Государственное бюджетное образовательное учреждение
высшего образования Московской области
«Университет «Дубна»
Филиал «Протвино»
Кафедра «Информационные технологии»

Т.Н. Кульман

Подготовка и оформление курсовых работ по дисциплине «Теория и технология проектирования»

Электронное методическое пособие

Рекомендовано
кафедрой информационных технологий
филиала «Протвино» государственного университета
«Дубна»
в качестве методического пособия для студентов,
обучающихся по направлению
«Информатика и вычислительная техника»

Протвино
2016

ББК 32.973.2-018.1
К90

Рецензент:
кандидат физико-математических наук,
главный специалист ООО «Систел»
Е.В. Клименков

Кульман, Т.Н.
К90 Подготовка и оформление курсовых работ по дисциплине «Теория и технология проектирования»: электронное методическое пособие / Т.Н. Кульман. — Протвино: 2016. — 44с.

Предназначено для студентов очного и заочного отделений направления «Информатика и вычислительная техника».

В пособии рассматриваются правила подготовки, определяются требования к содержанию, структуре, выбору темы и оформлению курсовых работ. Приведен пример выполнения курсовой работы.

Исполнение настоящих требований обязательно для всех преподавателей кафедры информационных технологий, осуществляющих руководство курсовыми работами, и для всех студентов, выполняющих курсовые работы по дисциплине «Теория и технология проектирования».

ББК 32.973.2-018.1

Введение

Дисциплина «Теория и технология проектирования» (ТТП) посвящена изучению жизненного цикла программного обеспечения /ПО/, принципов проектирования информационных систем /ИС/, структурному и объектно-ориентированному подходам в проектировании. Особое внимание уделяется объектно-ориентированному подходу, освоению унифицированного языка моделирования UML. Рассматриваются бизнес-процессы, спецификация требований, анализ и проектирование ПО. Изучают-ся теоретические и практические вопросы технологии создания (*CASE*-технологии) ПО.

Учебно-методическое пособие предназначено для студентов очного и заочного отделений специальности «Программное обеспечение вычислительной техники и автоматизированных систем» и направления «Информатика и вычислительная техника».

В данном пособии использовались Положение о выполнении и защите курсовых работ в университете «Дубна» [1] и рекомендации по выполнению курсовых работ [2, 3].

Результатом освоения дисциплины «Теория и технология про-ектирования» является выполнение курсовой работы. Главными задачами при этом будут:

– овладение комплексом знаний по теоретическим и прикладным основам проектирования ИС на базе объектно-ориентированного подхода;

– формирование навыков самостоятельного практического применения современных методов и средств проектирования ПО информационных систем;

– подробное изучение инструментальных средств *IBM Rational Rose* в части построения моделей и диаграмм языка *UML*;

– приобретение практического опыта взаимодействия с заказчиком;

– проведение проектирования конкретной ИС в избранной предметной области (возможно, её фрагмента) и доведение (в отдельных случаях) разработки до получения работоспособной базы данных;

– обучение представлению, оформлению и описанию работы, используя стандарты подготовки презентаций и публикаций на компьютере.

Выполнение данной курсовой работы приводит к первым контактам с заказчиками ИС, к осмыслению практического процесса

проектирования, к осознанию ответственности за выполнение проектирования и разработки системы, — всё это является фундаментом профессиональных и практических навыков студентов, востребованных в их будущей специальности.

Необходимо отметить, что при разработке курсовой рабо-ты можно использовать свободно распространяемый продукт *StarUML,* являющийся инструментом моделирования программного обеспечения и поддерживающий большинство типов диаграмм *UML.*

1 Общие требования к курсовой работе

1.1 Правила выбора темы для курсовой работы

По давней традиции кафедры Информационных технологий, тему для курсовой работы по ТТП большинство студентов выбирает самостоятельно с поиском заказчика темы и с консультациями по вопросам проектирования в выбранной предметной области. Такой подход обучает студентов работе с заказчиком, разработке системы, начиная с формулирования требований, спецификаций, согласованию требований и функциональных возможностей с заказчиком. Студенты приближены к «реальной действительности» и получают хороший урок в избранной профессии. Если студент не может самостоятельно выбрать тему, – её ему предлагает преподаватель.

Темы курсовых работ формулируются в течение первого месяца семестра, в котором они выполняются. Далее темы утверждаются заведующим кафедрой. Как правило, за каждым студентом закрепляется индивидуальная тема. Однако, в случае сложных ра-бот, одну тему могут выполнять двое или трое студентов. Если студент не справляется с заданной темой, возможна её замена с последующим снижением оценки.

Руководитель составляет задание на курсовую работу, в которой приводятся: целевая установка и исходные данные; основные вопросы, подлежащие решению; ожидаемые результаты и предполагаемая практическая реализация, а также временные сроки (начало выполнения работы, контрольные точки, срок сдачи). Задание может быть выдано в электронном виде.

Как правило, руководителем курсовой работы является преподаватель, ведущий семинарские занятия. Работа проводится студентом самостоятельно, преподаватель оказывает помощь и консультации, а также осуществляет контроль своевременности и качества выполняемых этапов.

1.2 Защита курсовой работы

В результате выполнения курсовой работы студенты за две не-дели до установленного срока защиты представляют руководите-лю:

– законченный вариант модели проекта системы с использо-ванием инструментальных средств *IBM Rational Rose*;

– презентацию в электронном виде;

– отчет о курсовой работе в виде файла формата *MS Word* или аналогичном в *Open Office*.

Отчёт оформляется в соответствии с правилами, описанными в п. 3.2. Время выступления – около 10 минут.

Не допускаются к защите курсовые работы, полностью или в значительной степени выполненные не самостоятельно, а также небрежно оформленные.

Защита курсовой работы проводится до начала зачётной сес-сии и проходит в открытом режиме с демонстрацией презентации и выступлением перед аудиторией. В этом случае действительно необходимо «защищать» свою работу, уметь рассказать о ней и от-ветить на возникшие вопросы.

1.3 Последовательность выполнения работы

При выполнении курсовой работы рекомендуется придержи-ваться следующей последовательности шагов:

– установление контактов с заказчиком,

– выбор темы самостоятельно или с помощью преподавате-ля (из утверждённого списка),

– составление глоссария проекта,

– создание модели вариантов использования,

– анализ вариантов использования,

– проектирование системы,

– реализация системы,

– подготовка презентации;

– подготовка текста отчёта,

– представление и обсуждение результатов в ходе семестра,

– защита курсовой работы.

Понятно, что некоторые шаги могут проводиться параллельно и итерационно. На протяжении всей работы предполагается со-гласование принимаемых решений с заказчиком.

1.4 Критерии оценки курсовой работы

Оценка за курсовую работу учитывает:

1. самостоятельность и оригинальность работы;
2. степень сложности разработанной модели;

3. наличие описания элементов модели в виде присоединенных текстовых файлов;

4. создание различных диаграмм на основе языка моделирования *UML*, в том числе диаграмм состояний и диаграммы «сущность – связь», отображающей схему базы данных.

5. оформление курсовой работы с использованием современных компьютерных выразительных средств;

6. качество презентации, умение коротко и ясно рассказать о своей работе;

7. умение представить результаты работы в письменном виде;

8. понимание и знания, проявленные при ответах на вопросы во время защиты.

Рекомендуемые критерии оценки [1]:

– «отлично» выставляется студенту, показавшему глубокие знания, примененные им при самостоятельном исследовании избранной темы, способному обобщить практический материал и сделать на основе анализа выводы;

– «хорошо» выставляется студенту, показавшему в работе и при ее защите полное знание материала, всесторонне осветившему вопросы темы, но не в полной мере проявившему самостоятельность в исследовании;

– «удовлетворительно» выставляется студенту, раскрывшему в работе основные вопросы избранной темы, но не проявившему самостоятельности в анализе или допустившему отдельные неточности в содержании работы;

– «неудовлетворительно» выставляется студенту, не раскрывшему основные положения избранной темы и допустившему грубые ошибки в содержании работы.

При получении неудовлетворительной оценки работа должна быть переработана с учетом высказанных замечаний и представлена на защиту в сроки, установленные преподавателем. В случае несогласия студента с поставленной оценкой, он имеет право об-ратиться на кафедру с апелляцией. В этом случае кафедра назна-чает комиссию для повторной защиты.

2 Средства проектирования, необходимые при выполнении курсовой работы

Курсовая работа выполняется с использованием объектно-ориентированных методов анализа и проектирования ПО. Большинство современных методов основано на применении языка

визуального моделирования *UML*. Проектирование ведётся в рам-ках *CASE*-средств, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

2.1 Объектно-ориентированный подход Объектно-ориентированный подход (ООП), как один из способов проектирования и разработки программных систем, основан на понятии объекта. Современное программирование моделирует окружающий нас мир или его небольшую часть, т.е. программы строятся на основе объектов. «Объекты, как абстракции реально-го мира, представляют собой отдельные насыщенные связные информационные единицы » [6]. Объект является экземпляром класса, класс – это описание множества однотипных объектов.

Классическое изложение фундаментальных понятий и принципов объектно-ориентированного анализа и проектирования содержится в книге Г.Буча [6], краткое изложение ООП можно найти в [4, 8, 9, 12].

В рамках рассматриваемого подхода разработка любой системы начинается с проектирования и создания набора классов. Классы можно расширять путём добавления новых свойств и методов. Классы, обладая определённым поведением и реакцией на изменение внешних условий, взаимодействуют между собой, принимают, обрабатывают и передают данные, и, таким образом, решают необходимые задачи.

Для ООП концептуальной базой является объектная модель. В ней выделяется четыре главных элемента:

- абстрагирование,
- инкапсуляция,
- модульность,
- иерархия

и три дополнительных:

- типизация,
- параллелизм,
- сохраняемость.

Объектно-ориентированный подход возник как реакция на преодоление растущей сложности программных систем. Вслед за объектно-ориентированным программированием (языки *Simula, Smalltalk, C++* и др.) появились объектно-ориентированные анализ и проектирование.

Для реализации приведённых выше элементов объектных моделей применяется язык моделирования *UML*.

2.2 Унифицированный язык моделирования UML

Унифицированный язык моделирования *UML (Unifi ed Modeling Language)* представляет собой язык для определения, представления, проектирования и документирования программных систем в различных предметных областях. Описание этого языка можно найти в литературе [7—12], а также на соответствующих сайтах. *UML* содержит стандартный набор диаграмм и нотаций, с помощью которых осуществляется визуальное моделирование.

Рассмотрим 2 группы моделей и соответствующие им диаграммы [4].

Структурные *(structural),* или статические, модели:

– диаграммы классов *(class diagrams)* — для моделирования статической структуры классов системы и связей между ними;

– диаграммы компонентов *(component diagrams)* — для моделирования иерархии компонентов системы;

– диаграммы размещения *(deployment diagrams)* — для моделирования физической архитектуры системы.

Модели поведения *(behavioral),* или динамические:

– диаграммы вариантов использования *(use case diagrams)* — для моделирования функциональных требований к системе (в виде сценариев взаимодействия пользователей с системой);

– диаграммы взаимодействия *(interaction diagrams)* — для моделирования поведения объектов системы при переходе из одного состояния в другое. Существует два вида диаграмм взаимодействия:

- диаграммы последовательности *(sequence diagrams)* и
- кооперативные диаграммы *(collaboration diagrams)*;

– диаграммы состояний *(statechart diagrams)* — для моделирования поведения объектов системы при переходе из одного состояния в другое;

– диаграммы деятельности *(activity diagrams)* — для моделирования поведения системы в рамках различных вариантов использования, или потоков управления.

Как видно из приведённого выше перечня, диаграммы *UML* позволяют описать разрабатываемую систему с различных точек зрения и позволяют объединить отдельные представления в единое целое.

2.3 Краткое описание диаграмм UML

Диаграммы вариантов использования предназначены для описания функциональных требований в виде вариантов использования. Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым объектом (действующим лицом) и отражает функциональные требования к системе с точки зрения пользователей. Цель построения диаграмм вариантов использова-ния — это моделирование и документирование функциональных требований в самом общем виде.

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов в рамках варианта использования или некоторой операции класса. На диаграммах отображаются объекты и сообщения, которыми они обмениваются между собой.
В дальнейшем объекты соотносятся с классами, а сообщения диа-грамм взаимодействия — с операциями классов.

Диаграммы классов определяют множества классов, интерфейсов, кооперaций, отношений между ними и моделируют статический вид системы. На диаграммах классов изображаются атрибуты и операции классов, а также ограничения, которые накладываются на связи между классами. Диаграммы классов применяются для моделирования словаря системы, простых кооперaций и логической схемы базы данных. Отношения между класса-ми могут быть весьма сложными, для реализации этих отношений используются зависимости, обобщения и ассоциации.

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также про-цесс смены состояний объекта в результате наступления некото-рых событий. Диаграммы состояний не надо создавать для каж-дого класса, они применяются только в сложных случаях. Если объект может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, для него может потребоваться такая диаграмма.

Диаграммы деятельности очень похожи на блок-схемы, которые показывают, как поток управления переходит от одной деятельности к другой, то есть описывается последовательность выполнения операций во времени. Такие диаграммы являются полезными при описании поведения, включающего большое количество параллельных процессов и при программировании этих процессов, поскольку можно графически изобразить все ветви и определить, когда их необходимо синхронизировать.

Интересно отметить [7], что диаграмму деятельности можно «представить как вывернутую наизнанку» диаграмму взаимодействий. Диаграмма взаимодействий — это взгляд на объекты, которые передают друг другу сообщения, а диаграмма деятельно-сти — взгляд на операции, которые передаются от одного объекта другому.

Диаграммы компонентов моделируют физический уровень системы. На них изображаются компоненты ПО и связи между ними. Компонентами называются физические модули кода, представляющие собой как библиотеки исходного кода, так и исполняемые файлы. Например, если программирование систе-мы ведётся на *C++*, то отдельными компонентами будут файлы *.CPP, .H, .EXE*. Зависимости между компонентами отражают порядок их компиляции.

Диаграмма размещения создаётся для отображения физических взаимосвязей между программными и аппаратными компонентами системы. Она представляет физическую реализацию готового приложения и является хорошим средством для того, чтобы показать размещение объектов и компонентов в распределенной системе. Как правило, диаграмма размещения создаётся одна на проект.

Детальное рассмотрение всех диаграмм с примерами их применения приведено в книгах [7—12].

2.4 CASE-средства

CASE-средства *(Computer Aided Software Engineering)* — пред-ставляют собой совокупность методологий анализа, проектирова-ния, разработки и сопровождения сложных систем программного обеспечения [13].

В настоящее время технология создания ПО, как упорядоченная совокупность взаимосвязанных технологических процессов в рамках жизненного цикла ПО [4], должна поддерживаться инструментальными *CASE*-средствами. Практически все ведущие компании по разработке технологий и программных продуктов (*IBM, Microsoft, Oracle, Borland, Computer Associates, Sybase* и др.) располагают развитыми технологиями создания ПО; их краткое описание приведено в работе [4]. Обзор Российского рынка *CASE*-средств можно найти в книге [13].

При подготовке курсовой работы рекомендуется применять *CASE*-средство *IBM Rational Rose*, являющееся средством визуального моделирования, использующим язык *UML*.

IBM Rational Rose входит в состав технологии *IBM Rational Suite*, являющейся одной из современных технологий для моделирова-ния программных систем с использованием широкого круга ин-струментальных средств и платформ.

Основными компонентами *IBM Rational Suite* являются:

– репозиторий, представляющий собой базу данных проекта;

– графический интерфейс;

– браузер — средство навигации по проекту;

– средства контроля проекта и сбора статистики — позволяет создавать, изменять и проверять корректность модели;

– генератор документов;

– генератор кода для поддерживаемых языков, включая *C/C++*. Создаются заготовки программ для последующей работы программистов;

– инструмент для создания схемы БД с генерацией описания на языке *SQL*.

IBM Rational Rose предлагает плавный процесс (итерационный и инкрементный) разработки ИС. Любые модели, создаваемые с помощью данного средства, являются взаимосвязанными: бизнес-модель, функциональная модель, модель анализа, модель проектирования, модель базы данных, модель компонентов и модель физического развертывания системы.

3 Оформление отчёта по курсовой работе

3.1 Структура отчёта

Как упоминалось выше (см. п. 1.3), курсовая работа представляет собой объединённую в единое целое совокупность 3-х компонент: законченной модели ПО, электронной презентации и отчёта. В данном разделе речь пойдёт о подготовке отчёта.

Отчет по курсовой работе предоставляется в электронном виде и в бумажной копии в папке. **Титульный лист** отчёта вклю-чает все необходимые исходные данные о работе и приведён в Приложении А.

Поскольку разрабатываемая система представляется в виде нескольких различных взаимосвязанных между собой моделей, то и структура отчёта будет соответствовать этим моделям.

«Оглавление» включает в себя названия глав, параграфов и номера листов.

Во «**Введении»** необходимо кратко описать:

– цель выполнения данной работы,

– методы исследования,
– предметную область (кратко),
– формулирование задач (задачи), которые нужно решить для достижения поставленной цели.

Первая глава «Постановка задачи» содержит описание задания.

Вторая глава «Анализ требований» описывает глоссарий, диаграмму вариантов использования, действующих лиц и варианты использования.

В **Третьей главе** «Анализ системы» приводятся диаграммы взаимодействия между объектами (последовательности и кооперативные), соответствующие потокам событий вариантов использования, диаграммы классов анализа системы. Диаграммы сопровождаются пояснениями к объектам, присутствующим на них.

Четвертая глава «Проектирование» содержит иерархию классов системы и определение пакетов. Для каждого класса дается описание, включающее в себя описания ответственности класса, его атрибутов и операций. В этой главе производится преобразование классов «анализа» в проектные классы. Сложный класс «анализа» может быть разбит на несколько классов, преобразован в пакет или подсистему. Также выделяются архитектурные уровни, приводятся диаграммы классов системы, отображающие связи между классами, и диаграммы состояний, описывающие поведение экземпляров отдельных классов.

Пятая глава «Реализация» характеризует физический уровень системы, в ней приводятся диаграммы компонентов для каждого пакета и для системы в целом, а также диаграмма размещения с комментариями. Если вариант задания предполагает создание схемы БД, то такая схема также должна быть включена в отчет.

В **«Заключении»** подводится общий итог работы, описываются полученные результаты, можно привести предложения по развитию моделей проекта. Завершают отчет «**Приложения»** и «**Библиографический список»,** подготовка которых рассматривается в п. 3.6, 3.7.

3.2 Правила оформления

Отчёт по курсовой работе печатается на стандартных листах формата А4 (210х297 мм). **Объем работы должен составлять 25—30 страниц** компьютерного текста, набранного шрифтом *Times New Roman* черного цвета с полуторным интервалом,

высота букв, цифр и других знаков — не менее 1,8 мм (кегль равен 12). Полужирный шрифт не применяется. Абзацный отступ — 1,25 (5 знаков). Напечатанный текст должен иметь поля: верхнее — 20 мм, правое — 10 мм, левое — 30 мм, нижнее — 20 мм.

При оформлении текста необходимо соблюдать следующие правила [1, 2]:

1. Страницы должны иметь сквозную нумерацию, вклю-чая приложения (номер указывается в центре нижнего поля без точки), при этом титульный лист считается первой страницей, оглавление — второй, введение — третьей и так далее. Номер страницы на титульном листе не проставляется.
2. Не допускается оставлять на листе первые и последние строки абзацев.
3. В оглавлении работы указывается перечень всех глав и параграфов курсовой работы, а также номера страниц, с которых они начинаются. Слово «ОГЛАВЛЕНИЕ» размещается по центру страницы в виде заголовка прописными буквами.
4. Введение и заключение не нумеруются.
5. Главы, параграфы, пункты работы должны иметь порядковые номера в пределах всего документа, обозначаются арабскими цифрами, разделённые точками и записанные с абзацного отступа, после номера в тексте точку не ставят.
6. Следует использовать режим выравнивания «по ширине».
7. Клавишей *Enter*, в основном, нужно пользоваться только в конце абзаца.
8. Для подстрочных/надстрочных символов, а также для записи числителя и знаменателя дробей следует использовать 10 кегль.
9. Греческие буквы в тексте и формулах должны быть курсивными.
10. После знаков препинания: точка, запятая, двоеточие, вопросительный и восклицательный знак — ставится пробел (отбивка); тире отбивается с двух сторон; дефис не отбивается; внутри скобок и кавычек пробел не ставится.
11. Заголовки в тексте и оглавлении должны совпадать.
12. Разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры.
13. В конце работы приводится библиографический список, оформленный в соответствии со стандартами.

3.3 Формулы

При описании математических формул нужно использовать редактор формул (*Microsoft Equation*). Ссылки в тексте на порядковые номера формул дают в круглых скобках, например: в формуле (1).

Расшифровка символов, входящих в формулу, должна быть приведена непосредственно под формулой. Значения каждого символа записывают с новой строки в той последовательности, в какой они приведены в формуле. Первая строка расшифровки должна начинаться со слова «где» без двоеточия после него, например:

$$r = \sqrt{\frac{S}{\pi}} \qquad (1)$$

где r — радиус окружности;

S — площадь круга;

π — число π.

3.4 Рисунки

Рисунки следует нумеровать арабскими цифрами сквозной нумерацией. Каждый рисунок должен быть подписан, при этом используется слово «Рисунок». Слово «Рисунок» и наименование располагают посередине строки без кавычек, например, следующим образом: Рисунок 1 — Диаграмма вариантов использования. В конце наименования рисунка точка не ставится. Сокращение слова «Рисунок» не допускается.

3.5 Таблицы

Таблицы применяют для представления цифровой информации, для наглядности и удобства сопоставления значений. Название таблицы должно отражать её содержание, быть точным и кратким. Номер таблицы следует помещать над таблицей слева без абзацного отступа. Название таблицы располагают в одну строчку с ее номером через тире.

Например: Таблица 1 — Глоссарий проекта.

Таблица (номер) — (название таблицы)

На все формулы, рисунки и таблицы обязательно должны быть ссылки в тексте отчета курсовой работы.

3.6 Приложения

В приложении могут размещаться фрагменты описания схемы БД, программного кода, графики, таблицы с результатами и т. п.

Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением букв Ё, З, Й, О, Ч, Ь, Ы, Ъ. После слова «Приложение» следует буква, обозначающая его последовательность.

Например: Приложение А Генерация кода описания БД. Если в работе используется специфическая терминология, то в конце работы может быть помещён перечень принятых терминов с соответствующими разъяснениями.

3.7 Библиографический список

Ссылки на источник, указанный в списке литературы, оформляется в виде квадратных скобок с указанием номера книги, например: [1, 2, 5]. Также можно ссылаться на источники Интернет. На все источники, указанные в списке литературы, обязательно должны быть ссылки в тексте работы. Например:

Книга одного, двух или трёх авторов

1. Шеферд, Дж. Программирование на Microsoft Visual Studio C++.NET / Дж. Шефферд. — М. : Издательско-торговый дом «Русская редакция», 2003. — 928 с.
2. Блинова, Т. А. Компьютерная графика / Т. А. Блинова, В. Н. Порев. — К. : Издательство Юниор, 2005. — 520 с.
3. Якобсон, А. Унифицированный процесс разработки программного обеспечения / А. Якобсон, Г. Буч, Дж. Рамбо. — СПб.: Питер, 2002. – 496 с.

Книга четырёх и более авторов

1. Программирование на *Microsoft Visual C++* для профессионалов / Д. Д. Круглински [и др.] Пер. с англ. — СПб. : Питер, 2002. — 864 с.: ил.

Источники Интернет

Здесь обязательно надо указывать название сайта и название статьи, на которую производится ссылка:

1. Перспективные технологии и новые разработки. Информация о технологии.

http://www.sibpatent.ru/default.asp?khid=27332&code=063551&sort=2.

2. Иванов, Ф. Ф, Петров, В. В., Сидоров, Т. Т., Соловье-ва, А. А. Фрагмент онтологии физической химии и его модель // Электронный журнал «Исследовано в России», 3, с. 10—14, 1998. http://zhumal.ape.relam.ru/articles/ 1998/003.pdf.

4 Пример выполнения курсовой работы

В качестве примера выполнения курсовой работы рассмотрим проектирование системы автоматизированной продажи билетов на междугородние автобусные рейсы.

Оглавление работы выглядит следующим образом:

ОГЛАВЛЕНИЕ

4.1 Введение

В процессе разработки любого программного продукта важное место занимает этап его проектирования. В последнее время широкое распространение получили визуальные модели, представляющие собой средства для описания, проектирования и документирования системы.

Целью курсовой работы является проектирование системы, предназначенной для автоматизации процесса продажи билетов на междугородние автобусные рейсы. Для выполнения работы выбрано *CASE*-средство моделирования *IBM Rational Rose*, основанное на применении языка *UML*.

Во введении следует кратко описать теоретические вопросы, связанные с проектированием**.**

4.2. Постановка задачи

В рамках поставленной цели необходимо выполнить следующие задачи:

– описать и смоделировать предметную область, применяя диаграмму вариантов использования;

– подробно проанализировать модель системы на примере одного из вариантов использования (создать классы и рассмотреть диаграммы взаимодействия);

– спроектировать архитектуру системы;

– спроектировать физическую реализацию системы на основе диаграмм размещения и компонентов;

– создать схему базы данных.

4.3 Анализ требований

После описания предметной области и постановки задачи, вы-являются основные объекты, фигурирующие в системе. На основе этих объектов строится глоссарий предметной области, представ-ленный в таблице 1.

Таблица 1 — Глоссарий предметной области

Термин	Обозначение в проекте	Значение
Каталог автобусных рейсов	*BusLineCatalog*	Перечень автобусных рейсов, осуществляющих перевозку пассажиров
Льготник	*PrivelegesPassanger*	Пассажир, имеющий право на льготы
Пассажир	*Passenger*	Лицо, приобретающее билет
Система расчета	*CalculationSystem*	Компьютерная программа, рассчитывающая стоимость проезда
Система регистрации	*RegistrationSystem*	Компьютерная программа, принимающая данные от пассажира и обрабатывающая их

Далее изучаются функциональных требований к системе и строится диаграмма вариантов использования. На этой диаграмме присутствуют действующие лица, варианты использования и связи между ними.

Действующее лицо (*actor)* — это роль, которую пользователь играет по отношению к системе. Действующие лица могут представлять собой роли людей либо некоторых внешних систем (см. таблицу 2).

Таблица 2 — Действующие лица

Действующее лицо	Наименование в проекте	Описание
Пассажир	*Passenger*	Лицо, приобретающее билет
Система расчета	*CalculationSystem*	Компьютерная программа, рассчитывающая стоимость проезда
Система регистрации	*RegistrationSystem*	Компьютерная программа, принимающая данные от пассажира и обрабатывающая их

Разрабатываемая система предполагает наличие следующих вариантов использования:

– *OrderTicket* — заказ билета, либо заявка на аннулирование заказа;

– *InformationForCalculation* — передача информации для расчета стоимости билета в систему расчета;

– *CalculationPrice* — передача информации о стоимости биле-та пассажиру;

– *ConfirmOrder* — подтверждение заказа;

– *AddAndPrintTicket* — добавление информации о заказе в базу данных и печать билета;

– *RemoveTicket* — аннулирование заказа.

Диаграмма вариантов использования представлена на рисунке 1.

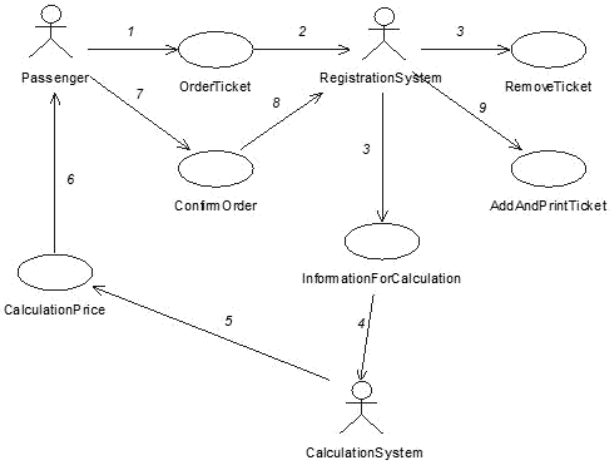

Рисунок 1 — Диаграмма вариантов использования

В курсовой работе будет анализироваться вариант использования *OrderTicket.*

Данный вариант использования начинает выполняться, когда пассажир хочет заказать билет или аннулировать свой заказ. Система запрашивает требуемое действие, затем выполняется один из основных потоков событий: заказ билета или аннулирование заявки.

Параллельно с проектированием к вариантам использования необходимо добавить описания, включающие краткое описание варианта использования, основной поток событий, альтернативные потоки, предусловия и постусловия. Все эти данные записываются в файл *MS Word* или текстовый файл и прикрепляются к соответствующему варианту использования. В дальнейшем эти сведения будут включаться в документацию системы.

В качестве примера рассмотрим **Вариант использования *OrderTicket*** и его описание в файле *MS Word.*

Краткое описание

Вариант использования позволяет пассажиру заказать билет на интересующий его рейс. Также пассажир может аннулировать свой заказ.

Основной поток событий

Вариант использования начинает выполняться, когда пассажир хочет заказать билет или аннулировать свой заказ.

1. Система запрашивает требуемое действие (заказать билет или аннулировать заказ).

2. После того, как пассажир указывает действие, выполняет-ся один из подчинённых потоков («Заказ билета» или «Аннулиро-вание заявки»).

Заказ билета

1. Пассажир указывает дату поездки, номер рейса, пункт отправления, пункт прибытия, свои данные.

2. Пассажир выбирает посадочное место из числа доступных мест.

3. Заявка отправляется в систему регистрации.

Аннулирование заявки

1. Пассажир указывает номер билета.

2. Заявка на аннулирование заказа отправляется в систему регистрации.

Альтернативные потоки

1. Нет свободных мест на выбранном рейсе.

Если на выбранном пассажиром рейсе нет свободных мест, система сообщает ему об этом и предлагает выбрать другой рейс.

2. Допущена ошибка при вводе данных.

Если допущена ошибка при вводе данных пассажиром либо не заполнены обязательные поля, система сообщает об этом и просит исправить ошибку.

Предусловия

Отсутствуют.

Постусловия

Если вариант использования завершится успешно, заявка будет отправлена в систему регистрации.

4.4 Анализ системы

В процессе анализа варианта использования *OrderTicket* разрабатываются: диаграммы классов, реализующие вариант использования, и диаграммы взаимодействия, отражающие взаимодействие объектов в процессе реализации сценариев варианта использования. Все эти диаграммы помещаются в кооперацию с именем рассматриваемого варианта использования. Кооперация представляет собой описание совокупности объектов, реализующих некоторое поведение. Связь между вариантом использования и его реализацией изображается на специальной диаграмме трассировки (рисунок 2).

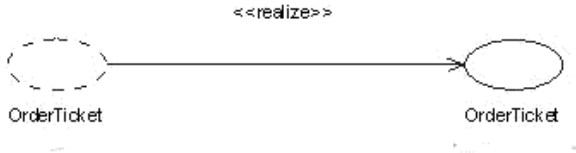

Рисунок 2 — Реализация варианта использования *OrderTicket*

Дальнейший шаг — создание классов анализа.

В потоках событий варианта использования выявляются клас-сы трех типов:

– граничные классы (*Boundary*) — служат посредниками при взаимодействии внешних объектов с системой;

– классы-сущности (*Entity*) — представляют собой ключевые абстракции (понятия) разрабатываемой системы;

– управляющие классы (*Control*) — обеспечивают координацию поведения объектов в системе.

Классы, участвующие в варианте использования *OrderTicket*, представлены в таблице 3.

Таблица 3 — Классы, участвующие в варианте использования *OrderTicket*

Название класса	Описание	Тип
Passenger	Лицо, приобретающее билет.	*Entity*
PrivelegesPassanger	Пассажир, имеющий право на льготы.	*Entity*
BusLineCatalog	Перечень автобусных рейсов, осуществляющих перевозку пассажиров.	*Entity*
RegistrationSystem	Компьютерная программа, принимающая данные от пассажира и обрабатывающая их.	*Entity*
OrderTicketcsForm	Интерфейс заказа билет.а	*Boundary*
BusLineCatalogSystem	Системный интерфейс взаимодействия системы регистрации с каталогом автобусных рейсов.	*Boundary*
OrderController	Проверка корректности данных и работы системы.	*Control*

Диаграмма классов анализа представлена на рисунке 3.

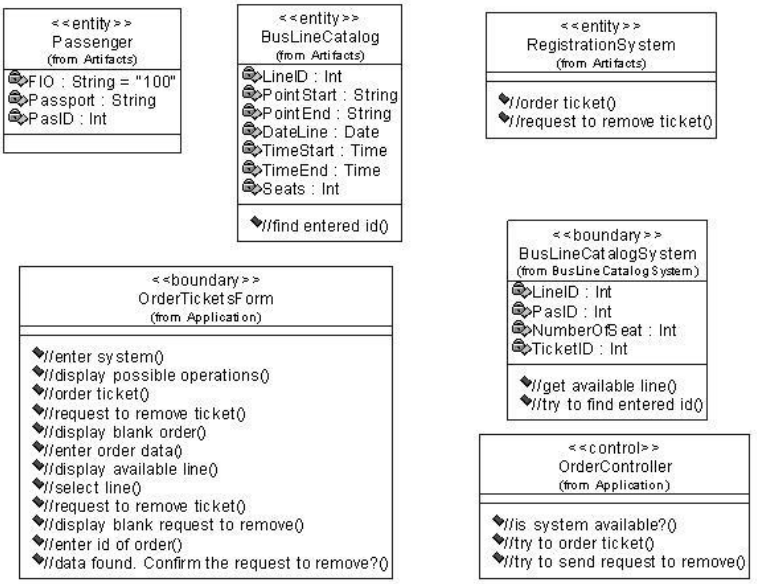

Рисунок 3 — Диаграмма классов

Диаграмма классов содержит некоторые атрибуты и методы классов. Они возникают в процессе разработки диаграмм взаимо-действия.

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов. На таких диаграммах отображается ряд объектов и те сообщения, которыми они обмениваются между собой. Существует два вида диаграмм взаимодействия: диаграм-мы последовательности и кооперативные диаграммы.

Диаграммы последовательности отражают поток событий (упорядоченных по времени), происходящих в рамках варианта использования. Все объекты, требуемые системе для выполнения варианта использования, представлены в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между объектами. От объектов вниз идут пунктирные линии, называемые линиями жизни.

Кооперативные диаграммы, в отличие от диаграммы последовательности, акцентируют внимание на связях между объектами.

Из кооперативной диаграммы легче понять связи между объектами, но труднее уяснить последовательность событий.

Диаграммы взаимодействия в рамках варианта использования *OrderTicket* представлены на рисунках 4—9:

− рисунок 4 — Диаграмма последовательности, описывающая основной поток событий *OrderTicket-BasicFlow*;

− рисунок 5 — Диаграмма последовательности, описывающая поток заказа билета *OrderTicket-OrderTicket;*

− рисунок 6 — Диаграмма последовательности, описывающая поток заявки на аннулирование заказа *OrderTicket-RequestToRemoveTicket;*

− рисунок 7 — Кооперативная диаграмма, описывающая основной поток событий *OrderTicket-BasicFlow;*

− рисунок 8 — Кооперативная диаграмма, описывающая поток заказа билета *OrderTicket-OrderTicket;*

− рисунок 9 — Кооперативная диаграмма, описываю-щая поток заявки на аннулирование заказа *OrderTicket-RequestToRemoveTicket.*

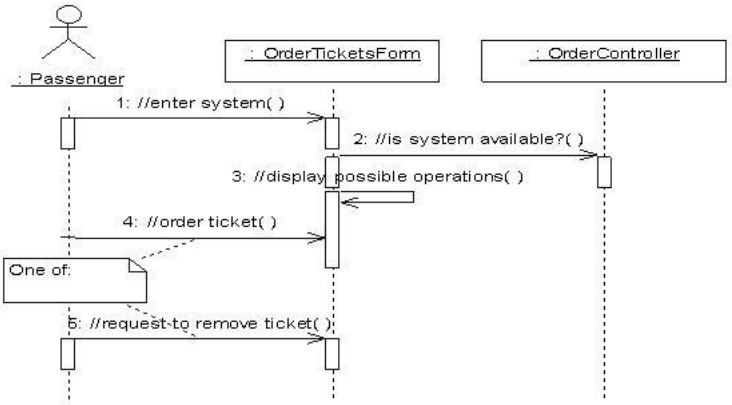

Рисунок 4 — Диаграмма последовательности *OrderTicket-BasicFlow*

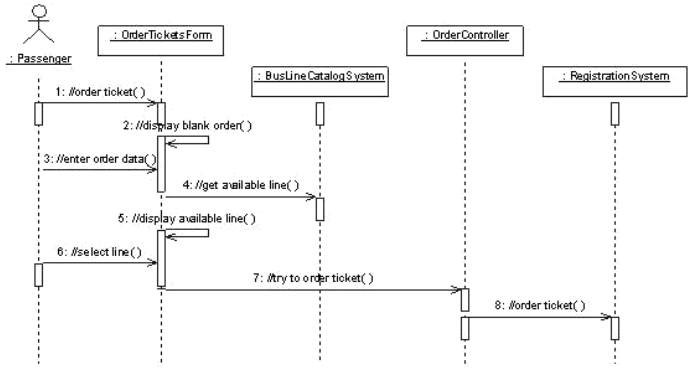

Рисунок 5 — Диаграмма последовательности *OrderTicket-OrderTicket*

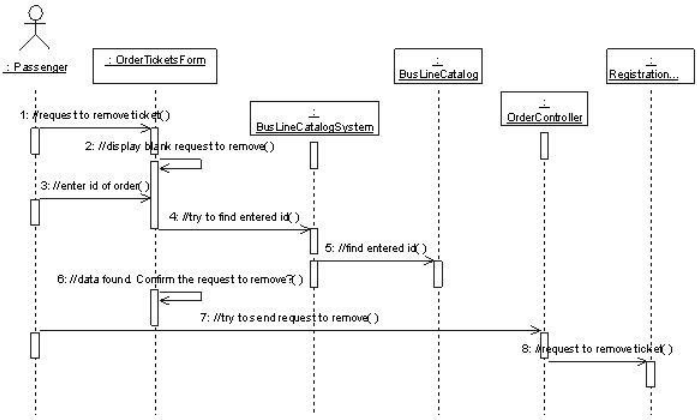

Рисунок 6 — Диаграмма последовательности *OrderTicket-RequestToRemoveTicket*

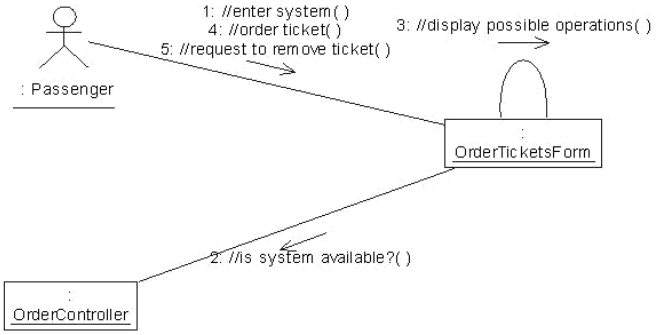

Рисунок 7 — Кооперативная диаграмма *OrderTicket-BasicFlow*

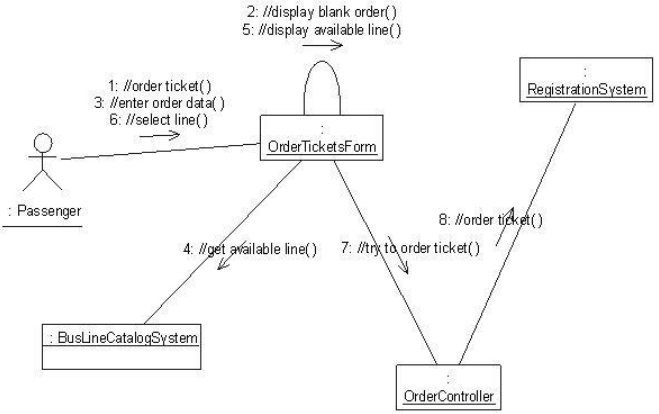

Рисунок 8 — Кооперативная диаграмма *OrderTicket-OrderTicket*

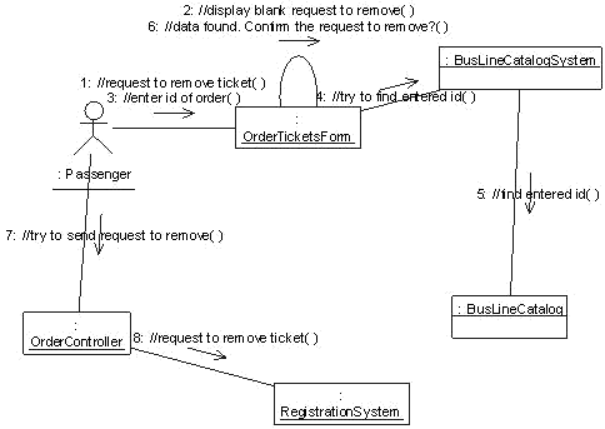

Рисунок 9 — Кооперативная диаграмма *OrderTicket-RequestToRemoveTicket*

4.5 Проектирование

Для облегчения понимания архитектуры системы целесообразно в процессе ее проектирования объединять классы, имеющие сходное логическое значение, в пакеты.

В рамках данной работы выделено два пакета, представляющих собой архитектурные уровни системы:

– *Application* — содержит элементы прикладного уровня;

– *Business Services* — содержит элементы, реализующие бизнес-логику системы.

Уровень *Application* содержит в себе классы *OrderTicketsForm* и *OrderController*.

Уровень *Business Services* включает подсистему *BusLineCatalogSystem*, содержащую одноименный класс, и па-кет *Artefacts*, содержащий классы-сущности *BusLineCatalog, Passenger, PrivelegesPassanger* и *RegistrationSystem.*

Отображение логики представления модели в браузере показано на рисунке 10**.**

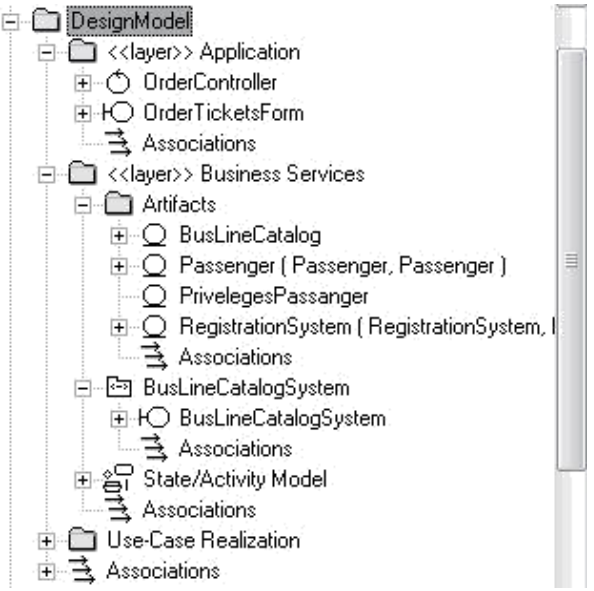

Рисунок 10 — Структура логического представления в браузере

Рассмотрим описание классов, участвующих в варианте использования *OrderTicket* (см. в таблице 3). Описание атрибутов классов и описание операций представлено в таблицах 4—11.

Таблица 4 — Описание атрибутов класса *Passenger*

Имя	Описание	Тип
PasID	Идентификатор пассажира	*Int*
FIO	ФИО пассажира	*String*
Passport	Паспортные данные пассажира	*String*

Таблица 5 — Описание атрибутов класса *BusLineCatalog*

Имя	Описание	Тип
LineID	Идентификатор рейса	*Int*
PointStart	Пункт отправления	*String*
PointEnd	Пункт прибытия	*String*
DateLine	Дата	*Date*
TimeStart	Время отправления	*Time*
TimeEnd	Время прибытия	*Time*
Seats	Количество мест	*Int*

Таблица 6 — Описание атрибутов класса *BusLineCatalogSystem*

Имя	Описание	Тип
LineID	Идентификатор рейса	Int
PasID	Идентификатор пассажира	Int
NumberOfSeat	Номер места	Int
TicketID	Идентификатор заказа	Int

Таблица 7 — Описание операций класса *RegistrationSystem*

Имя	Описание
order ticket	Заказ билета (передача заказа для дальнейшей обработки)
request to remove ticket	Удаление заказа (передача заявки на удаление для дальнейшей обработки)

Таблица 8 — Описание операций класса *BusLineCatalog*

Имя	Описание
find entered id	Поиск введенного идентификатора билета

Таблица 9 — Описание операций класса *BusLineCatalogSystem*

Имя	Описание
get available line	Просмотр доступных рейсов
try to find entered id	Запрос на поиск введенного идентификатора билета при удалении заказа

Таблица 10 — Описание операций класса *OrderController*

Имя	Описание
is system available?	Проверка доступности системы
try to order ticket	Запрос на заказ билета
try to send request to remove	Запрос на заявку на удаление заказа

Таблица 11 — Описание операций класса *OrderTicketsForm*

Имя	Описание
enter system	Вход в систему
display possible operations	Вывод доступных операций
order ticket	Заказ билета
request to remove ticket	Запрос на удаление заказа
display blank order	Вывод формы для заполнения заказа
enter order data	Ввод данных для заказа

Имя	Описание
display available line	Вывод доступных рейсов
select line	Выбор рейса
display blank request to remove	Вывод формы для заполнения заявки на удаление заказа
enter id of order	Ввод идентификатора билета
data found. Confirm the request to remove	Данные найдены, подтверждение заявки на удаление заказа

Диаграмма классов анализа с указанием атрибутов и операций показана на рисунке 3.

Полная диаграмма классов системы со связями представлена на рисунке 11. На этой диаграмме задействованы различные виды связей, которые влияют на получаемый при генерации программ-ный код. В диаграммах классов используются следующие виды связей:

– ассоциация (*association*) — показывает, что объекты одной сущности (класса) связаны с объектами другой сущности;

– зависимость (*dependence)* — один класс использует объ-екты другого;

– ассоциированный класс (*association class*) — предназна-чен для задания дополнительных атрибутов для связи;

– наследование (*generalization*) — позволяет указать наследование классов, один класс является родительским по отношению к другому;

– реализация (*realization*) — один класс создан на основе шаблона другого класса.

Специальными видами ассоциаций являются агрегирование (*aggregation*) и композиция (*compozition*). На рисунке 11 примене-ны ассоциации, наследование и агрегация.

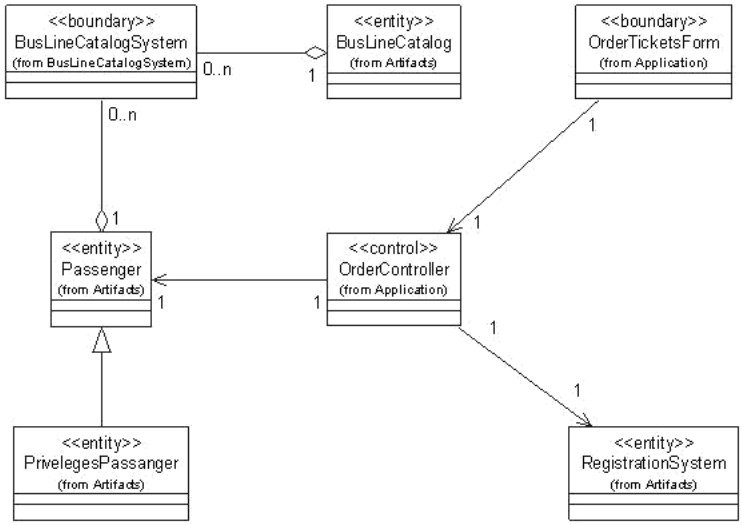

Рисунок 11 — Полная диаграмма классов

Для определения возможных состояний классов в системе используются диаграммы состояний. Класс *RegistrationSystem* обладает сложным поведением, поэтому для него целесообразно разработать диаграмму состояний, представленную на рисунке 12.

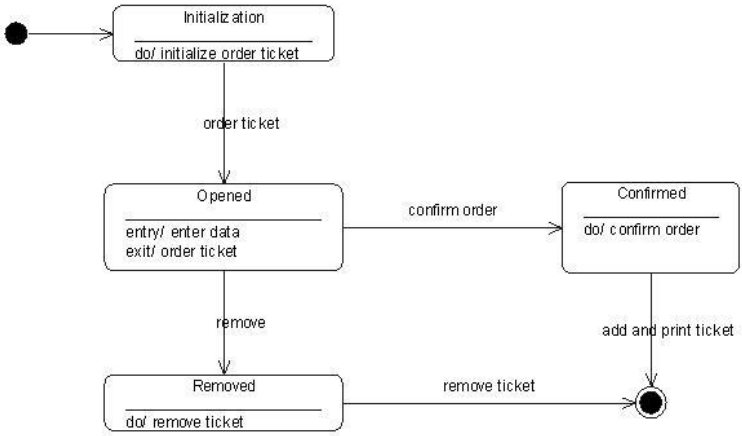

Рисунок 12 — Диаграмма состояний для класса *RegistrationSystem*

Основными изобразительными элементами на диаграмме состояний являются: начальная точка, состояние (прямоугольник с закруглёнными углами), переходы между состояниями (сплошная линия со стрелкой) и конечная точка. Состояние содержит переменные состояния и различные виды деятельности.

4.6 Реализация

В этой главе рассматривается моделирование физического уровня системы (диаграмма компонентов и диаграмма размещения), а также создание схемы базы данных.

Диаграмма компонентов служит для представления компонен-тов программного обеспечения и связей между ними (рисунок 13). Выделяются два вида компонентов: исполняемые компоненты и библиотеки кода. Такие диаграммы применяются для генерации «скелетного» кода модели.

Так, если система разрабатывается на языке *C++*, то класс *Passenger* преобразуется в компоненты *Passenger:* тело (выделен тёмным цветом) и заголовок класса.

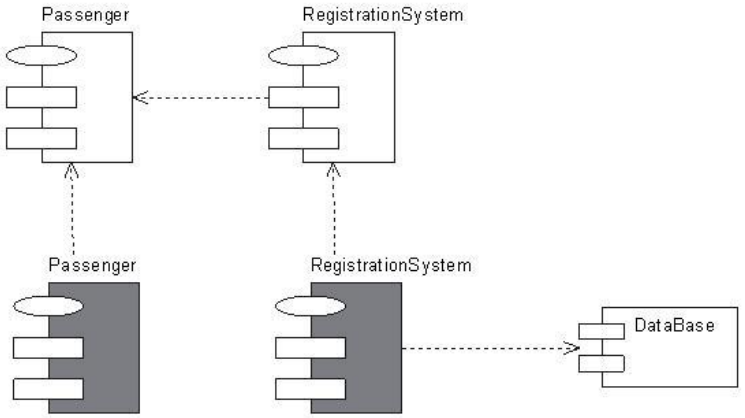

Рисунок 13 — Диаграмма компонентов системы

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы, показывает физическое расположение сети и различных компонентов в ней (рисунок 14). На диаграмме выделены её основные элементы: узлы (*node*), представляющие собой вычислительный

ресурс, и соединения (*connection*) — каналы взаимодействия узлов.

Элементы диаграммы размещения:

1. *Terminal* — устройство для взаимодействия пассажира с системой;
2. *RegistrationServer* — устройство для обработки данных;
3. *RegistrationSystem* — система регистрации данных для заказа билета;
4. *CalculationSystem* — система расчета стоимости билета;
5. *Printer* — устройство печати билетов;
6. *DataBase* — база данных.

База данных является неотъемлемой частью практически любой информационной системы. *IBM Rational Rose* предоставляет возможность автоматической генерации кода при создании базы данных.

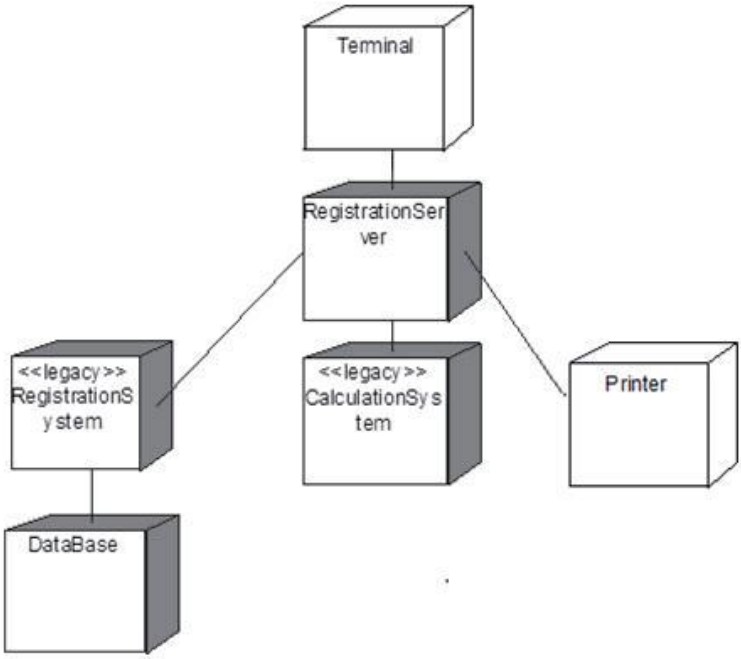

Рисунок 14 — Диаграмма размещения

Диаграмма «сущность-связь», необходимая для генерации кода базы данных, представлена на рисунке 15. В качестве сущностей выступают классы-сущности (*entity classes*), которые, как правило, содержат хранимую информацию. В дальнейшем, для каждого такого класса создают таблицу в базе данных.

Сгенерированный код схемы базы данных приведён в Приложении.

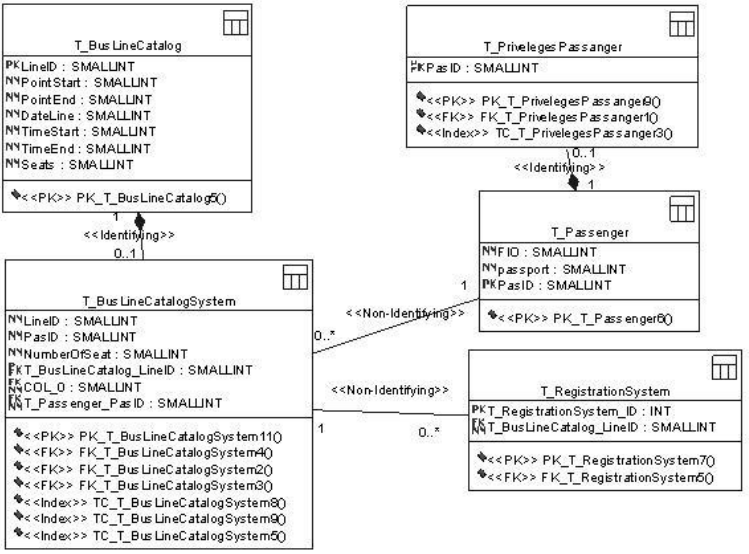

Рисунок 15 — Диаграмма «сущность-связь»

4.7 Заключение

Целью данной курсовой работы являлось проектирование системы для автоматизации процесса продажи билетов на междугородние автобусные рейсы. В ходе выполнения данной работы решены сформулированные задачи, а именно:

− смоделирована и описана предметная область, разработана диаграмма вариантов использования;

− подробно проанализирована модель системы на стадии заказа билета пассажиром;

− сгенерирован код формирования базы данных.

Для проведения анализа модели системы на стадии заказа билета были созданы диаграммы следующих видов:

− диаграмма классов анализа;

− диаграммы взаимодействия (диаграммы последовательности и кооперативные диаграммы);

− полная диаграмма классов системы;

− диаграмма состояний;

− диаграмма компонентов;

− диаграмма размещения;

− диаграмма «сущность-связь» для создания базы данных.

Полученные результаты позволяют считать цель выполнения данной работы достигнутой.

После заключения приводятся библиографический список и Приложение.

4.8 Приложение

В Приложении размещён полученный с помощью *IBM Rational Rose* код создания базы данных, фрагмент которого показан ниже.

КОД СОЗДАНИЯ БАЗЫ ДАННЫХ

```
……
SET QUOTED_IDENTIFIER
ON GO
CREATE TABLE “S_1”.“T_PrivelegesPassanger”
    ( “PasID” SMALLINT NOT NULL,
    CONSTRAINT “PK_T_PrivelegesPassanger9” PRIMARY
KEY NONCLUSTERED (“PasID”)
    )
GO
CREATE INDEX “TC_T_PrivelegesPassanger3” ON “S_1”.“T_
PrivelegesPassanger” (“PasID”)
GO
CREATE TABLE “S_1”.“T_Passenger” (
    “FIO” SMALLINT NOT NULL,
    “passport” SMALLINT NOT NULL,
    “PasID” SMALLINT NOT NULL,
    CONSTRAINT “PK_T_Passenger6” PRIMARY KEY
NONCLUSTERED (“PasID”)
    )
GO
CREATE TABLE “S_1”.“T_BusLineCatalogSystem”
    ( “LineID” SMALLINT NOT NULL,
    “PasID” SMALLINT NOT NULL,
```

```
        “NumberOfSeat” SMALLINT NOT NULL,
        “T_BusLineCatalog_LineID” SMALLINT NOT NULL,
        “COL_0” SMALLINT NOT NULL,
        “T_Passenger_PasID” SMALLINT NOT NULL,
        CONSTRAINT “PK_T_BusLineCatalogSystem11”
PRIMARY KEY NONCLUSTERED (“T_BusLineCatalog_LineID”)
        )
    GO
    CREATE INDEX “TC_T_BusLineCatalogSystem8” ON
“S_1”.“T_BusLineCatalogSystem” (“T_Passenger_PasID”)
    GO
    CREATE INDEX “TC_T_BusLineCatalogSystem9” ON
“S_1”.“T_BusLineCatalogSystem” (“COL_0”)
    GO
    CREATE INDEX “TC_T_BusLineCatalogSystem5” ON
“S_1”.“T_BusLineCatalogSystem” (“T_BusLineCatalog_LineID”)
    GO
    CREATE TABLE “S_1”.“T_RegistrationSystem” (
        “T_RegistrationSystem_ID” INT IDENTITY NOT NULL,
        “T_BusLineCatalog_LineID” SMALLINT NOT NULL,
        CONSTRAINT “PK_T_RegistrationSystem7” PRIMARY
KEY NONCLUSTERED (“T_RegistrationSystem_ID”)
        )
    GO
    CREATE TABLE “S_1”.“T_BusLineCatalog”
        ( “LineID” SMALLINT NOT NULL,
        “PointStart” SMALLINT NOT NULL,
        “PointEnd” SMALLINT NOT NULL,
        “DateLine” SMALLINT NOT NULL,
        “TimeStart” SMALLINT NOT NULL,
        “TimeEnd” SMALLINT NOT NULL,
        “Seats” SMALLINT NOT NULL,
        CONSTRAINT “PK_T_BusLineCatalog5” PRIMARY KEY
NONCLUSTERED (“LineID”)
        )
    GO
    ALTER TABLE “S_1”.“T_RegistrationSystem” ADD
CONSTRAINT “FK_T_RegistrationSystem5” FOREIGN KEY
(“T_BusLineCatalog_LineID”) REFERENCES “S_1”.“T_
BusLineCatalogSystem” (“T_BusLineCatalog_LineID”)
    И т. д.
```

Подобный код описания таблиц БД пригоден для его ввода в соответствующую СУБД (в данном случае, *SQL Server*) и получе-ния первой версии работоспособной базы данных.

5 Требования к презентации

Основная цель презентации курсовой работы состоит в том, чтобы научиться кратко и наглядно представлять результаты сво-ей работы [3]. Поскольку студенты довольно редко рассказывают о своих приложениях и программах, то именно подготовка презентации и выступление перед аудиторией для многих студентов являются непривычной и трудной задачей. Обучение представлению своей работы, умению отстаивать свои взгляды во время дис-куссии также входит в круг вопросов, решаемых при подготовке и защите курсовой работы.

Поскольку эти вопросы важны для итоговой отметки и в це-лом для развития студентов, остановимся на них более подробно.

Первый шаг — это определение главной идеи, вокруг которой и будет строиться презентация. Главная идея должна быть связана с объектом исследования в курсовой работе. Затем разрабатываются текст и графические блоки. Необходимо, чтобы текстовые блоки чередовались с интересными изображениями или наглядными примерами. Хорошо подготовленные наглядные материалы помогают как выступающему, так и слушателям.

Рекомендуется, чтобы текстовые кадры состояли не более чем из шести строк, написанных большими буквами с использовани-ем пустого пространства между ними. Графические кадры обыч-но представляют собой изображения компьютерной графики, диаграммы, рисунки, схемы. Графические материалы должны быть упрощенной версией аналогичных материалов в тексте курсовой работы. Можно добавить анимационные элементы — они сразу «оживляют» презентацию.

При использовании наглядных материалов существует несколько полезных советов:

– удостоверьтесь, что все видят демонстрируемые материалы;

– дайте аудитории время, чтобы прочитать и понять смысл прежде, чем им предстоит продолжить слушать;

– один кадр должен выражать одну идею;

– иллюстрируйте только главные пункты, а не всю работу;

– не демонстрируйте то, что может конфликтовать с тем, о чём вы рассказываете;

– не читайте то, что написано в кадре, вместо этого перефразируйте и дополните;

– наглядные материалы — не главное, они лишь дополняют слова докладчика, а не наоборот.

Когда подготовлены презентация и доклад, необходимо перейти к тренировке речи. Желательно рассказывать о своей работе, не читая, для этого нужно несколько раз потренироваться дома.

Примерный список слайдов, который целесообразно подготовить для презентации курсовой работы может выглядеть следующим образом:

1. Слайд, содержащий название работы, ФИО автора и руко-водителя.
2. Цель работы, выполненные задачи и используемые программные средства.
3. Глоссарий предметной области.
4. Диаграмма вариантов использования.
5. Представление разрабатываемого варианта использования.
6. Классы системы.
7. Диаграммы последовательностей.
8. Кооперативные диаграммы.
9. Полная диаграмма классов.
10. Диаграмма компонентов.
11. Диаграмма размещения.
12. Заключение
13. И др.

6 Темы курсовых работ

Как было сказано выше, желательно каждому студенты предложить и согласовать с руководителем свою тему, у которой будет заказчик. В этом случае все решения придётся координировать с заказчиком — работа начинает приобретать более глубокий смысл.

В качестве примеров таких работ, выполненных студентами ранее, можно привести следующие:

– Проектирование и реализации приёма и передачи телемеханической информации для энергетических систем в рамках про-токола РПТ ЭВМ.

– Проектирование и реализация системы температурного контроля протонной установки, предназначенной для медицинских целей.

– Проектирование и анализ системы контроля исполнения работ в геоинформационной системе «Южные сети».

– Проектирование и реализация системы учёта отходов и гранул производственной фирмы «*Real Plast*».

– Проект автоматизированной биллинговой системы (с проработкой функции пополнения лицевого счёта) в Протвинской локальной сети «*Protvino Network*».

– Анализ конфигурационных баз данных *SCADA* СИСТЕЛ.

– Моделирование цикла работы намоточного станка.

– И др.

Для тех, кто не смог подобрать работу, связанную с заказчиком, существует список тем курсовых работ:

1. Проектирование работы отдела кадров с проработкой функции увольнения и с проработкой функции приёма на работу (работа для двух человек).
2. Проектирование работы компьютерного центра.
3. Проектирование системы бронирования билетов в туристической фирме.
4. Проектирование системы работы учебных курсов.
5. Проектирование системы заключения договоров.
6. Проект автоматизации работы бухгалтерии предприятия.
7. Проектирование фирмы по продаже квартир.
8. Проектирование системы взаимодействия налоговой инспекции с юридическими и физическими лицами.
9. Проектирование системы начисления заработной платы.
10. Моделирование предприятия, занимающегося торгово-закупочной деятельностью.
11. Моделирование фирмы, занимающейся строительной дея-тельностью.
12. Проектирование системы приёма в детской больнице (ведение карточки истории болезни).
13. Разработка системы инвентаризационного учёта.
14. Проектирование системы регистрации клиентов на трени-ровки для фитнес-клуба.
15. Проектирование системы тестирования знаний студентов.
16. Проектирование системы для кадрового агентства.
17. Автоматизация работы библиотекаря.
18. Проектирование системы сетевого маркетинга.
19. Автоматизация управления расписанием учебных занятий.
20. Система обработки заказов торгово-посреднической компании.
21. Проект системы реализации продукции.

22. Автоматизация получения кредита в банке.
23. Автоматизация работы салона красоты.
24. Проектирование структуры управления компьютерным магазином.
25. Проектирование работы интернет-магазина. И т. д.

Библиографический список

1. Положение о выполнении и защите курсовых работ (проектов) в университете «Дубна» // Приложение к приказу ректора № 2221 от 14.10.2010.

2. Кульман, Т. Н. Подготовка и оформление курсовой работы по дис-циплине «Программированию на языке высокого уровня» / Т. Н. Куль-ман, М. М. Губаева, М. П. Астафьева. — Дубна : Международный уни-верситет природы, общества и человека «Дубна», 2009. — 34 с.

3. Кульман, Н. Ю. Подготовка курсовых работ по дисциплине «Ком-пьютерная графика» / Н. Ю. Кульман, Т. Н. Кульман. — Дубна : Меж-дународный университет природы, общества и человека «Дубна», 2011. — 47 с.

4. Вендров, А. М. Проектирование программного обеспечения эко-номических информационных систем: Учебник. — 2-е изд., перераб. и доп. / А. М. Вендров. — М.: Финансы и статистика, 2005. — 554 с.

5. Вендров, А. М. Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. посо-бие. — 2-е изд., перераб. и доп. / А. М. Вендров — М.: Финансы и стати-стика, 2006. — 192 с.

6. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд.: Пер. с англ. / Г. Буч — М.: Из-дательство Бином, СПб.: Невский диалект, 1998. — 560 с.

7. Боггс, У. UML и Rational Rose: Пер. с англ. / У. Боггс, М. Боггс — М.: Лори, 2001. — 582 с.

8. Буч, Г. Язык UML. Руководство пользователя: Пер. с англ. / Г. Буч, Дж. Рамбо, А. Джекобсон — М.: ДМК, 2000. — 432 с.

9. Шмуллер, Д. Освой самостоятельно UML за 24 часа: Пер. с англ. / Д. Шмуллер. — М.: Издательский дом «Вильямс», 2002. — 352 с.

10. Кватрани, Т. Визуальное моделирование с помощью Rational Rose 2002 и UML: Пер. с англ. / Т. Кватрани. — М.: Вильямс, 2003 — 192 с.

11. Фаулер, М. UML в кратком изложении: Пер. с англ. / М. Фаулер, К. Скотт. — М.: Символ-Плюс, 2002. — 192 с.

12. Трофимов, С. А. CASE-технологии: практическая работа в Rati-onal Rose. Изд. 2-е. Пер. с англ. / С. А. Трофимов. — М.: Бином-Пресс, 2002. — 288 с.

13. Калянов, Г. Н. CASE-технологии. Консалтинг в автоматизации биз-нес-процессов. — 3-е изд. — М.: Горячая линия-Телеком, 2002. — 320 с.

Приложение

Приложение А Образец титульного листа курсовой работы

Государственное бюджетное образовательное учреждение
высшего образования Московской области
«Университет «Дубна»

Филиал «Протвино»
Кафедра «Информационные технологии»
(наименование кафедры)

КУРСОВАЯ РАБОТА ПО

_____
(наименование учебной дисциплины)

ТЕМА: _____
(наименование темы)

Выполнил: студент
_____ группы
_____ курса

_____
(Ф.И.О.)
Руководитель:

_____
(ученая степень, ученое звание, занимаемая должность)

Дата защиты: _____
Оценка: _____

_____
(подпись руководителя)

Содержание

ДЛЯ ЗАМЕТОК

Электронное учебное издание

Кульман Татьяна Николаевна

ПОДГОТОВКА И ОФОРМЛЕНИЕ КУРСОВЫХ РАБОТ ПО ДИСЦИПЛИНЕ «ТЕОРИЯ И ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ»

ЭЛЕКТРОННОЕ МЕТОДИЧЕСКОЕ ПОСОБИЕ

Филиал «Протвино»
государственного университета «Дубна
142281 г. Протвино Московской обл.,
Северный проезд, д. 9