Министерство Образования Республики Беларусь

Учреждение образования

«Полесский государственный университет»

Факультет банковского дела

Кафедра высшей математики и информационных технологий

Дисциплина: СТАРИС

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ИНЖИНИРИНГОВОГО ПРЕДПРИЯТИЯ С ПОМОЩЬЮ CASE-СРЕДСТВ**

Студент: гр. 14ИТ-1 Домась А. Н.\_\_\_\_\_\_\_\_\_

Руководители: Штепа В.Н., к.т.н., доцент

Разинков А.И., ассистент

Пинск 2016

# Реферат

Пояснительная записка к курсовому проекту: 43 страницы, 21 рисунок, 8 источников.

Объектом исследования являются программный продукт фильтрации данных инжинирингового предприятия с помощью использования запросов в БД.

Предметом исследования является инжиниринговое предприятие.

Целью данного курсового проекта является создание системы обработки запросов с помощью CASE – средств.

Задача исследования предопределяются целью исследования и состоят в том, чтобы: создать диаграммы UML и кооперативные диаграммы при помощи систем моделирования IBM Rational Rose Enterprise Edition.

Методы исследования: при реализации рассматриваемой методики использовались отраслевые и международные стандарты, CASE –технологии, современные инструментальные среды и пакеты моделирования, так же использовался метод системного анализа проекта.

Разработанная информационная система может использоваться для последующего построения и создания приложения для инжинирингового предприятия и последующего его внедрения. Что упростит работу сотрудников и повысит скорость обработки запросов.

**Содержание**

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc463797631)

[2 ОПИСАНИЕ ПОТОКОВ СОБЫТИЙ 7](#_Toc463797632)

[3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ С ПОМОЩЬЮ МЕТОДА ПРОЕКТИРОВАНИЯ «COMET» 12](#_Toc463797633)

[4 ОПИСАНИЕ ДИАГРАММ 19](#_Toc463797634)

[5 ОПИСАНИЕ ГЕНЕРАЦИИ КОДА 34](#_Toc463797635)

[ЗАКЛЮЧЕНИЕ 35](#_Toc463797636)

# Введение

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем, создаваемых в различных областях. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями:

− сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов;

− наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема);

− необходимость интеграции существующих и вновь разрабатываемых приложений;

− функционирование в неоднородной среде на нескольких аппаратных платформах;

Системы стали настолько велики, что физических и умственных возможностей и способностей человека уже просто недостаточно для того, чтобы спроектировать сложную систему за один шаг, и даже для того, чтобы просто представить, вообразить все возможности и потребности проектируемой системы, ее архитектуру и программное обеспечение. Однако замечено, что визуальная информация воспринимается наиболее успешно и полно [1].

Цель: создание модели системы «дизайнерской фирмы» на основе использования языка UML. Для построения информационной системы в языке UML, будем использовать семь видов диаграмм этого языка. Все диаграммы данной курсовой работы разработаны с помощью системы моделирования IBM Rational Rose Enterprise Edition.

Задачи курсового проекта:

1. Анализ предметной области курсового проекта.
2. Построение семи диаграмм на языке UML (вариантов использования, последовательности, кооперации, классов, состояний, компонентов, размещения).
3. На основе полученных диаграмм сгенерировать код.
4. **Аналитическая часть**

**1.1 Описание предметной области**

Предметной областью называется часть реальной системы, представляющая интерес для данного исследования. При проектировании автоматизированных информационных систем предметная область отображается моделями данных нескольких уровней. Число уровней зависит от сложности решаемых задач, но в любом случае включает концептуальный и логический уровни.

В данном курсовом проекте предметной областью является работа инжинирингового предприятия, которая занимается выполнением заказов клиентов. У каждого работника имеются свои обязанности, например: «Директор» занимается авторизацией работников фирмы, а так же работает с БД, в обязанности «Проект-менеджера» входят: оформление заказа, печать и выдача чека, уведомление «Инженеров», которые в свою очередь передают выполненный заказ «Ведущему инженеру», который представляет заказ «Заказчику».

# 1.2 Анализ проблем предметной области

После проведения исследования предметной области были выявлены следующие проблемы:

− с увеличением количества клиентов, увеличивается количество информационных потоков, что приводит к снижению скорости обслуживания клиентов из-за несвоевременного получения информации, ошибочной информации или вовсе из-за отсутствия таковой. Это является недопустимым, так как может затянуть процесс обслуживания, привести к неверному и (или) несвоевременному выполнению заказа;

− затраты большого количества временных и человеческих ресурсов на обработку документов и необходимость увеличивать эти ресурсы с ростом количества клиентов;

− в связи с большим бумажным документооборотом возрастает нагрузка на персонал, в связи с чем возрастает количество ошибок;

− из-за необходимости получать документы и данные из других подразделений возникают временные задержки, связанные с их доставкой.

## **Вывод по главе 1**

В данной главе представлены и проанализированы предметная область и анализ проблемных участков предметной области. И в ходе всего были разработаны этапы проектирования модели системы обработки запросов на основе использования языка UML:

1. выбрать предметную область и исследовать ее;
2. создать семь диаграмм на языке UML;
3. определиться с расширением, классами и возможностями объектов;
4. сгенерировать код, при помощи написанных диаграмм.

Так же, в ходе изучения предметной области, был выбран язык, на который будет сгенерирован код и выбрана система моделирования IBM Rational Rose Enterprise Edition.

# Описание потоков событий

**2.1 Теоретические сведения**

Одним из требований языка UML является самодостаточность диаграмм для представления информации о моделях проектируемых систем. Однако, диаграммы вариантов использования описывают то, что делает система, без уточнения того, как она это делает.

Для реального описания системы потребуется более специфические данные, которые отражены в потоке событий. Потоки событий уточняют или детализируют последовательность действий, совершаемых системой при выполнении ее вариантов использования, а также описывают логику переходом через варианты использования.

Потоки событий – это текстовые описания пошагового выполнения прецедентов, они понятны не только разработчику, но и стороннему читателю. Их задача – еще больше детализировать описание функциональности системы до того, как разработчики приступят к написанию программного кода, и устранить возможное недопонимание требуемой функциональности, как можно больше сблизить представления разработчика о системе и заказчика.

Потоки событий бывают трех типов: основной, альтернативный и поток ошибок.

Основной (главный) поток описывает наилучший сценарий либо наиболее используемый путь исполнения прецедента.

Альтернативный поток специфицирует отклонения от основного потока, которые не рассматривают как ошибочные.

Поток ошибок рассматривается как отклонение от альтернативного или основного, которое порождает условия формирования ошибки.

Действующее лицо (актер) – это, то что взаимодействует с создаваемой системой. Если варианты использования описывают все, что происходит внутри области действия системы, действующее лица определяют все, что находится вне ее [2].

## **Описание актеров и вариантов использования**

В программе присутствует 5 актёров – «Заказчик», «Директор», «Менеджер», «Инженер», «Ведущий инженер».

Директор, инженер и менеджер могут являться пользователями системы. Для работы с системой пользователю необходимо пройти авторизацию, которая позволяет определить пользователя. У каждого работника фирмы есть свой логин и пароль, поэтому они могут получить доступ к системе. У заказчика нет своего логина и пароля, соответственно и доступа к системе у него нет. Клиент посещает предприятие, чтобы зарегистрировать свой заказ.

Вариант использования «Авторизация» дает возможность пользователю получить доступ к системе. Для этого пользователь вводит логин и пароль.

Вариант использования «Просмотр каталогов» дает возможность покупателю посмотреть списки услуг фирмы.

Вариант использования «Выбор услуги» дает возможность заказчику выбрать нужную ему услугу и перейти к оформлению заказа.

Вариант использования «Оплата услуги» дает возможность клиенту оплатить выбранный заказ.

Вариант использования «Выдача чека» дает возможность проект-менеджеру выдать чек покупателю после оплаты заказа.

Вариант использования «Печать чека» дает возможность проект-менеджеру напечатать чек после оплаты заказа.

Вариант использования «Оформление заказа» дает возможность проект-менеджеру перейти к оформлению выбранного заказа и уведомить инженера о новом заказе.

Вариант использования «Заполнение каталогов» и «Уведомление о изменениях» дает возможность арт-директору выполнять изменения данных каталогов.

Вариант использования «Выполнение заказа» дает возможность инженеру передать готовый проект ведущему инженеру.

Вариант использования «Передача выполненного заказа» дает возможность ведущему инженеру отдать готовый заказ клиенту.

## **Основной и альтернативный потоки событий**

Основной поток событий варианта использования «Авторизация»:

1. Вариант использования начинается, когда пользователь запускает приложение.
2. Для использования приложения пользователю необходимо ввести свой логин и пароль.
3. Пользователь вводит логин и пароль
4. Приложение подтверждает введенный логин и пароль. Если логин или пароль был введен неправильно, то выполняется альтернативный поток событий А1.
5. Вариант использования завершается.

Альтернативный поток событий А1:

1. Приложение информирует пользователя о том, что введенный логин и пароль не верны.
2. Вариант использования завершается.

Основной поток событий варианта использования «Оформление заказа»:

1. Вариант использования начинается, когда Менеджер прошел вариант использования «Авторизация» и на начальной форме выбирает функцию «Оформление заказа».
2. После выбора функции «Оформление заказа», ведущий-инженер, введя инфо о заказе, сохраняет заказ в базе данных.
3. Приложение сохраняет данные о заказе. Эти данные будут использоваться дизайнером.

Альтернативный поток событий варианта использования «Заполнение каталогов»:

1. Вариант использования начинается, когда арт-директор прошел вариант использования «Авторизация» и на начальной форме выбирает функцию «Заполнение каталогов».
2. Арт-директор переходит к форме заполнения заказа и заполняет нужную информацию.
3. Сохранение данных.
4. Вариант использования завершается.

Основной поток событий варианта использования «Уведомление о заказе»:

1. Вариант использования начинается, когда проект-менеджер прошел вариант использования «Авторизация».
2. После того, как проект-менеджер оформил заказ и выбирает функцию «Уведомление о заказе».
3. Вариант использования завершается.

Основной поток событий варианта использования «Оплата заказа»:

1. Вариант использования начинается, когда клиент прошел вариант использования «Просмотр каталогов» и «Оформление заказа».
2. Заказчик оплачивает выбранный им заказ.
3. Вариант использования завершается.

Основной поток событий варианта использования «Выдача чека»:

1. Вариант использования начинается, когда проект-менеджер прошел вариант использования «Оформление заказа» и «Оплата заказа».
2. Менеджер печатает чек.
3. Вариант использования завершается.

**Вывод по главе 2**

В данной главе проанализированы актеры и возможные варианты использования. Было изучено поведение системы (т.е. функциональность, которую она обеспечивает), системные прецеденты, системное окружение и связи между ними.

В этой главе были определены:

* Общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
* Сформировались общие требования к финальному поведению проектируемой системы
* Разработали исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей

Подготовили исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

# 3. Проектирование информационной системы с помощью метода проектирования «comet»

IDEF — методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем, позволяет отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах.

IDEF0 – контекстная диаграмма, являющаяся вершиной древовидной структуры диаграмм. Представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Основу методологии IDEF0 составляет графический язык описания процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Взаимодействие работ между собой и внешним миром описывается в виде стрелок. В IDEF0 различают 5 видов стрелок:

- вход (англ. input) – материал или информация, которые используются и преобразуются работой для получения результата (выхода). Вход отвечает на вопрос «Что подлежит обработке?». В качестве входа может быть как материальный объект (сырье, деталь, экзаменационный билет), так и не имеющий четких физических контуров (запрос к БД, вопрос преподавателя). Допускается, что работа может не иметь ни одной стрелки входа. Стрелки входа всегда рисуются входящими в левую грань работы;

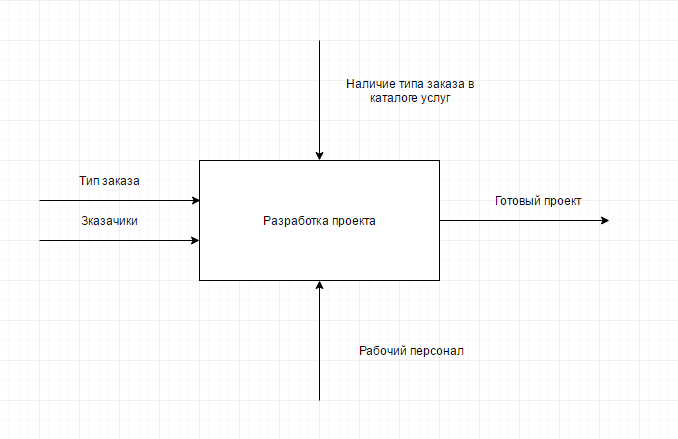
- управление (англ. control) – управляющие, регламентирующие и нормативные данные, которыми руководствуется работа. Управление отвечает на вопрос «В соответствии с чем выполняется работа?». Управление влияет на работу, но не преобразуется ей, т.е. выступает в качестве ограничения. В качестве управления могут быть правила, стандарты, нормативы, расценки, устные указания. Стрелки управления рисуются входящими в верхнюю грань работы. Если при построении диаграммы возникает вопрос, как правильно нарисовать стрелку сверху или слева, то рекомендуется ее рисовать как вход (стрелка слева);

- выход (англ. output) – материал или информация, которые представляют результат выполнения работы. Выход отвечает на вопрос «Что является результатом работы?». В качестве выхода может быть как материальный объект (деталь, автомобиль, платежные документы, ведомость), так и нематериальный (выборка данных из БД, ответ на вопрос, устное указание). Стрелки выхода рисуются исходящими из правой грани работы;

- механизм (англ. mechanism) – ресурсы, которые выполняют работу. Механизм отвечает на вопрос «Кто выполняет работу или посредством чего?». В качестве механизма могут быть персонал предприятия, студент, станок, оборудование, программа. Стрелки механизма рисуются входящими в нижнюю грань работы;

- вызов (англ. call) – стрелка указывает, что некоторая часть работы выполняется за пределами рассматриваемого блока. Стрелки выхода рисуются исходящими из нижней грани работы [3].

На рисунке 1 представлена контекстная диаграмма информационной системы.

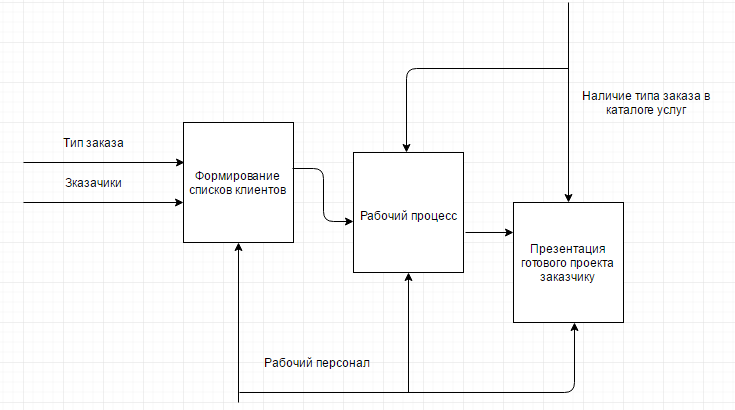


**Рисунок 1 – Общая диаграмма информационной системы**

На диаграмме, представленной на рисунке 1, мы видим, что:

1. Входными данными являются информация о типе заказа и информация о заказчиках.
2. Управляющим документом или управлением является документ «Каталог услуг инжинирингового предприятия».
3. Управляющим персоналом является арт-директор.
4. На выходе мы получаем готовый проект.

Декомпозиция — научный метод, использующий структуру задачи и позволяющий заменить решение одной большой задачи решением серии меньших задач, пусть и взаимосвязанных, но более простых. Декомпозиция общей функциональной модели представлена на рисунке 3.2



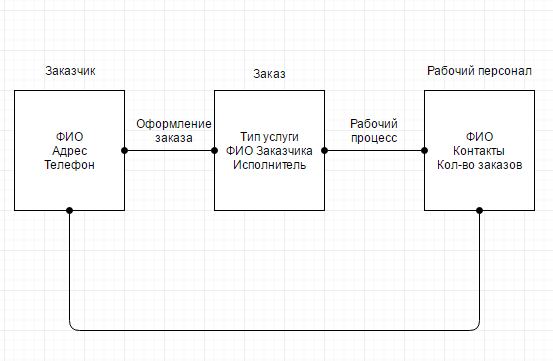
**Рисунок 2 – Декомпозиция общей функциональной диаграммы информационной системы**

На рисунке 2 мы видим, что информационная система разбивается на 3 основных блока:

* «Формирование списка клиентов», входными данными для данного блока являются: информация о типе заказа, информация о заказчике. Информация о типе заказа выделяется в отдельный поток, так как она является наиболее важной при функционировании системы. При завершении работы данного блока мы получаем рабочий процесс.
* «Рабочий процесс», входными данными является готовый список клиентов: тип заказа и заказчик. На выходе мы получаем презентацию готового проекта заказчику.
* «Презентация готового проекта заказчику», входными данными является результат рабочего процесса. На выходе мы получаем готовый проект, непосредственно переданный заказчику.

Так же из рисунка 2 видно, что инженера является управляющим персоналом на всех трёх блоках, от которого непосредственно зависит сам процесс разработки процесса.

Модель сущность-связь (ER-модель) — модель данных, позволяющая описывать схемы предметной области, выделять ключевые сущности и обозначать связи, которые могут устанавливаться между этими сущностями. На рисунке 3 представлена модель сущность-связь.



**Рисунок 3 – Модель сущность-связь информационной системы**

На рисунке 3 мы видим 3 сущности: заказчик, заказ, инженер.

Сущность «Заказчик» содержит в себе всю необходимую информацию о заказчиках.

Сущность «Заказ» содержит в себе всю необходимую информацию о заказах.

Сущность «Инженер» содержит в себе всю необходимую информацию о дизайнерах и их занятости.

При построении функциональной модели системы альтернативой методологии SADT (IDEF0) является методология диаграмм потоков данных (Data Flow Diagrams, DFD). В отличие от IDEF0, предназначенной для проектирования систем вообще, DFD предназначена для проектирования информационных систем. Ориентированность этой методологии на проектирование автоматизированных систем делает ее удобным и более выгодным инструментом при построении функциональной модели TO-BE.

Как и в IDEF0, основу методологии DFD составляет графический язык описания процессов.

Модель системы в нотации DFD представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе. Модель системы содержит контекстную диаграмму и диаграммы декомпозиции.

Принципы построения функциональной модели с помощью DFD аналогичны принципам методологии IDEF0. Вначале строится контекстная диаграмма, где отображаются связи системы с внешним окружением. В дальнейшем выполняется декомпозиция основных процессов и подсистем с построением иерархии диаграмм.

Согласно DFD источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям – потребителям информации

Поток данных определяет информацию (материальный объект), передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т. д.

Каждый поток данных имеет имя, отражающее его содержание. Направление стрелки показывает направление потока данных. Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним – двунаправленным.

На диаграммах IDEF0 потоки данных соответствуют входам и выходам, но в отличие от IDEF0 стрелки потоков на DFD могут отображаться входящими и выходящими из любой грани внешней сущности, процесса или накопителя данных.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом.

Каждый процесс должен иметь имя в виде предложения с глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже.

Преобразование информации может показываться как с точки зрения процессов, так и с точки зрения систем и подсистем. Если вместо имени процесса «Рассчитать допускаемую скорость» написать «Подсистема расчета допускаемых скоростей», тогда этот блок на диаграмме стоит рассматривать, как подсистему.

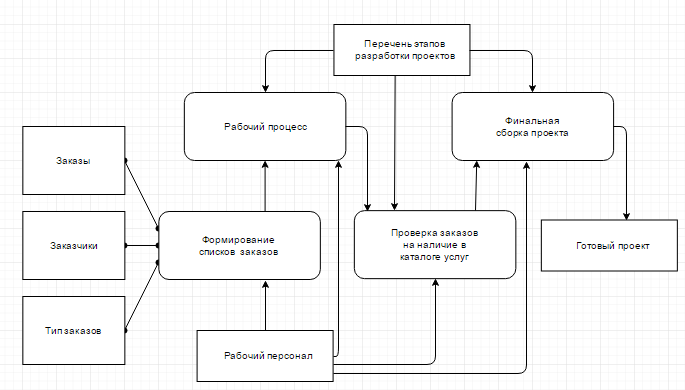
Накопитель (хранилище) данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде ящика в картотеке, области в оперативной памяти, файла на магнитном носителе и т.д.

Накопителю обязательно должно даваться уникальное имя и номер в пределах всей модели (всего набора диаграмм). Имя накопителя выбирается из соображения наибольшей информативности для разработчика. Таким образом, накопитель данных может представлять собой всю базу данных целиком, совокупность таблиц или отдельную таблицу. Такое представление накопителей в дальнейшем облегчит построение информационной модели системы.

Внешняя сущность (терминатор) представляет собой материальный объект или физическое лицо, выступающие как источник или приемник информации (заказчики, персонал, программа, склад, инструкция). Внешние сущности на DFD по смыслу соответствуют управлению и механизмам, отображаемым на контекстной диаграмме IDEF0.

Определение некоторого объекта, субъекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ проектируемой информационной системы. В связи с этим внешние сущности, как правило, отображаются только на контекстной диаграмме DFD. В процессе анализа и проектирования некоторые внешние сущности могут быть перенесены на диаграммы декомпозиции, если это необходимо, или, наоборот, часть процессов (подсистем) может быть представлена как внешняя сущность.



**Рисунок 4 – DFD модель информационной системы**

На данной диаграмме мы видим 4 внешние сущности: «Заказчики», «Заказы», «Тип заказов», «Инженера».

Так же на диаграмме присутствует один накопитель данных, который по сути является документом: «Перечень этапов разработки проекта», и ещё один, который по сути является информацией и не переносится на бумажные носители: «Проверка заказа на наличии в каталоге услуг».

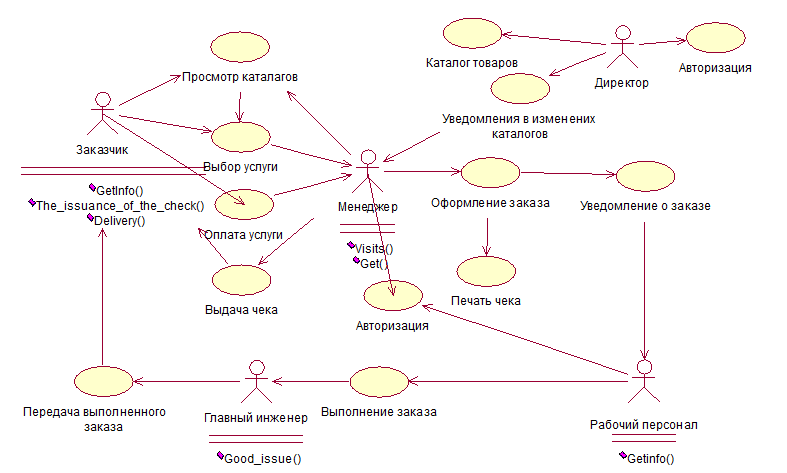
Все потоки данных являются однонаправленными, что видно из рисунка 4. Все остальные элементы данной диаграммы довольно стандартны и уже были описаны выше.

**4. Описание диаграмм**

**4.1 Диаграмма использования (Use-case диаграмма)**

Диаграмма использования или, как ее обычно называют, Use case - это описание сценария поведения, которому следуют действующие лица (Actors). Диаграмма вариантов использования описывает функциональное назначение системы, то есть то, что система будет делать в процессе своего функционирования. Эта диаграмма является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне [4].

На рисунке 5 представлена диаграмма вариантов использования.

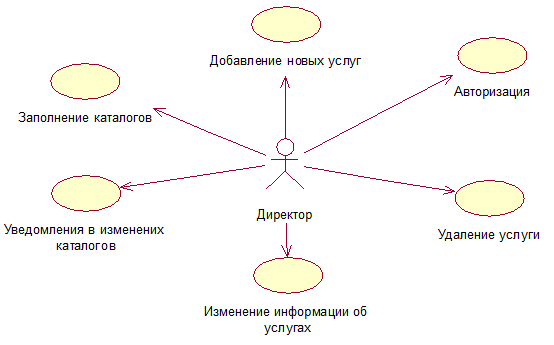


**Рисунок 5 – Диаграмма использования**

На диаграмме представлено абстрактное действующее лицо Пользователь. В свою очередь Пользователь подразделяется на конкретные лица: заказчик, арт-директор, проект-менеджер, дизайнер, ведущий дизайнер. Что бы получить доступ к приложению, пользователю необходимо пройти авторизацию. У проект-менеджера имеется свой логин и пароль, поэтому при вводе верного логина и пароля они получают доступ к приложению. Так как заказчик не знает логин и пароль, то и у доступа к приложению у него нет.

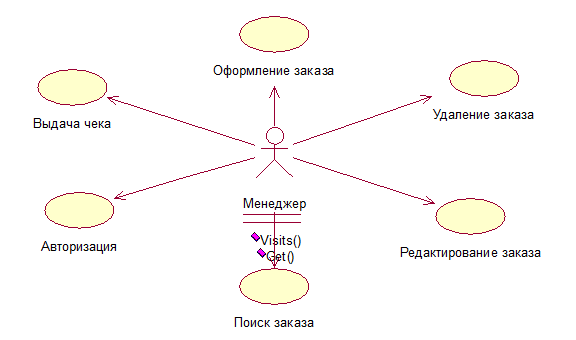
Заказчик приходит на предприятие – просматривает каталог – оформляет заказ – получает чек, в то время как менеджер заносит заказ в БД и уведомляет инженера о новом заказе, который в свою очередь принимает заказ, передает готовый проект ведущему инженеру, а он уже доставляет заказ клиенту.

На рисунке 6 изображена диаграмма использования, которая детализирует работу директора с приложением.



**Рисунок 6 – Диаграмма использования (детализация работы директора)**

На рисунке 7 изображена диаграмма использования, которая детализирует работу менеджера с приложением.



**Рисунок 7 – Диаграмма использования (детализация работы менеджера)**

**4.2 Диаграмма последовательности (Sequence Diagram).**

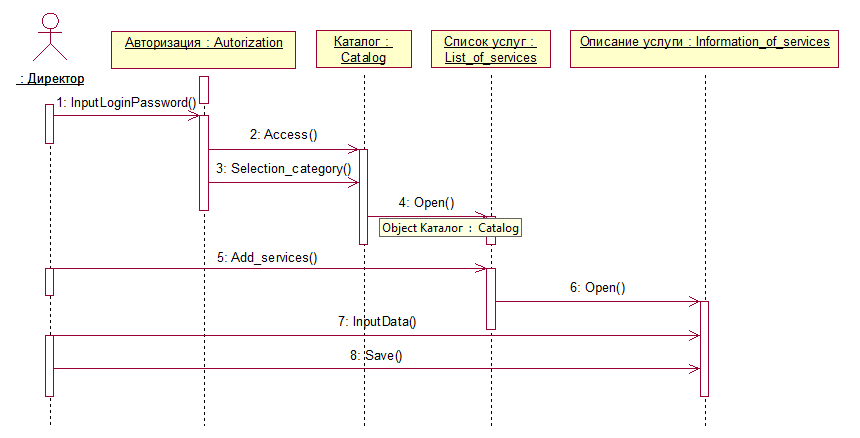
Этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

На диаграмме последовательности изображаются объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

Каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой [5].

Сообщение - законченный фрагмент информации, который отправляется одним объектом другому. При этом прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

На рисунке 8 представлена диаграмма последовательности действий директора.

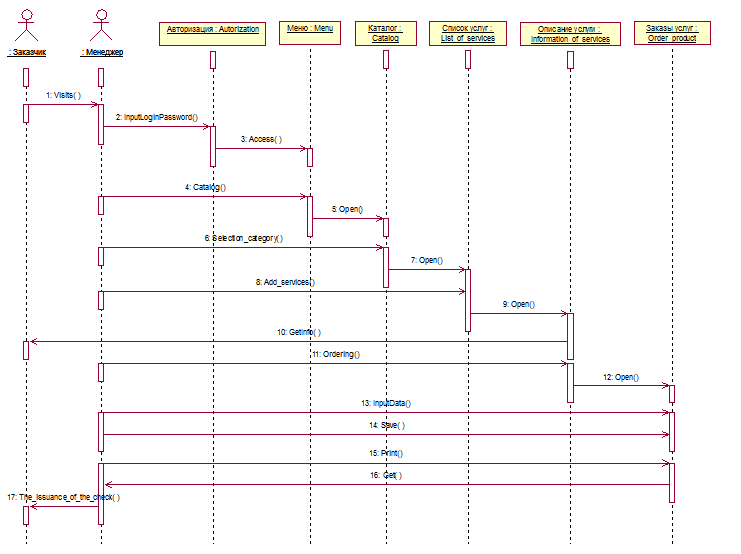


**Рисунок 8 – Диаграмма последовательности действий администратора**

Описание последовательности действий:

1. InputLoginPassword – ввод логина и пароля.
2. Access – получение доступа к начальной форме выбора действий.
3. Selection\_category – выбор категории услуг.
4. Open – открытие формы для ввода данных.
5. Add\_services – ввод нового товара.
6. Open – oткрытие формы для ввода информации о услуге.
7. InputData – ввод информации о новой услуге.
8. Save – сохранение данных.

На рисунке 9 представлена диаграмма последовательности действий оформления нового заказа.

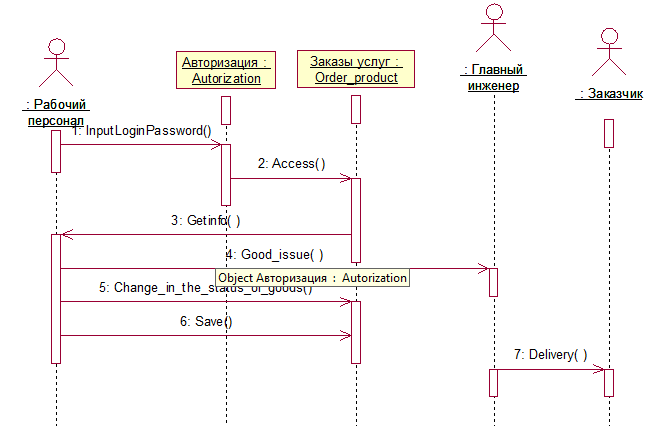


**Рисунок 9 – Диаграмма последовательности оформления нового заказа**

Описание последовательности действий:

1. Visits – клиент заказал услугу.
2. InputLoginPassword – ввод логина и пароля.
3. Access – получение доступа к меню.
4. Catalog – выбор пункта меню «Каталог».
5. Open – открытие каталога.
6. Selection\_category – выбор категории услуги.
7. Open – открытие списка товаров.
8. Add\_services – добавление нового товара.
9. Open – открытие формы «Информация о заказе».
10. GetInfo – ввод информации о новом заказе.
11. Ordering ­– утверждение заказа.
12. Open – открытие формы «Заказы услуг».
13. InputData – ввод информации о заказчике.
14. Savе – сохранение.
15. Print – печать чека.
16. Get – забираем чек.
17. The\_issuance\_of\_the\_check – выдача чека заказчику.

На рисунке 10 представлена диаграмма последовательности действий рабочего персонала.



**Рисунок 10 – Диаграмма последовательности действий складского персонала**

Описание последовательности действий:

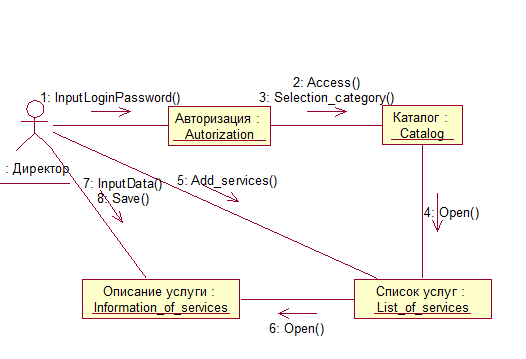
1. InputLoginPassword – ввод логина и пароля.
2. Access – доступ к заказам.
3. Getinfo – получение информации о новом заказе.
4. Good\_issue – выдача заказа ведущему инженеру.
5. Change\_in\_the\_status\_of\_goods – изменение статуса заказа.
6. Save – сохранение.
7. Delivery – доставка.

**4.3 Диаграмма кооперации (Collaboration Diagram)**

Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

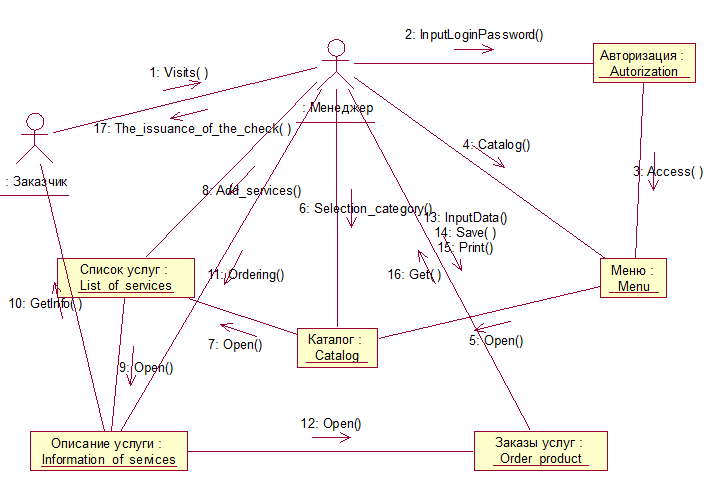
В отличие от диаграммы последовательности, на диаграмме кооперации изображаются только отношения между объектами, играющими определенные роли во взаимодействии. С другой стороны, на этой диаграмме не указывается время в виде отдельного измерения. Поэтому последовательность взаимодействий и параллельных потоков может быть определена с помощью порядковых номеров. Следовательно, если необходимо явно специфицировать взаимосвязи между объектами в реальном времени, лучше это делать на диаграмме последовательности.

На рисунке 11 представлена диаграмма кооперации действий арт-директора.



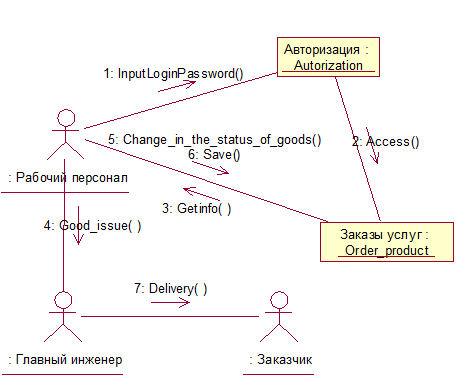
**Рисунок 11 – Диаграмма кооперации действий директора**

На рисунке 12 представлена диаграмма кооперации ввода нового заказа.



**Рисунок 12 – Диаграмма кооперации** **ввода нового заказа**

На рисунке 13 представлена диаграмма кооперации действий инженера.



**Рисунок 13 – Диаграмма кооперации** **действий рабочего персонала**

# 4.4 Диаграмма классов

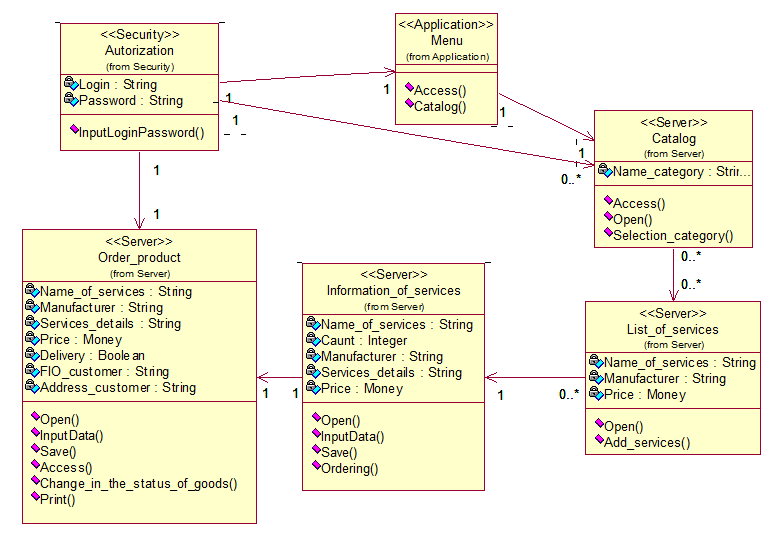
Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. Диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. Эти знания интерпретируются в базовых понятиях языка UML, таких как классы, интерфейсы и отношения между ними и их составляющими компонентами.

Класс в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Обязательным элементом обозначения класса является его имя. Оно должно быть уникальным. По мере проработки отдельных компонентов диаграммы, описания классов дополняются атрибутами (свойствами) и операциями (методами) [6].

Кроме внутреннего устройства или структуры классов, на соответствующей диаграмме указываются различные отношения между классами.

На рисунке 14 изображена главная диаграмма классов.



**Рисунок 14 – Главная диаграмма классов модели системы «инжинирингового предприятия»**

На диаграмме классов изображено 6 классов: «Autorization», «Menu», «Catalog», «List\_of\_services», «information\_of\_services», «Order\_product».

Для группировки классов используются следующие пакеты: Server, Application, Security. Пакет Security содержит класс Autorization, пакет Applicatin – класс Menu и пакет Server – Catalog, List\_of\_services, Information\_of\_services, Order\_Product.

Класс Autorization содержит такие атрибуты как Login ( String), Password ( String). Этот класс взаимосвязан с классом Menu и классом Catalog.

Класс Menu содержит в себе операции, такие как Access и Catalog. Этот класс взаимосвязан с классом Catalog.

Класс Catalog содержит в себе такие операции как Access, Open, Selection\_category и атрибуты: Name\_Category. Этот класс взаимосвязан с классом List\_of\_services.

Класс List\_of\_services содержит в себе такие операции как Open, AddServices и атрибуты: Name\_of\_services, Manufacturer,Price. Этот класс взаимосвязан с классом information\_of\_services.

Класс information\_of\_services содержит в себе такие операции как Open, InputData, Save, Ordering и атрибуты: Name\_of\_services, Count, Manufacturer, Product, Price.

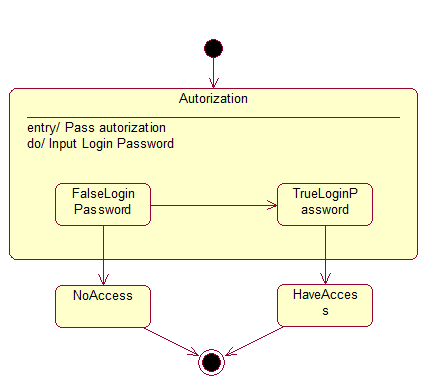
Класс Order\_product содержит в себе такие операции как Open, InputData, Save, Access, Print, Change\_in\_status\_of\_services и атрибуты: Name\_of\_services, Count, Manufacturer, Product, Price, Delivery, FIO\_customer, address\_customer.

**4.5 Диаграмма состояний (Statechart Diagram)**

Диаграмма состояний описывает процесс изменения состояний классов. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Именно для описания реакции объекта на подобные внешние воздействия и используются диаграммы состояний.

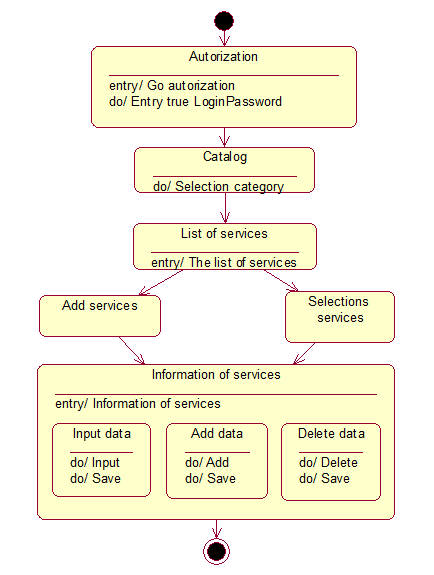
Главное предназначение этой диаграммы - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

На рисунке 15 представлена диаграмма состояний авторизации.



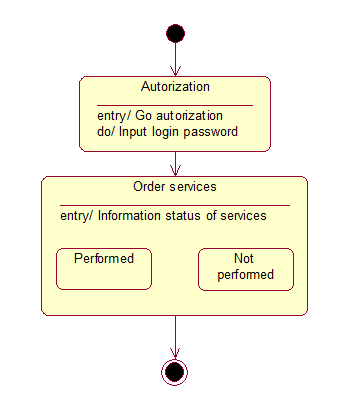
**Рисунок 15 – Диаграмма состояний авторизации**

На рисунке 16 представлена диаграмма состояний данных о услуге.



**Рисунок 16 – Диаграмма состояний данных о услуге**

На рисунке 17 представлена диаграмма состояний статуса заказа.

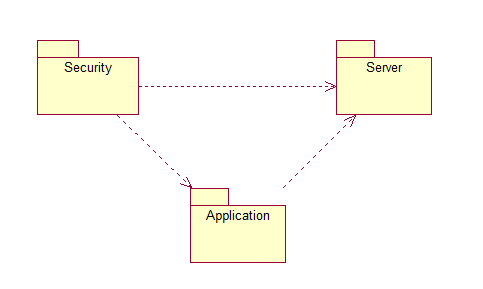


**Рисунок 17 – Диаграмма состояний статуса заказа**

# 4.6 Диаграмма компонентов (Component Diagram)

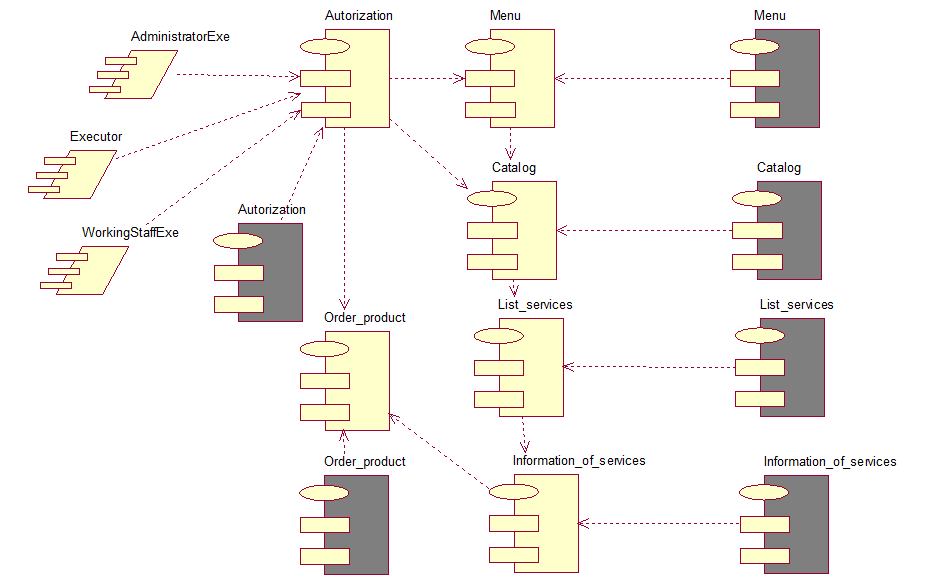
Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними [7].

На рисунке 18 изображены пакеты и их взаимосвязи.



**Рисунок 18 – Взаимосвязь пакетов**

На основание взаимосвязей пакетов, можно связать компоненты между собой. На рисунке 19 представлена диаграмма компонентов «System».

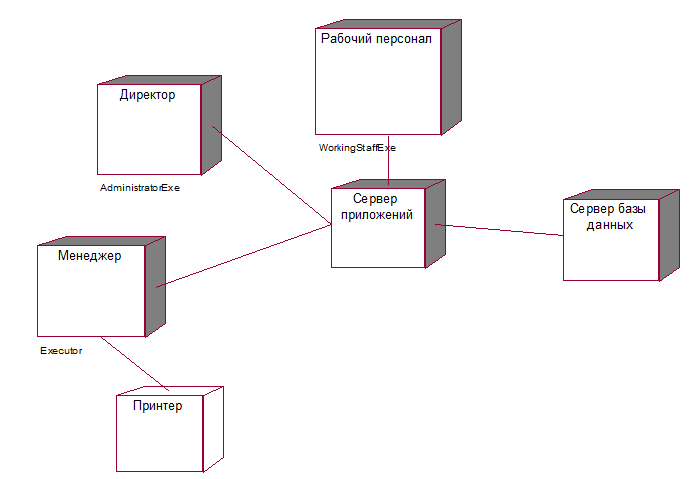


**Рисунок 19 – Диаграмма компонентов «System»**

* 1. **Диаграмма размещений (Statechart Diagram)**

Если разрабатываемая программа предполагает «локальную» работу на одном компьютере, то построением диаграммы компонентов проектирование системы и заканчивается. Однако сложные программные системы часто реализуются в сетевом варианте, предполагающем использование разных вычислительных платформ и технологий доступа к данным. Для представления общей конфигурации и топологии распределенной программной системы, а также маршрутов передачи информации между аппаратными устройствами применяется диаграмма развертывания.

На рисунке 20 представлена диаграмма размещений.



**Рисунок 20 - Диаграмма размещений**

**5. Описание генерации кода**

**Код ANSI C++**

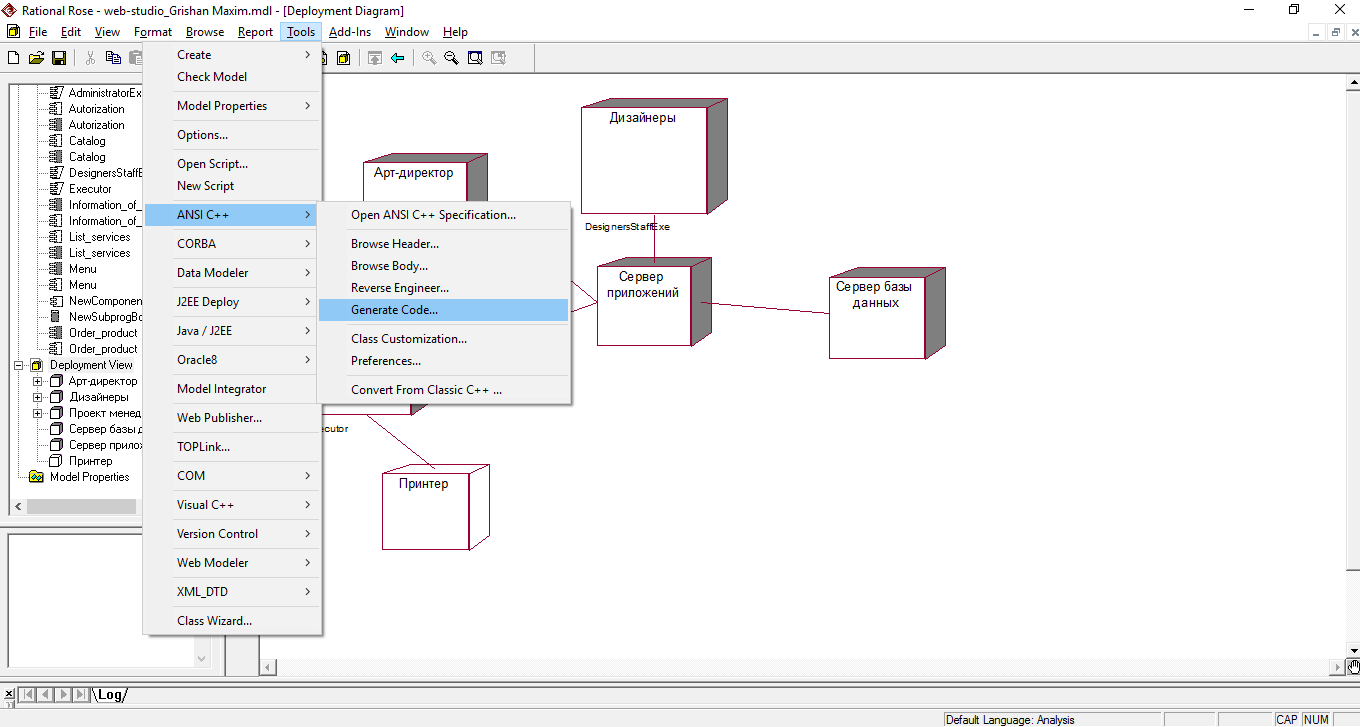
Rational Rose генерирует код, принимая во внимание номенклатуру созданных компонентов в совокупности с их стереотипами. Для каждого компонента без стереотипа система генерирует файл .h, содержащий информацию объявления и определения соответствующего класса. Если компонент снабжен стереотипом Package Specification, генерируется файл .h с объявлением класса. Если же при этом существует надлежащий компонент со стереотипом Package Body, генерируется и файл .срр с определением класса [8].

Как только компоненты, представляющие файлы .h и .срр, созданы, им должен быть поставлен в соответствие язык программирования (C++).

После создания компонентов, представляющих файлы .h, с ними следует сопоставить те или иные классы модели.

Код может быть сгенерирован для пакета в целом, для отдельного компонента либо группы компонентов. В качестве имени файла, в который помещается код, выбирается наименование пакета или компонента.

Для того, чтобы сгенерировать код, нужно выделить группу необходимых компонентов, затем меню Tools ⇒ ANSI C++ ⇒ Generate code (Рисунок 21).

**Рисунок 21 – Генерация кода ANSI C++**

Результат генерации кода можно увидеть в приложениях А и Б.

**Заключение**

В результате курсового проектирования была разработана информационная система "инжинирингового предприятия". Основой для создания информационной системы послужили проблемы предметной области. В качестве среды разработки было выбрано - IBM Rational Rose Enterprise Edition, предназначенное для автоматизации этапов анализа и проектирования предметной области.

После изучения универсального языка моделирования UML и принципов использования среды Rational Rose, была построена концептуальная и логическая модели информационной системы.

Использование данной ИС упрощает доступ к персональным данным заказчика, централизует хранение всех данных о заказчике. Благодаря этому представляется возможность сократить численности административно-управляющего персонала и расходов на зарплату, повысить уровень качества обслуживания. Все это приводит к повышению управляемости фирмы в целом.

Результаты проектирования могут являться основой для разработки конечного продукта информационной системы сети инжинирингового предприятия.

# Список использованных источников

1. Форум портал [Электронный ресурс] // URL: http://www.yaklass.ru/materiali?chtid=511&mode=cht.
2. Мастерская Dr.dimdim [Электронный ресурс] // URL: http://rema44.ru/resurs/study/dbprj/dbprj.html.
3. Форум info-system [Электронный ресурс] // URL: http://www.interface.ru/home.asp?artId=1635.
4. Инструментальные средства [Электронный ресурс] // URL: https://hlp4asp.wordpress.com/2011/2013/датологическое-моделирование/
5. Обучение в интернете [Электронный ресурс] // URL: http://www.lessons-tva.info/edu/e-inf2/m2t4\_4.html.
6. Официальный сайт Microsoft [Электронный ресурс] // URL: http://msdn.microsoft.com/ru-ru/library/ms175503.aspx#SSMSProcedure.
7. Портал образовательных ресурсов [Электронный ресурс] // URL: http://wiki.vspu.ru/workroom/pib22-2013/comp/index/lab4
8. Официальный сайт Microsoft SQL Server [Электронный ресурс] // URL: http://technet.microsoft.com/ru-ru/library/ms143506(v=sql.105).aspx.

# Приложение А

# Файлы Source

**Код класса «Autorization»**

#include "Autorization.h"

//##ModelId=566771D5001E

Autorization::InputLoginPassword()

{

}

**Код класса «Catalog»**

**#include "Catalog.h"**

**//##ModelId=566772F400DC**

**Catalog::Access()**

**{**

**}**

**//##ModelId=56677B9C038B**

**Catalog::Open()**

**{**

**}**

**//##ModelId=566A1C66030C**

**Catalog::Selection\_category()**

**{**

**}**

**Код класса «information\_of\_services»**

#**include "Iformation\_of\_goods.h"**

**//##ModelId=565CF3E50267**

**Iformation\_of\_services::Open()**

**{**

**}**

**//##ModelId=566777D10030**

**Iformation\_of\_services::InputData()**

**{**

**}**

**//##ModelId=566777EB02F7**

**Iformation\_of\_services::Save()**

**{**

**}**

**//##ModelId=5668B9FE009E**

**Iformation\_of\_services::Ordering()**

**{**

**}**

**Код класса «List\_of\_services»**

#include "List\_of\_services.h"

//##ModelId=566A1D7A0206

List\_of\_services::Open()

{

}

//##ModelId=566A1E44039E

List\_of\_services::Add\_services()

{

}

**Код класса «Menu»**

#include "Menu.h"

//##ModelId=56677B5B00D1

Menu::Access()

{

}

//##ModelId=56677B7B027A

Menu::Catalog()

{

}

**Код класса «Order\_product»**

#include "Order\_Product.h"

//##ModelId=566791B202AB

Order\_Product::Open()

{

}

//##ModelId=566791BA0248

Order\_Product::InputData()

{

}

//##ModelId=566791C502D8

Order\_Product::Save()

{

}

//##ModelId=5667935500C2

Order\_Product::Access()

{

}

//##ModelId=5667941A0190

Order\_Product::Change\_in\_the\_status\_of\_goods()

{

}

//##ModelId=566796250348

Order\_Product::Print()

{

}

**Приложение Б**

# Файлы Header

**Код класса «Autorization»**

#ifndef AUTORIZATION\_H\_HEADER\_INCLUDED\_A992502F

#define AUTORIZATION\_H\_HEADER\_INCLUDED\_A992502F

//##ModelId=566771C1028C

class Autorization

{

public:

//##ModelId=566771D5001E

InputLoginPassword();

private:

//##ModelId=5668B7840301

String Password;

//##ModelId=5668B769031F

String Login;

};

#endif /\* AUTORIZATION\_H\_HEADER\_INCLUDED\_A992502F \*/

**Код класса «Catalog»**

#ifndef CATALOG\_H\_HEADER\_INCLUDED\_A99210C5

#define CATALOG\_H\_HEADER\_INCLUDED\_A99210C5

//##ModelId=565CF3390125

class Catalog

{

public:

//##ModelId=566772F400DC

Access();

//##ModelId=56677B9C038B

Open();

//##ModelId=566A1C66030C

Selection\_category();

private:

//##ModelId=566A28A70112

String Name\_category;

};

#endif /\* CATALOG\_H\_HEADER\_INCLUDED\_A99210C5 \*/Код класса

**«information\_of\_services»**

#ifndef IFORMATION\_OF\_SERVICES\_H\_HEADER\_INCLUDED\_A9926BAB

#define IFORMATION\_OF\_SERVICES\_H\_HEADER\_INCLUDED\_A9926BAB

//##ModelId=565CF38A0006

class Iformation\_of\_services

{

public:

//##ModelId=565CF3E50267

Open();

//##ModelId=566777D10030

InputData();

//##ModelId=566777EB02F7

Save();

//##ModelId=5668B9FE009E

Ordering();

private:

//##ModelId=566A1B1E028B

String Name\_of\_services;

//##ModelId=566A230503D0

Integer Caunt;

//##ModelId=566A243C0228

String Manufacturer;

//##ModelId=566A24AE0075

String Services\_Details;

//##ModelId=566A25DC020D

Money Price;

};

#endif /\* IFORMATION\_OF\_SERVICES\_H\_HEADER\_INCLUDED\_A9926BAB \*/

**Код класса «List\_of\_services»**

#ifndef LIST\_OF\_SERVICES\_H\_HEADER\_INCLUDED\_A9925C95

#define LIST\_OF\_SERVICES\_H\_HEADER\_INCLUDED\_A9925C95

//##ModelId=566A1D2C0332

class List\_of\_services

{

public:

//##ModelId=566A1D7A0206

Open();

//##ModelId=566A1E44039E

Add\_of\_services();

private:

//##ModelId=566A294403AE

String Name\_of\_services;

//##ModelId=566A2962004E

String Manufacturer;

//##ModelId=566A29AB03E1

Money Price;

};

#endif /\* LIST\_GOODS\_H\_HEADER\_INCLUDED\_A9925C95 \*/

**Код класса «Menu»**

#ifndef MENU\_H\_HEADER\_INCLUDED\_A9922F29

#define MENU\_H\_HEADER\_INCLUDED\_A9922F29

//##ModelId=565CF2AD0336

class Menu

{

public:

//##ModelId=56677B5B00D1

Access();

//##ModelId=56677B7B027A

Catalog();

};

#endif /\* MENU\_H\_HEADER\_INCLUDED\_A9922F29 \*/

**Код класса «Order\_product»**

#ifndef ORDER\_PRODUCT\_H\_HEADER\_INCLUDED\_A9926212

#define ORDER\_PRODUCT\_H\_HEADER\_INCLUDED\_A9926212

//##ModelId=566791460192

class Order\_Product

{

public:

//##ModelId=566791B202AB

Open();

//##ModelId=566791BA0248

InputData();

//##ModelId=566791C502D8

Save();

//##ModelId=5667935500C2

Access();

//##ModelId=5667941A0190

Change\_in\_the\_status\_of\_goods();

//##ModelId=566796250348

Print();