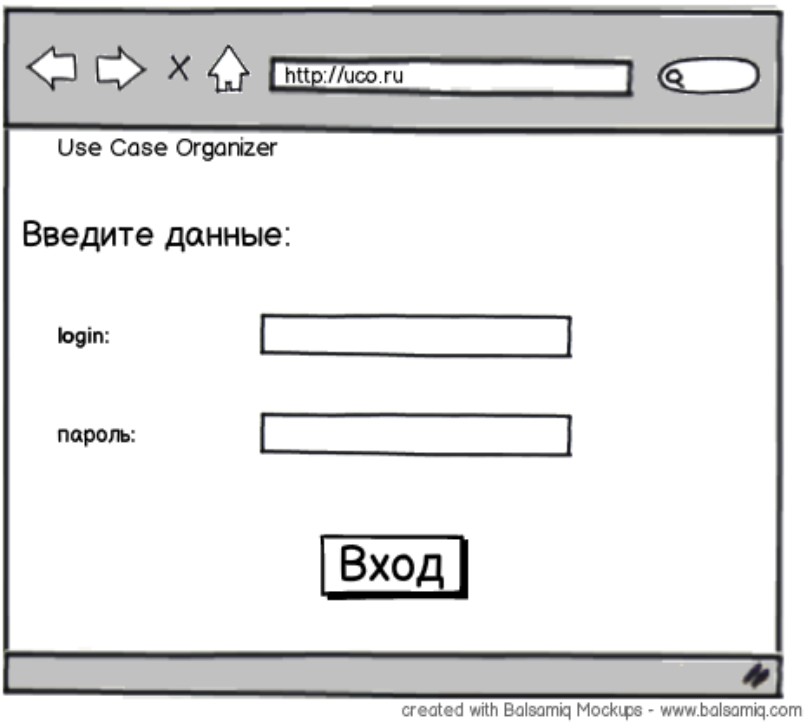

Рис. 2.4. Страница входа

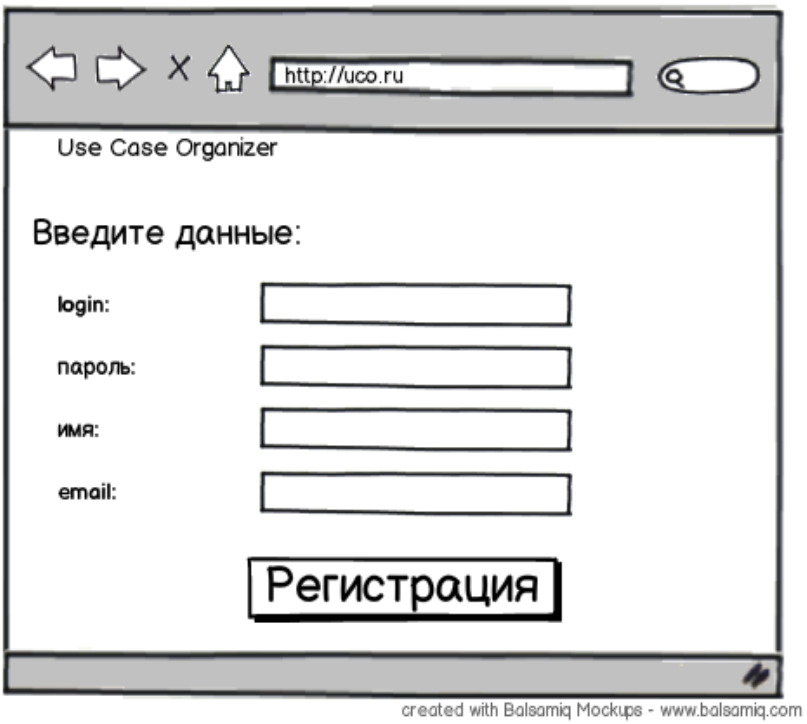

Рис. 2.5. Страница регистрации

Пользователь может просмотреть выбранный вариант использования (рис. 2.9), или отредактировать его (рис. 2.10). При просмотре ва-

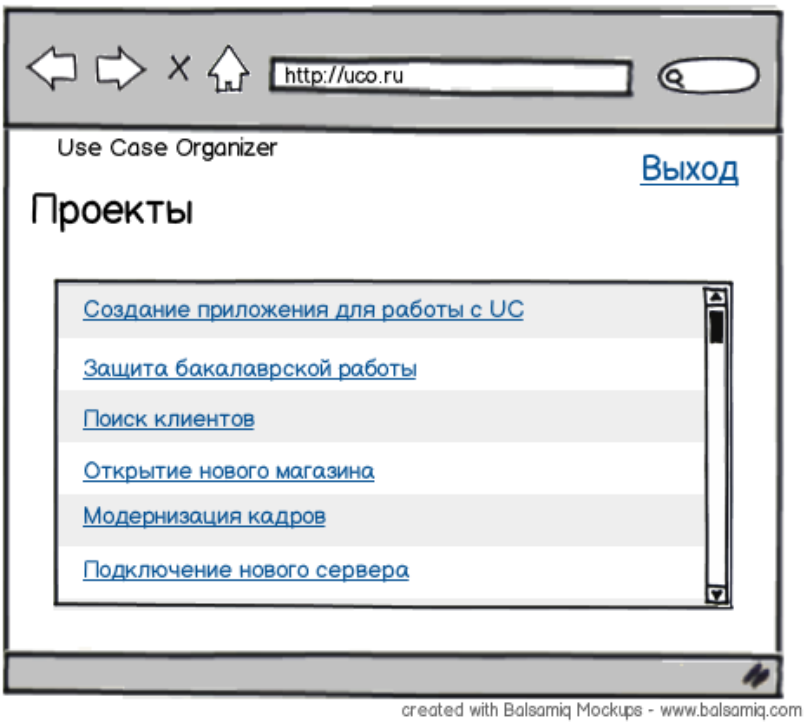

Рис. 2.6. Страница выбора проекта

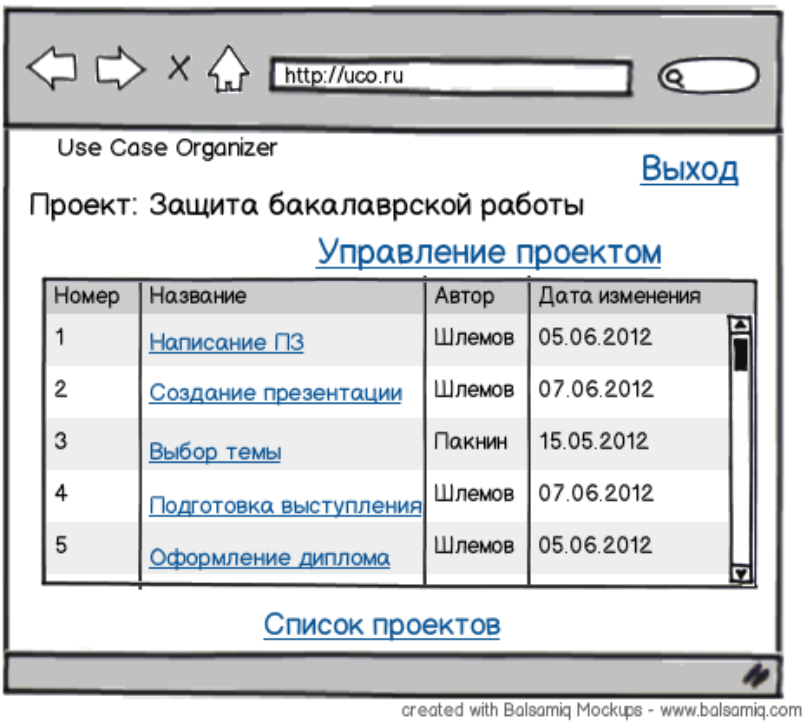

Рис. 2.7. Страница выбора варианта использования

рианта использования также отображается его автор и дата последнего изменения.

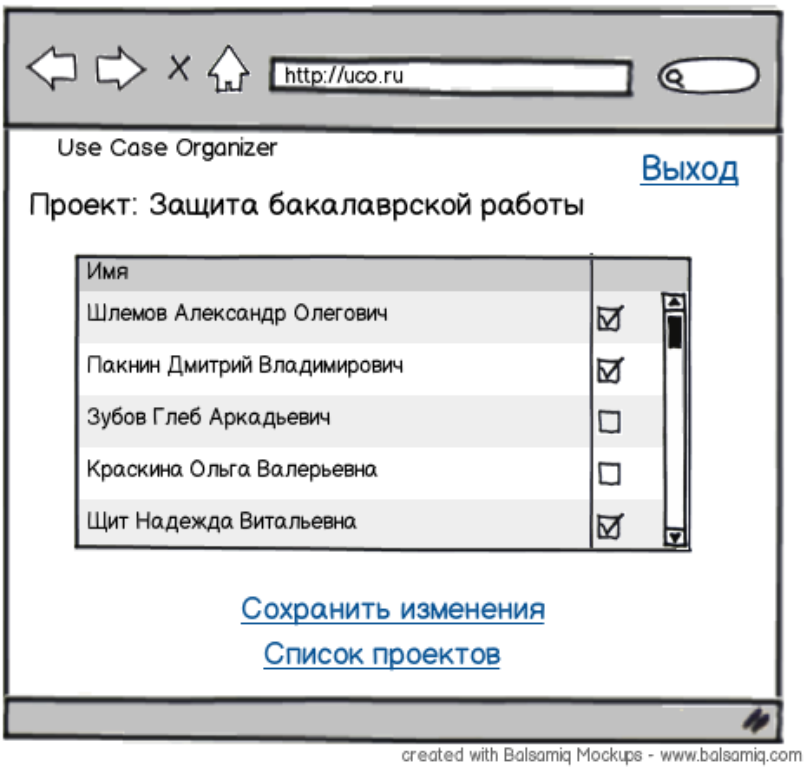

Рис. 2.8. Страница управления проектом

Редактирование варианта использования: пользователь сам вводит название, область действия, предусловие, минимальные гарантии и гарантии успеха. Так как всего существует три возможных уровня – разработчик выбирает один из них. Затем вводятся заинтересованные лица вместе с интересом для каждого их них. Основной сценарий формируется из шагов, нумерация которых происходит автоматически, и пользователь вводит только само описание шага. В расширениях пользователь сам должен записать, к какому шагу оно относится, затем написать его условие и вторичный шаг – решение проблемы. У каждого шага может быть несколько расширений, у каждого условия может быть несколько вторичных шагов для выхода на успешный вариант сценария.

Также пользователь может перейти на страницу поиска – на ней он может просмотреть информацию, собранную по какому-то признаку. Например, пользователь может на странице редактирования варианта

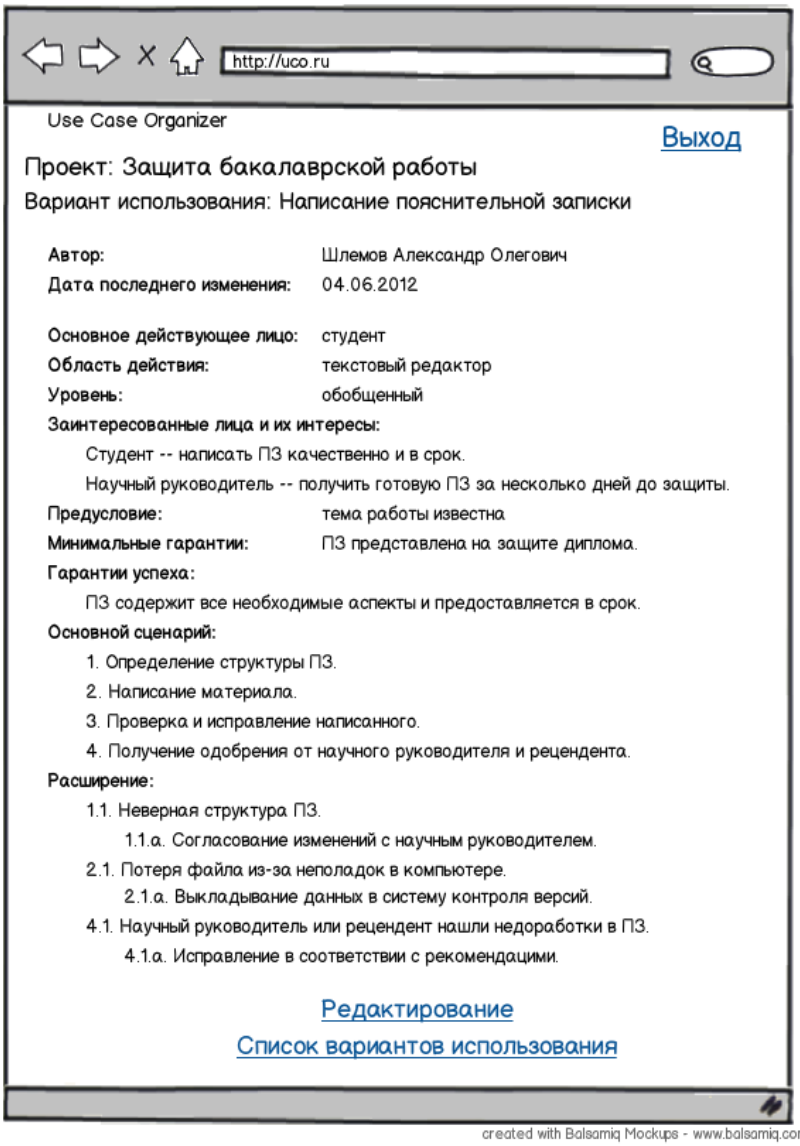

Рис. 2.9. Страница просмотра варианта использования

использования выбрать основное действующее лицо "студент"(он идет с ссылкой) и попадет на страницу поиска, где будут выписаны все варианты использования, в которых участвует данное действующее лицо.

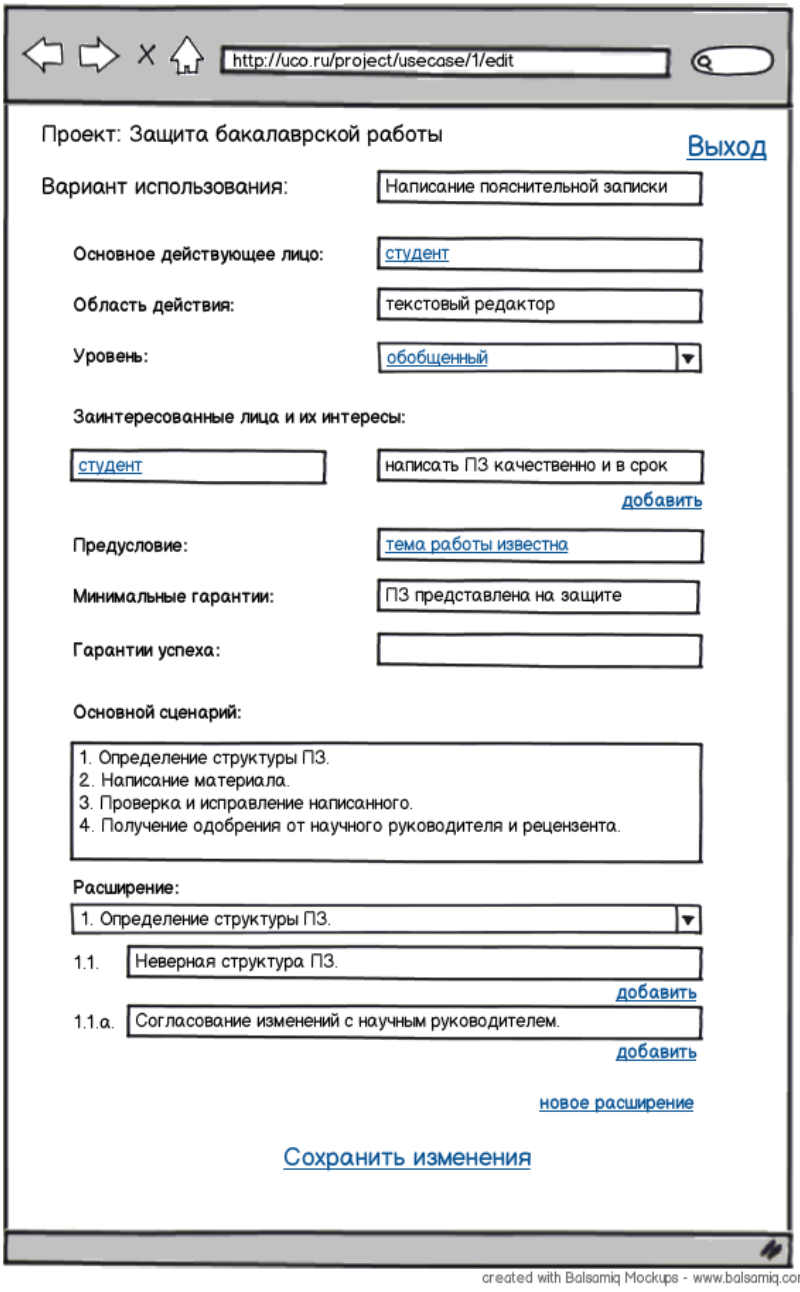

Рис. 2.10. Страница редактирования варианта использования

2.2. ER-диаграммы проектируемой информационной системы управления

На рис. 2.12 и рис. 2.13 представлены ER-диаграммы связей между таблицами базы данных для разрабатываемого приложения. Диаграмма предметной области. Рассматриваются таблицы: Use_case, Уровень, Гарантии_успеха, Предусловие, Шаг, Вторичный_шаг, Условие, Интерес, Действующее_лицо, Область_действия, Минимальные_гарантии. Область действия и уровень для каждого варианта использования единственны, поэтому между соответствующими таблицами существует связь "один к одному". Так как минимальные гарантии и гарантии успеха обычно содержат информацию об удовлетворении интереса нескольких заинтересованных лиц между таблицами Use_case и Минимальные_гарантии и Use_case и Гарантии_успеха организуется связь "один ко многим". Предусловий также может быть несколько для одного варианта использования, поэтому тоже связь "один ко многим но здесь также существует обратная связь "один к одному так как очень часто предусловие является ссылкой на успешно завершенный другой вариант использования. В каждом варианте использования участвует несколько действующих лиц, причем существуют различные роли – основное действующее лицо, заинтересованное лицо, действующее лицо (были описаны ранее), но при этом очень часто действующие лица в рамках одного проекта часто встречаются в различных вариантах использования с одинаковой ролью, поэтому между таблицами Use_case и Действующее_лицо существует связь "многие ко многим"и создается таблица Действующее_лицо_Use_case, которая имеет связь "один к одному"с таблицей Интерес. В данную таблицу записыватся интерес конкретного действующего лица в конткретном варианте использования, если его роль – заинтересованное лицо.

Главной частью в варианте использования является основной сценарий, а также его расширение. Каждый сценарий состоит из шагов, которые могут повторяться в различных вариантах использования – поэтому связь "многие ко многим"и создание таблицы Основной_сценарий, в которой записываются попарно id соответствующих вариантов использования и шагов. Так же существует расширение – описание альтернативного пути для достижения главной цели. Расширение состоит из условия, которое относится к определенному шагу основного сценария и списку вторичных шагов, которые решают создавшуюся проблему. Таким образом, у одного шага может быть несколько условий, а у одного условия может быть несколько вторичных шагов, что привело к созданию связей "один ко многим"между таблицами Шаг и Условие и Условие и Вторичный_шаг.

Проектное решение разработано с учетом принципов Объектно-Ориентированного Программирования: инкапсуляции и принцип открытости-закрытости. Принцип инкапсуляции заключается в разделении интерфейса от реализации. В интерфейсе только описывается, какие методы могут быть использованы, но не показано, каким образом, с помощью каких средств эти методы выполняются. На рис. 2.11 показана схема отношений между присутствующими интерфейсами в данном приложении. Принцип открытости-закрытости построен на том, что программный модуль открыт для дополнений, но закрыт для изменения уже написанного.

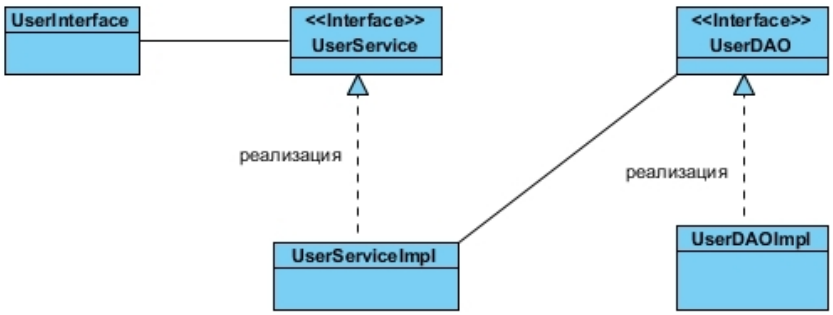

Рис. 2.11. Схема реализации работы с интерфейсом.

2.3. Используемые средства

В данном дипломном проекте использовался язык программирования Java 1.6 с использованием технологий: Wicket 1.5.5, Maven 3.0.4, Mybatis 3.1.0 и Springframework 3.1.1.

2.3.1. Java

Язык программирования Java [9], [10] – один из самых распространенных языков программирования в мире. Данный язык был создан в 1995 году компанией Sun Microsystems. В основе технологии лежит понятие виртуальной машины, которая позволяет отделить язык разработки от целевой платформы путем компиляции исходных текстов в промежуточное представление (байт-код), которое потом и выполняется на виртуальной машине. Java является полностью объектно-ориентированным языком со строгой типизацией. В нем отсутствуют средства работы с прямой памятью, а все объекты размещаются в куче. Сборщик мусора позволяет разработчику сконцентрироваться на логике работы приложения, самостоятельно обеспечивая высвобождение более неиспользуемой памяти. Java является кроссплатформенным, а также предоставляет мощные и удобные средства для работы в сети. Для решения поставленной задачи Java является наиболее подходящим, т.к.:

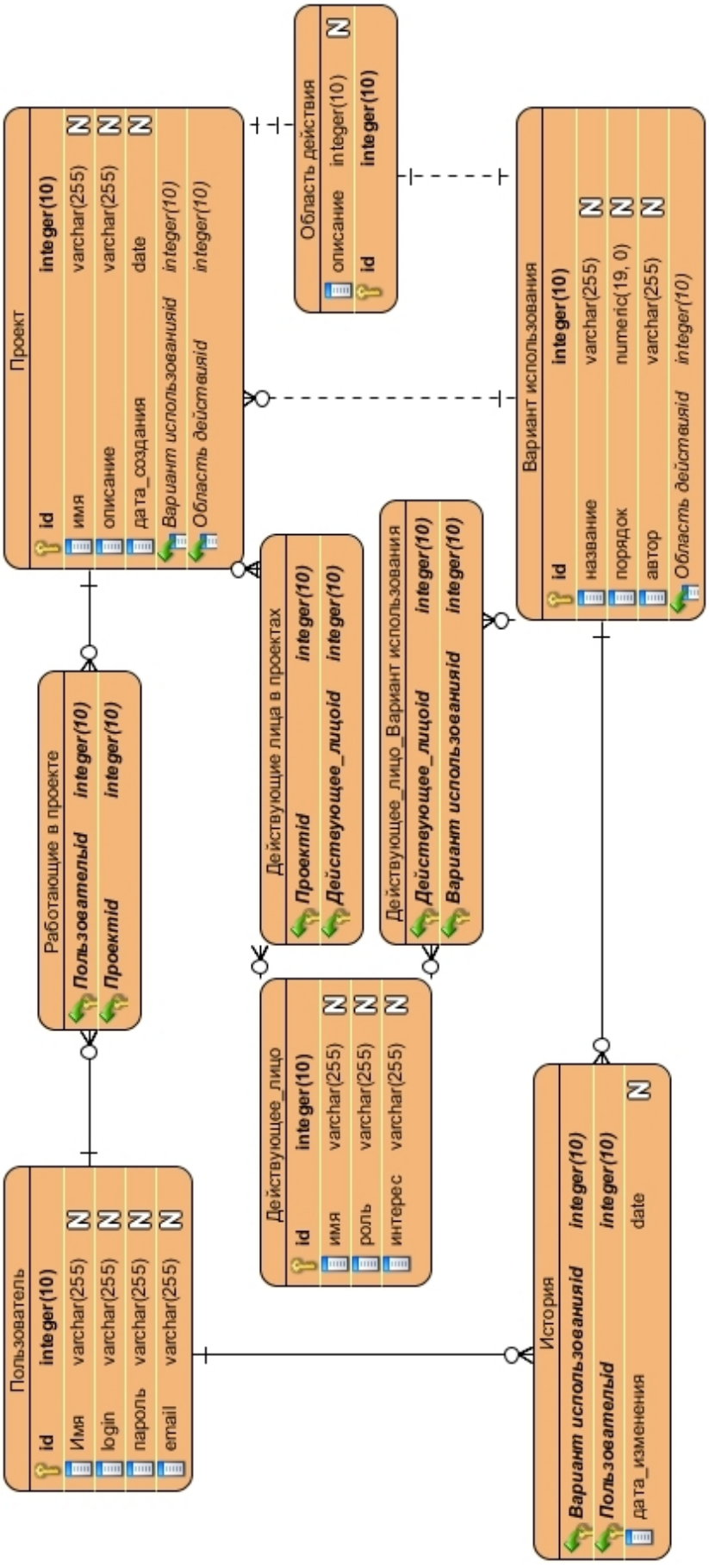

Рис. 2.12. ER-диаграмма модели всего проектного решения

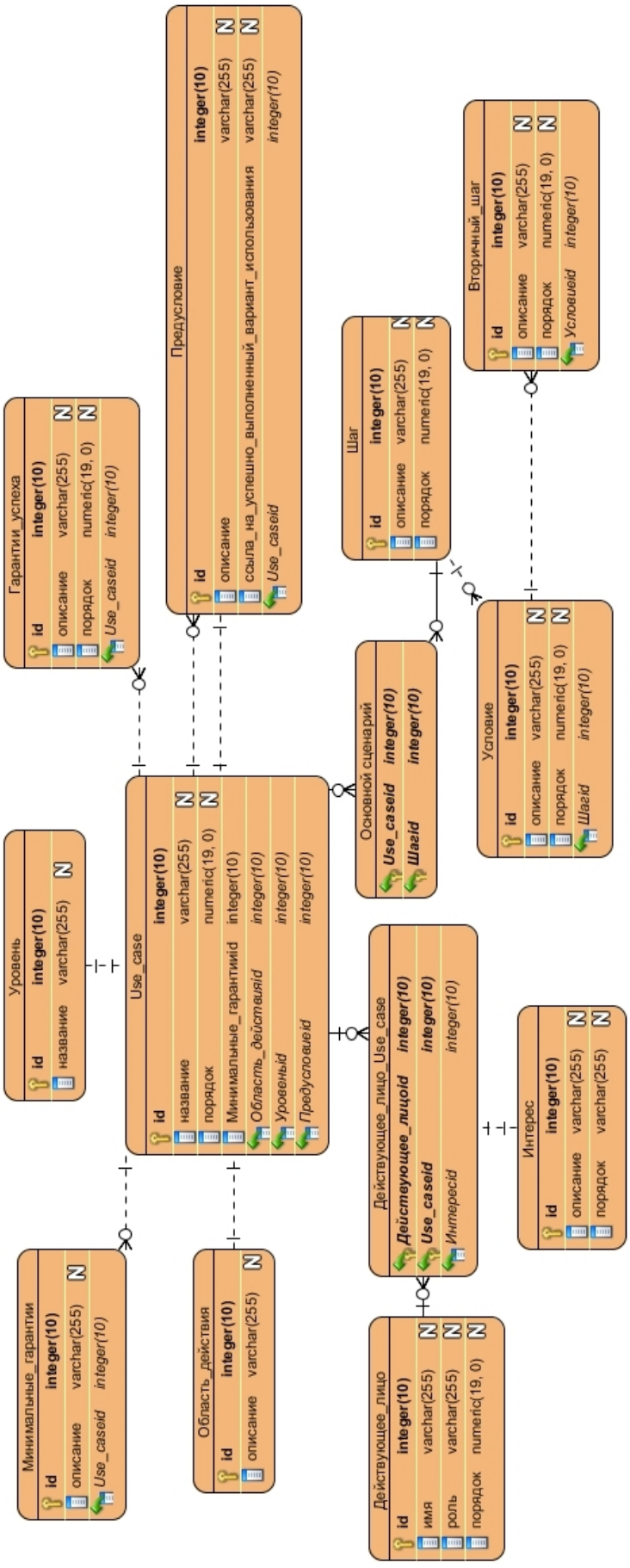

Рис. 2.13. ER-диаграмма модели предметной области

1. разрабатываемая система является web-приложением;
2. является несложным для изучения;
3. предоставляет широкий спектр целевых платформ;
4. обеспечивает максимальную возможную область внедрения системы.

2.3.2. PostreSQL

Используется PostgreSQL[11] – свободная объектно-реляционная система управления базами данных, которая базируется на языке SQL[12]. Данная СУБД была создана в 1995 году студентами преподавателя Майкла Стоунбрейкера из Калифорнийского университета. Достоинства данной СУБД для разработки данного проектного решения:

1. поддержка базы данных практически неограниченного размера, что очень удобно, так как со временем планируется, что у приложения будет много пользователей, которые будут создавать большое количество проектов и вариантов использования;
2. использование системы встроенных языков программирования, в данном случае нам необходимо только PL/Java, что значительно упрощает для разработчика инструмента взаимодействие с базой данных, хранящей все данные;
3. надежность.

2.3.3. Wicket

Работа с проектом, как с web-приложением осуществляется с помощью Wicket [13] – фреймворк с открытым исходным кодом и свободной

лицензией для создания web-приложений, который основан на использовании компонентов. Состояние компонентов пользовательского интерфейса сохраняется, когда пользователь запрашивает новую страницу, и затем восстанавливается, если запрос повторяется. Достоинством Wicket является то, что для работы требуется знать только Java и HTML, настройки записываются в стандартном файле web.xml. Концепция технологии: страница формируется из шаблона страницы (name.html), класса страницы (name.java) и файла свойств (name.properties). Шаблон страницы содержит html-разметку, в которую встраиваются wicket-компоненты (добавление атрибута wicket-id в html тэги). Класс страницы должен содержать все компоненты, указанные в шаблоне (явно создан и добавлен в страницу). Файл свойств необходим для локализации сообщений, выводимых на странице.

2.3.4. Mybatis

Все данные приложения должны храниться в базе данных. Для удобной работы между Java и базой данных (PostgreSQL) используется Mybatis – фреймворк для маппинга объектов СУБД к Java объектам. Конфигурация хранится в файле .xml. Достоинством Mybatis является то, что он умеет заполнять произвольные SQL запросы. Запросы хранятся в .xml файлах, называемых -sqlmap, при этом в запросах содержится информация о Java объектах. Mybatis дает возможность преобразовать результат SQL-запроса на объектную модель приложения, в результате чего разработчик получает дополнительную гибкость при работе с базами данных (особенно тех, структура которых плохо ложится на объектную модель).

2.3.5. Maven

Maven [14] – фреймвокр для автоматизации сборки проектов. Информация для проекта, работающего с Maven содержится в файле pom.xml – общей модели проектов. В ней описываются: имя, версия, авторы, тип проекта, связи с другими проектами, используемые при разработке плагины и описание способа их задействования. Жизненный цикл Maven:

1. validate – проверяется корректность метаинформации о проекте;
2. compile – компиляция исходников;
3. test – выполняются тесты классов из предыдущего шага;
4. package – скомпилированные классы упаковываются в необходимый формат (jar, war);
5. integration-test – упакованные классы отправлятся в среду интеграционного тестирования, прогоняются тесты;
6. verify – проверяется корректность пакета и удовлетворение требованиям качества;
7. install – пакет записывается в локальный репозиторий, откуда пакет будет доступен для использования, как зависимость в других проектах;
8. deploy – пакет отправляется на удаленный сервер, откуда другие разработчики его могут получить и использовать.

Достоинство Maven для данного проекта – возможность подключать различные плагины для успешной работы.

2.3.6. Spring

В официальной документации к Spring Framework написано: "Spring предоставляет облегченное решение по созданию готовых корпоративных приложений, при сохранении возможности декларативного управления транзакциями, дистанционного доступа к вашей логике, использования RMI или Web-служб, средств отправки по почте и различных возможностей обработки данных в базе данных. Потенциально, Spring может стать универсальным пунктом для всех ваших корпоративных приложений, однако Spring является модульной средой, позволяя вам использовать свои части без необходимости вводить остальные."В данном проекте используются следующие Spring модули:

1. spring-core. Базовый модуль, ядро Spring, предоставляет такие фундаментальные средства, как внедрение зависимости и управление компонентами;

2. spring-context. Модуль контекста – второй по важности модуль; содержит такие ключевые классы, как ApplicationContext и WebapplicationCon

3. spring-web. Модуль предоставляет возможность надежной, гибкой, удобной и быстрой разработки Web-приложения.

4. spring-orm. Модуль для объектно-реляционного связывания обеспечивает поддержку для интеграции с Mybatis. Преимуществом использования данного пакета является такие возможности, как легкость проверки, общие исключения данных, постоянное управление ресурсами, интегрированное управление транзакциями корпоративного класса.

В данной главе было разработано проектное решение для информационной системы управления вариантами использования программного

обеспечения, показана модель и разработанный интерфейс будущего приложения, так же были показаны взаимосвязи в используемой базе данным и выполнено краткое описание используемых средств и технологий.

Глава 3

Заключение

В результате данного дипломного проекта была разработана модель предметной области вариантов использования программного обеспечения и проектное решение для информационной системы управления вариантами использования программного обеспечения. Так же были представлены предварительные наработки по интерфейсу и логике работы будущего web-приложения, которое будет создано в дальнейшем.

Литература

1. *Макконнелл, С.* Совершенный код / С. Макконнелл. — Питер, 2007.

2. Исследование выполнения в срок разработок программного обеспечения. — 2009. http://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited/.

3. *Расмуссон, Джонатан.* Гибкое управление IT-проектами. Руководство для настоящих самураев. / Джонатан Расмуссон. — Питер, 2012.

4. *Элизабет Халл Кен Джексон, Джереми Дик.* Разработка и управление требованиями / Джереми Дик Элизабет Халл, Кен Джексон. — 2005.

5. *Коберн, Алистер.* Современные методы описания функциональных требований к системам / Алистер Коберн. — Лори, 2002.

6. Официальный сайт продукта CaliberRM. http://www.borland.com/products/caliber/.

7. Официальный сайт продукта Use Case Tool. http://uctool.sourceforge.net/.

8. Официальный сайт продукта Wiki Confluence. http://www.atlassian.com/software/confluence/overview/.

9. *Хорстманн, Кей.* Java 2. Библиотека профессионала. Том 1. Основы. / Кей Хорстманн. — Вильямс, 2010.

10. *Хорстманн, Кей.* Java 2. Библиотека профессионала. Том 2. Тонкости программирования. / Кей Хорстманн. — Вильямс, 2010.

11. *Дж. Уорсли, Дж. Дрейк.* PostgreSQL. Для профессионалов / Дж. Дрейк Дж. Уорсли. — Питер, 2003.

12. *Бьюли, Алан.* Изучаем SQL / Алан Бьюли. — Символ-Плюс, 2007.

13. *Vaynberg, Igor.* Apache Wicket Cookbook / Igor Vaynberg. — Packt Publishing, 2011.

14. *Srirangan.* Apache Maven 3 / Srirangan. — Packt Publishing, 2011.