Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования

**Поволжский государственный университет
телекоммуникаций и информатики**

Кафедра экономических и информационных систем

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания для выполнения курсового проекта для студентов специальности 230700 (Прикладная информатика) заочной формы образования

Составители: Диязитдинова А.Р.,

Самара, 2013 г.

Содержание

ЦЕЛИ И ЗАДАЧИ КУРСОВОГО ПРОЕКТА

Цель курсового проектирования – применение на практике знаний, полученных в процессе изучения курса «Проектирование информационных систем», и приобретение практических навыков при проектировании и создании информационных систем (ИС), с использованием методологии UML.

1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Объектно-ориентированный подход к проектированию ИС

В основе проектирования ИС лежит моделирование предметной области. Для того чтобы получить адекватный предметной области проект ИС в виде системы правильно работающих программ, необходимо иметь целостное, системное представление модели, которое отражает все аспекты функционирования будущей информационной системы. При этом под моделью предметной области понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное моделирование предметной области позволяет сократить время и сроки проведения проектировочных работ и получить более эффективный и качественный проект. Без проведения моделирования предметной области велика вероятность допущения большого количества ошибок в решении стратегических вопросов, приводящих к экономическим потерям и высоким затратам на последующее перепроектирование системы. Вследствие этого все современные технологии проектирования ИС основываются на использовании методологии моделирования предметной области.

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные).

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых функций интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

Функциональные методики рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от объектной методики заключается в четком отделении функций (методов обработки данных) от самих данных.

Объектно-ориентированный подход использует объектную декомпозицию, при этом статическая структура описывается в терминах объектов и связей между ними, а поведение системы описывается в терминах обмена сообщениями между объектами. Целью методики является построение бизнес-модели организации, позволяющей перейти от модели сценариев использования к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.

Концептуальной основой объектно-ориентированного подхода является объектная модель, которая строится с учетом следующих принципов: абстрагирование; инкапсуляция; модульность; иерархия; типизация; параллелизм; устойчивость.

Основными понятиями объектно-ориентированного подхода являются объект и класс.

Объект – предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс. ***Класс*** – это множество объектов, связанных общностью структуры и поведения. Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие полиморфизм может быть интерпретировано как способность класса принадлежать более чем одному типу. Наследование означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Большинство существующих методов объектно-ориентированного подхода включают язык моделирования и описание процесса моделирования. ***Процесс*** – это описание шагов, которые необходимо выполнить при разработке проекта. В качестве языка моделирования объектного подхода используется унифицированный язык моделирования UML, который содержит стандартный набор диаграмм для моделирования.

Диаграмма (Diagram) – это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Объектно-ориентированный подход обладает следующими преимуществами:

- дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей;
- позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем;
- естественна, поскольку ориентированна на человеческое восприятие мира.

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.

1.2 Унифицированный язык объектно-ориентированного моделирования UML

Унифицированный язык моделирования UML (Unified Modeling Language) представляет собой язык для определения, представления, проектирования и документирования программных систем, организационно-экономических систем, технических систем других систем различной природы.

UML содержит стандартный набор диаграмм для моделирования:

- диаграммы вариантов использования (use case diagrams);
- диаграммы классов (class diagrams);
- диаграммы поведения системы (behavior diagrams):
  - диаграммы взаимодействия (interaction diagrams);
    - диаграммы последовательности (sequence diagrams)
    - кооперативные диаграммы (collaboration diagrams);
  - диаграммы состояний (statechart diagrams);
  - диаграммы деятельностей (activity diagrams);

- диаграммы реализации (implementation diagrams):
  - диаграммы компонентов (component diagrams);
  - диаграммы размещения (deployment diagrams).

1.2.1 Диаграммы вариантов использования

Визуальное моделирование с использованием нотации UML можно представить как процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной бизнес-системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что бизнес-система должна делать в процессе своего функционирования.

Диаграмма вариантов использования (use case diagram) – диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Диаграмма вариантов использования - это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки. Создание диаграммы вариантов использования имеет следующие цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Назначение данной диаграммы состоит в следующем: проектируемая программная система представляется в форме так называемых вариантов использования, с которыми взаимодействуют внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой бизнес-системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами каждый вариант использования определяет набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой и собственно выполнение вариантов использования.

Базовыми элементами диаграммы вариантов использования являются вариант использования и актер.

Вариант использования (use case) – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами. Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать

Актер (actor) - представляют собой роли, а не конкретных людей или наименования работ. Актер может также быть внешней системой, которой необходима информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования.

В языке UML имеется несколько стандартных видов отношений между актерами и вариантами использования:

- ассоциации (association relationship) – **Рис. 1**;
- включения (include relationship) – **Рис. 2**;

– расширения (extend relationship) – **Рис. 3**
– обобщения (generalization relationship) – **Рис. 4**.

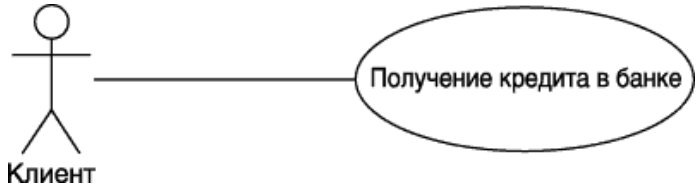

Рис. 1. Пример графического представления отношения ассоциации между актером и вариантом использования

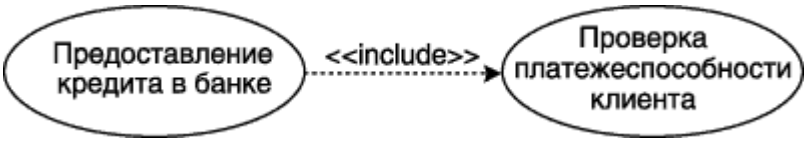

Рис. 2. Пример графического изображения отношения включения между вариантами использования

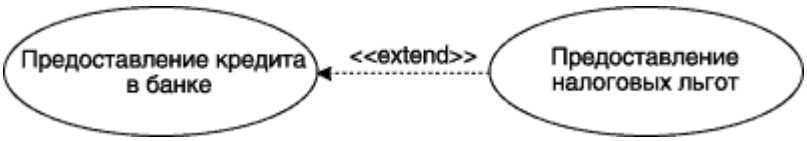

Рис. 3. Пример графического изображения отношения расширения между вариантами использования

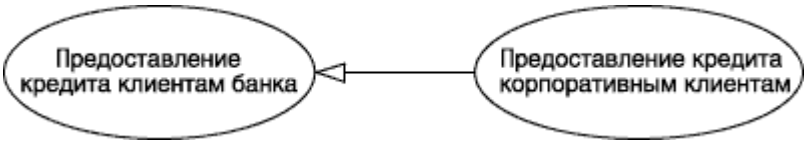

Рис. 4. Пример графического изображения отношения обобщения между вариантами использования

1.2.2 Диаграммы классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами. На данной диаграмме не указывается информация о временных аспектах функционирования системы.

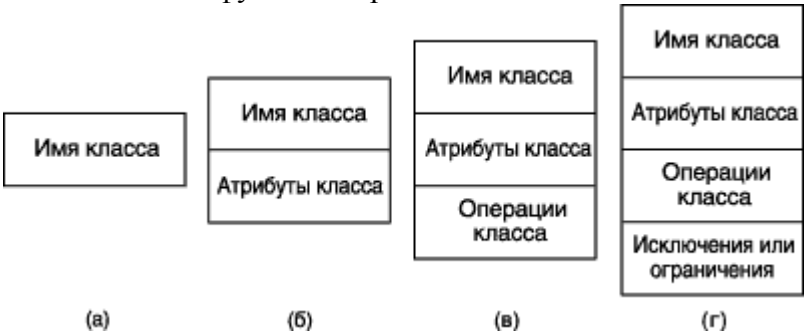

Рис. 5. Варианты графического изображения класса на диаграмме классов

Одним из несомненных достоинств языка UML является наличие механизмов расширения, которые позволяют ввести в рассмотрение дополнительные графические обозначения, ориентированные для решения задач из определенной предметной области. Язык UML содержит два специальных расширения: профиль для процесса разработки программного обеспечения (The UML Profile for Software Development Processes) и профиль для бизнес-моделирования (The UML Profile for Business Modeling).

В рамках первого из них предложено три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении различных диаграмм (**Рис. 6**):

- Управляющий класс (control class) – класс, отвечающий за координацию действий других классов. На каждой диаграмме классов должен быть хотя бы один управляющий класс, причем количество посылаемых объектам управляющего класса сообщений мало, по сравнению с числом рассылаемых ими. Управляющий класс отвечает за координацию действий других классов. У каждой диаграммы классов должен быть хотя бы один управляющий класс, контролирующий последовательность выполнения действий этого варианта использования. Как правило, данный класс является активным и инициирует рассылку множества сообщений другим классам модели. Кроме специального обозначения управляющий класс может быть изображен в форме прямоугольника класса со стереотипом <<control>>.
- Класс-сущность (entity class) – пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы. Класс-сущность содержит информацию, которая должна храниться постоянно и не уничтожается с уничтожением объектов данного класса или прекращением работы моделируемой системы, связанные с выключением системы или завершением программы. Как правило, этот класс соответствует отдельной таблице базы данных. В этом случае его атрибуты являются полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс пассивный и лишь принимает сообщения от других классов модели. Класс-сущность может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<entity>> .
- Граничный класс (boundary class) – класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но является составной частью системы. Граничный класс может быть изображен также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>>.

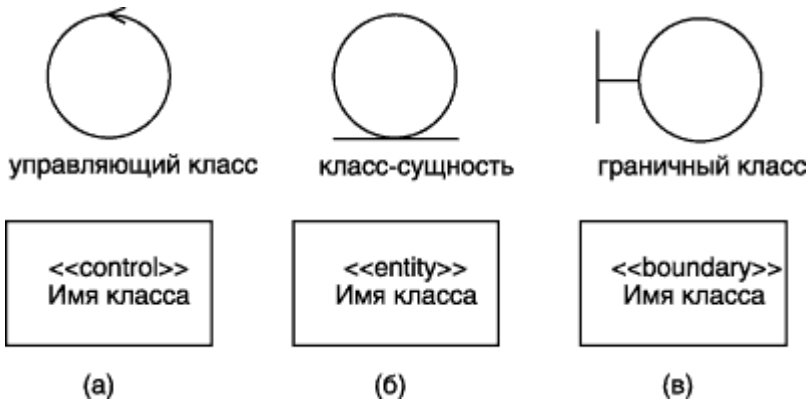

Рис. 6. Графическое изображение классов для моделирования программного обеспечения

В рамках второго профиля также предложено три специальных графических примитива, которые могут быть использованы для уточнения семантики отдельных классов при построении моделей бизнес-систем:

- Сотрудник (business worker) – класс, служащий на диаграмме классов для представления любого сотрудника, который является элементом бизнес-системы и взаимодействует с другими сотрудниками при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<worker>> или <<internalWorker>>.
- Сотрудник для связи с окружением (caseworker) – класс, служащий для представления в бизнес-системе такого сотрудника, который, являясь элементом бизнес-системы, непосредственно взаимодействует с актерами (бизнес-актерами) при реализации бизнес-процесса. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<caseWorker>>.
- Бизнес-сущность (business entity) – специальный случай класса-сущности, который также не инициирует никаких сообщений. Этот класс служит для сохранения информации о результатах выполнения бизнес-процесса в моделируемой бизнес-системе или организации. Этот класс также может быть изображен в форме прямоугольника класса со стереотипом <<business entity>>.

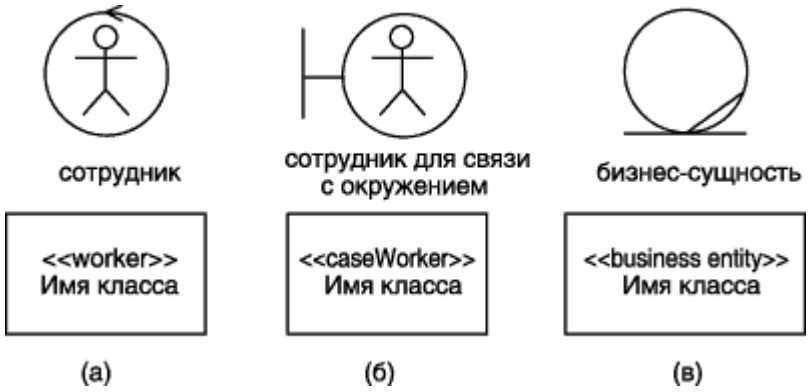

Рис. 7. Графическое изображение классов для моделирования бизнес-систем

Базовые отношения, изображаемые на диаграммах классов:
- Отношение ассоциации (association relationship) – **Рис. 8**;
- Отношение обобщения (generalization relationship) – **Рис. 9**;
- Отношение агрегации (aggregation relationship) – **Рис. 10**;
- Отношение композиции (composition relationship) – **Рис. 11**.

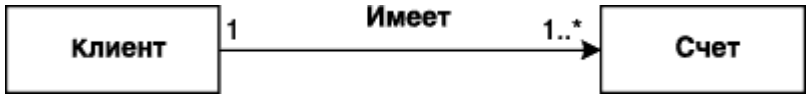

Рис. 8. Графическое изображение направленной бинарной ассоциации между классами

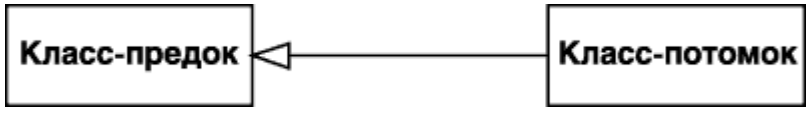

Рис. 9. Графическое изображение отношения обобщения

Рис. 10. Графическое изображение отношения агрегации

Рис. 11. Графическое изображение отношения композиции

1.2.3 Диаграммы состояний

Диаграммы состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. Таким образом, диаграмма состояний используется для моделирования поведения объектов системы при переходе из одного состояния в другое.

Состояние может быть задано в виде набора конкретных значений атрибутов объекта некоторого класса, при этом изменение отдельных значений этих атрибутов будет отражать изменение состояния моделируемого объекта или системы в целом. Однако не каждый атрибут класса может характеризовать состояние его объектов. Как правило, имеют значение только те свойства элементов системы, которые отражают динамический или функциональный аспект ее поведения. В этом случае состояние будет характеризоваться некоторым инвариантным условием, включающим в себя только принципиальные для поведения объекта или системы атрибуты классов и их значения.

Такое условие может соответствовать ситуации, когда моделируемый объект находится в состоянии ожидания возникновения внешнего события. В то же время нахождение объекта в некотором состоянии может быть связано с выполнением определенных действий. В последнем случае соответствующая деятельность начинается в момент перехода моделируемого элемента в рассматриваемое состояние, а после и элемент может покинуть данное состояние в момент завершения этой деятельности.

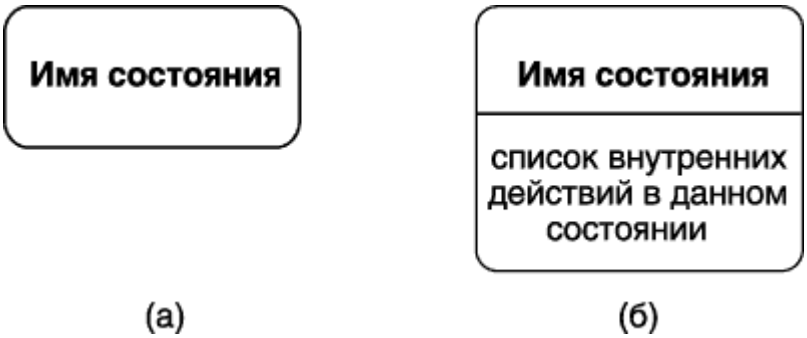

Рис. 12. Графическое изображение состояний на диаграмме состояний

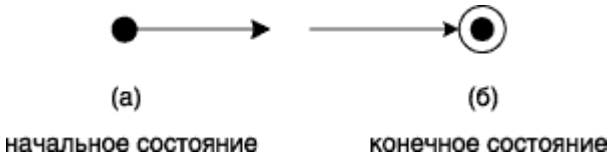

Рис. 13. Графическое изображение начального и конечного состояний

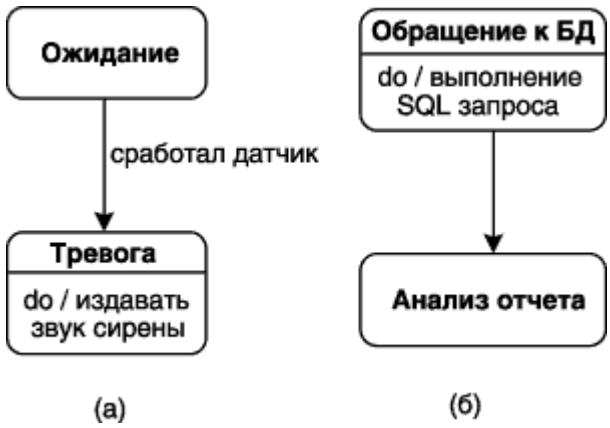

Рис. 14. Графическое изображение триггерного (а) и нетриггерного (б) переходов на диаграмме состояний

1.2.4 Диаграммы деятельностей

При моделировании поведения проектируемой или анализируемой программной системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и процедурной реализации выполняемых системой операций. Для этой цели, как правило, используются блок-схемы или структурные схемы алгоритмов. Каждая такая схема акцентирует внимание на последовательности выполнения определенных процедур или элементарных операций, которые в совокупности приводят к получению желаемого результата**.**

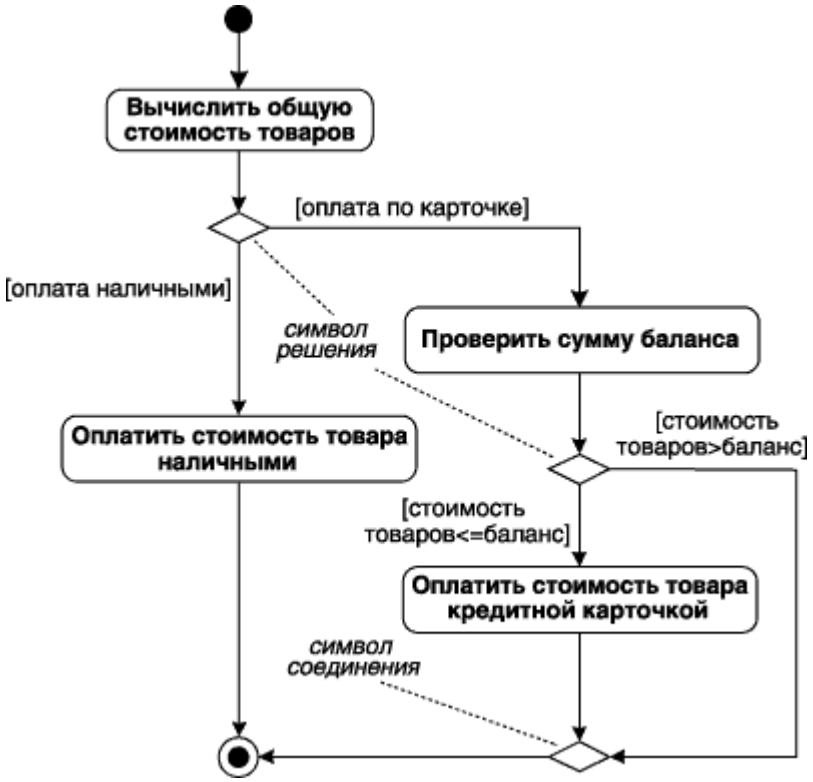

Рис. 15. Различные варианты ветвлений на диаграмме деятельности

1.2.5 Диаграммы взаимодействия

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображаются ряд объектов и те сообщения, которыми они обмениваются между собой.

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Говоря об этих диаграммах, имеют в виду два аспекта взаимодействия. *Во-первых*, взаимодействия объектов можно рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Во-вторых, можно рассматривать структурные особенности взаимодействия объектов. Для представления структурных особенностей передачи и приема сообщений между объектами используется диаграмма кооперации.

Диаграммы последовательности

Диаграмма последовательности (sequence diagram) - диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

На диаграмме последовательности неявно присутствует ось времени, что позволяет визуализировать временные отношения между передаваемыми сообщениями. С помощью диаграммы последовательности можно представить взаимодействие элементов модели как своеобразный временной график "жизни" всей совокупности объектов, связанных между собой для реализации варианта использования программной системы, достижения бизнес-цели или выполнения какой-либо задачи.

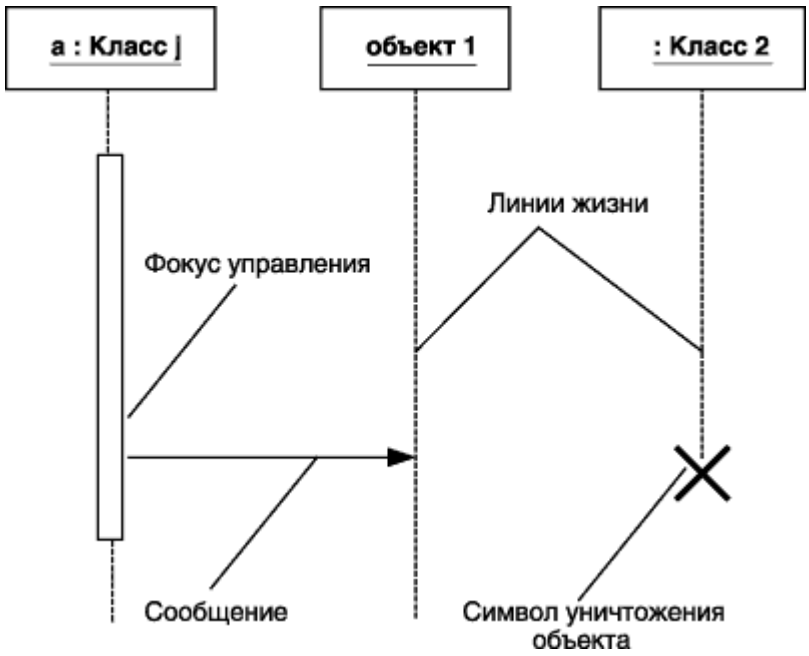

Рис. 16. Графические элементы диаграммы последовательности

Кооперативные диаграммы

Диаграмма кооперации предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования. С точки зрения аналитика или

архитектора системы в проекте важно представить структурные связи отдельных объектов между собой. Такое представление структуры модели как совокупности взаимодействующих объектов и обеспечивает диаграмма кооперации.

Кооперация (collaboration) – спецификация множества объектов отдельных классов, совместно взаимодействующих с целью реализации отдельных вариантов использования в общем контексте моделируемой системы.

Понятие кооперации – одно из фундаментальных в языке UML. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных вариантов использования или наиболее значимых операций в системе. Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

На диаграмме кооперации размещаются объекты, представляющие собой экземпляры классов, связи между ними, которые в свою очередь являются экземплярами ассоциаций и сообщения. Связи дополняются стрелками сообщений, при этом показываются только те объекты, которые участвуют в реализации моделируемой кооперации. Далее, как и на диаграмме классов, показываются структурные отношения между объектами в виде различных соединительных линий. Связи могут дополняться именами ролей, которые играют объекты в данной взаимосвязи. И, наконец, изображаются динамические взаимосвязи — потоки сообщений в форме стрелок с указанием направления рядом с соединительными линиями между объектами, при этом задаются имена сообщений и их порядковые номера в общей последовательности сообщений.

Одна и та же совокупность объектов может участвовать в реализации различных кооперaций. В зависимости от рассматриваемой кооперации, могут изменяться как связи между отдельными объектами, так и поток сообщений между ними. Именно это отличает диаграмму кооперации от диаграммы классов, на которой должны быть указаны все без исключения классы, их атрибуты и операции, а также все ассоциации и другие структурные отношения между элементами модели.

1.2.6 Диаграммы компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними.

Физическая система (physical system) – реально существующий прототип модели системы. Для представления физических сущностей в языке UML применяется специальный термин – компонент.

Компонент (component) – физически существующая часть системы, которая обеспечивает реализацию классов и отношений, а также функционального поведения моделируемой программной системы. Компонент предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты – собственное графическое представление. Компонентом может быть исполняемый код отдельного модуля, командные файлы или файлы, содержащие интерпретируемые скрипты.

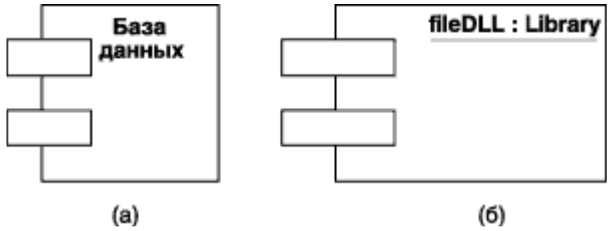

Рис. 17. Графическое изображение компонента

1.2.7 Диаграммы размещения

Диаграмма размещения (deployment diagram) - диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений - маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих только на этапе ее исполнения (run-time). При этом представляются только те компоненты программы, которые являются исполнимыми файлами или динамическими библиотеками. Компоненты, не используемые на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единственной для системы в целом, поскольку должна отражать все особенности ее реализации. Эта диаграмма, по сути, завершает процесс ООАП для конкретной программной системы и ее разработка, как правило, последний этап спецификации модели.

Рис. 18. Варианты изображения графических стереотипов узлов

2. ПРИМЕР РАЗРАБОТКИ ПРОЕКТА ИНФОРМАЦИОННОЙ СИСТЕМЫ С ПОМОЩЬЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА

В данном разделе приводится пример описания основных разделов курсового проекта. В качестве примера создания модели рассмотрим фрагмент проекта системы, организующей работу банкомата по обслуживанию клиента по его кредитной карте.

2.1 Диаграмма вариантов использования

На **Рис. 19** показан пример такой диаграммы для банковской системы с банкоматами. На данной диаграмме человеческие фигурки обозначают действующих лиц, овалы - варианты использования, а линии и стрелки - различные связи между действующими лицами и вариантами использования.

Для того чтобы фактически разработать систему, однако потребуются более конкретные детали. Эта детали описываются в документе, называемом «поток событий» (flow of events). Целью потока событий является документирование процесса обработки данных, реализуемого в рамках варианта использования. Этот документ подробно описывает, что будут делать пользователи системы и что - сама система.

Хотя поток событий и описывается подробно, он также не должен зависеть от реализации. Цель - описать то, что будет делать система, а не как она будет делать это. Обычно поток событий включает:

- краткое описание;
- предусловия (pre-conditions);
- основной поток событий;
- альтернативный поток событий;
- постусловия (post-conditions).

Рис. 19. Диаграмма вариантов использования

Краткое описание

Каждый вариант использования должен иметь краткое описание того, что он будет делать. Например, вариант использования «Перевести деньги» может содержать следующее описание:

Вариант использования «Перевести деньги» позволяет клиенту или служащему банка переводить деньги с одного счета до востребования или сберегательного счета на другой.

Предусловия

Предусловия варианта использования - это такие условия, которые должны быть выполнены, прежде чем вариант использования начнет выполняться сам. Например, таким условием может быть выполнение другого варианта использования или наличие у пользователя прав доступа, требуемых для запуска этого. Не у всех вариантов использования бывают предусловия.

Ранее упоминалось, что диаграммы вариантов использования не должны отражать порядок их выполнения. С помощью предусловий, однако, можно документировать и такую информацию. Так, предусловием одного варианта использования может быть то, что в это время должен выполняться другой.

Основной и альтернативный потоки событий

Конкретные детали вариантов использования описываются в основном и альтернативных потоках событий. Поток событий поэтапно описывает, что должно происходить во время выполнения заложенной в варианты использования функциональности, Поток событий уделяет внимание тому, что будет делать система, а не как она будет делать это, причем описывает все это с точки зрения пользователя. Основной и альтернативный потоки событий включают следующее описание:

- каким образом запускается вариант использования;

- различные пути выполнения варианта использования;
- нормальный, или основной, поток событий варианта использования;
- отклонения от основного потока событий (так называемые альтернативные потоки);
- потоки ошибок;
- каким образом завершается вариант использования.

Например, поток событий варианта использования «Снять деньги со счета» может выглядеть следующим образом:

Основной поток

1. Вариант использования начинается, когда клиент вставляет свою карточку в банкомат.
2. Банкомат выводит приветствие и предлагает клиенту ввести свой персональный PIN-код.
3. Клиент вводит PIN-код.
4. Банкомат подтверждает введенный код. Если код не подтвержден, выполняется альтернативный поток событий А1.
5. Банкомат выводит список доступных действий:
   - сделать вклад;
   - снять деньги со счета;
   - перевести деньги.
6. Клиент выбирает пункт «Снять деньги со счета».
7. Банкомат запрашивает, сколько денег надо снять.
8. Клиент вводит требуемую сумму.
9. Банкомат определяет, имеется ли на счете достаточно денег. Если денег недостаточно, выполняется альтернативный поток А2. Вели во время подтверждения суммы возникают ошибки, выполняется поток ошибок Е1.
10. Банкомат вычитает требуемую сумму из счета клиента.
11. Банкомат выдает клиенту требуемую сумму наличными.
12. Банкомат возвращает клиенту его карточку.
13. Банкомат печатает чек для клиента.
14. Вариант использования завершается.

Альтернативный поток А1. Ввод неправильного PIN-кода

1. Банкомат информирует клиента, что код введен неправильно.
2. Банкомат возвращает клиенту его карточку.
3. Вариант использования завершается.

Альтернативный вариант использования А2. Недостаточно денег на счете

1. Банкомат информирует клиента, что денег на его счете недостаточно.
2. Банкомат возвращает клиенту его карточку.
3. Вариант использования завершается.

Поток ошибок El. Ошибка в подтверждении запрашиваемой суммы

1. Банкомат сообщает пользователю, что при подтверждении запрашиваемой суммы произошла ошибка, и дает ему номер телефона службы поддержки клиентов банка.
2. Банкомат заносит сведения об ошибке в журнал ошибок. Каждая запись содержит дату и время ошибки, имя клиента, номер его счета и код ошибки.
3. Банкомат возвращает клиенту его карточку.
4. Вариант использования завершается.

Постусловия

Постусловиями называются такие условия, которые всегда должны быть выполнены после завершения варианта использования. Например, в конце варианта использования можно пометить флажком какой-нибудь переключатель. Информация такого типа вводит в состав постусловий. Как и для предусловий, с помощью постусловий можно вводить информацию о порядке выполнения вариантов использования системы. Если, например, после одного

из вариантов использования должен всегда выполняться другой, это можно описать как постусловие. Такие условия имеются не у каждого варианта использования.

2.2 Диаграмма последовательности

Диаграммы последовательности отражают поток событий, происходящих в рамках варианта использования. Например, вариант использования «Снять деньги со счета» предусматривает несколько возможных последовательностей, таких, как снятие денег, попытка снять деньги, не имея их достаточного количества на счете, попытка снять деньги по неправильному PIN-коду и некоторых других. Нормальный сценарий снятия некоторой суммы денег со счета показан на **Рис. 20**. Под сценарием понимается конкретный экземпляр потока событий.

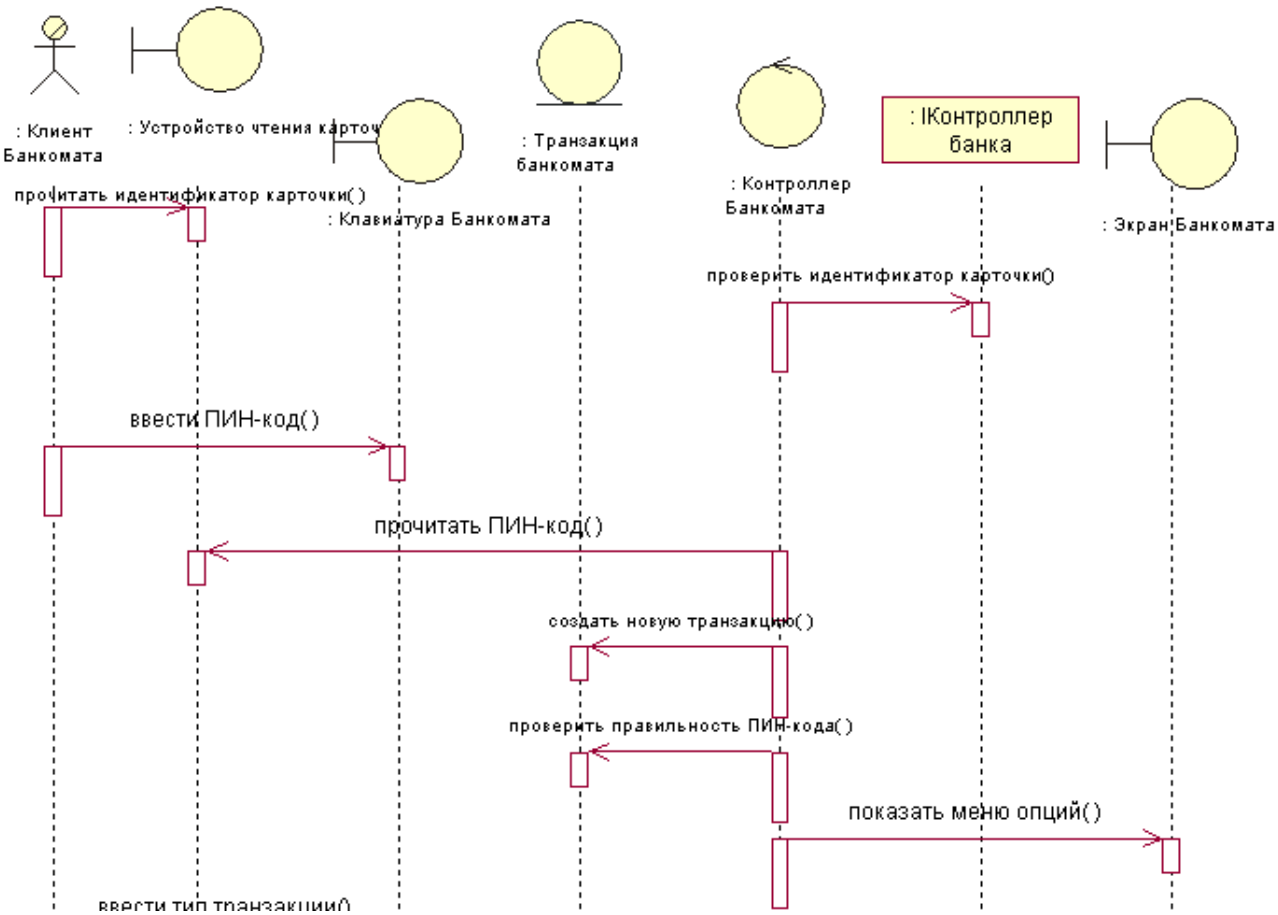

Рис. 20 Фрагмент диаграммы последовательности в рамках варианта использования «Снятие денег по кредитной карточке»

2.3 Диаграмма кооперации

На **Рис. 21** приведена кооперативная диаграмма, описывающая, как клиент снимает деньги со счета.

Как видно из рисунка, здесь представлена вся та информация, которая была и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако труднее уяснить последовательность событий.

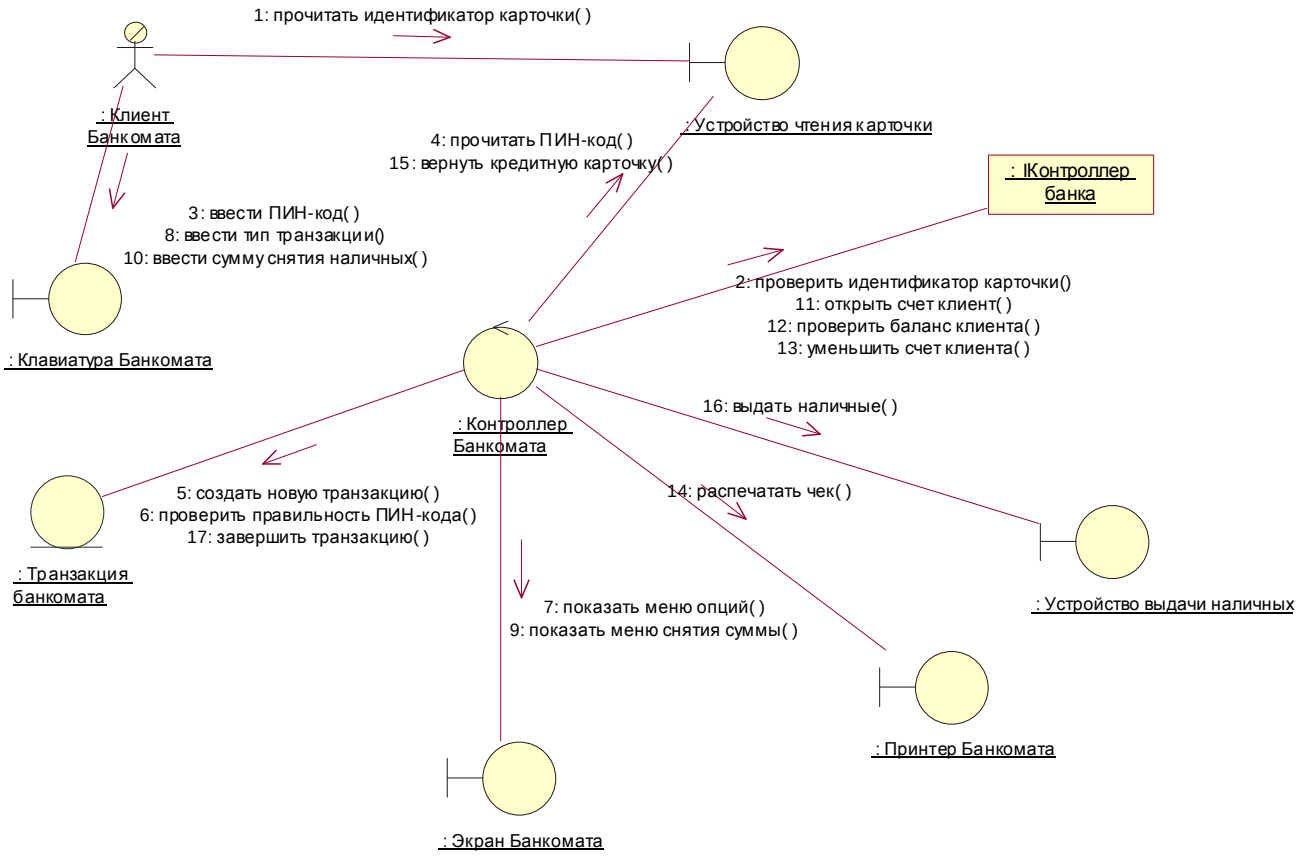

Рис. 21. Диаграмма кооперации в рамках варианта использования «Снятие наличных по кредитной карточке»

2.4 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Диаграмма классов для варианта использования «Снять деньги со счета» показана на **Рис. 22.**

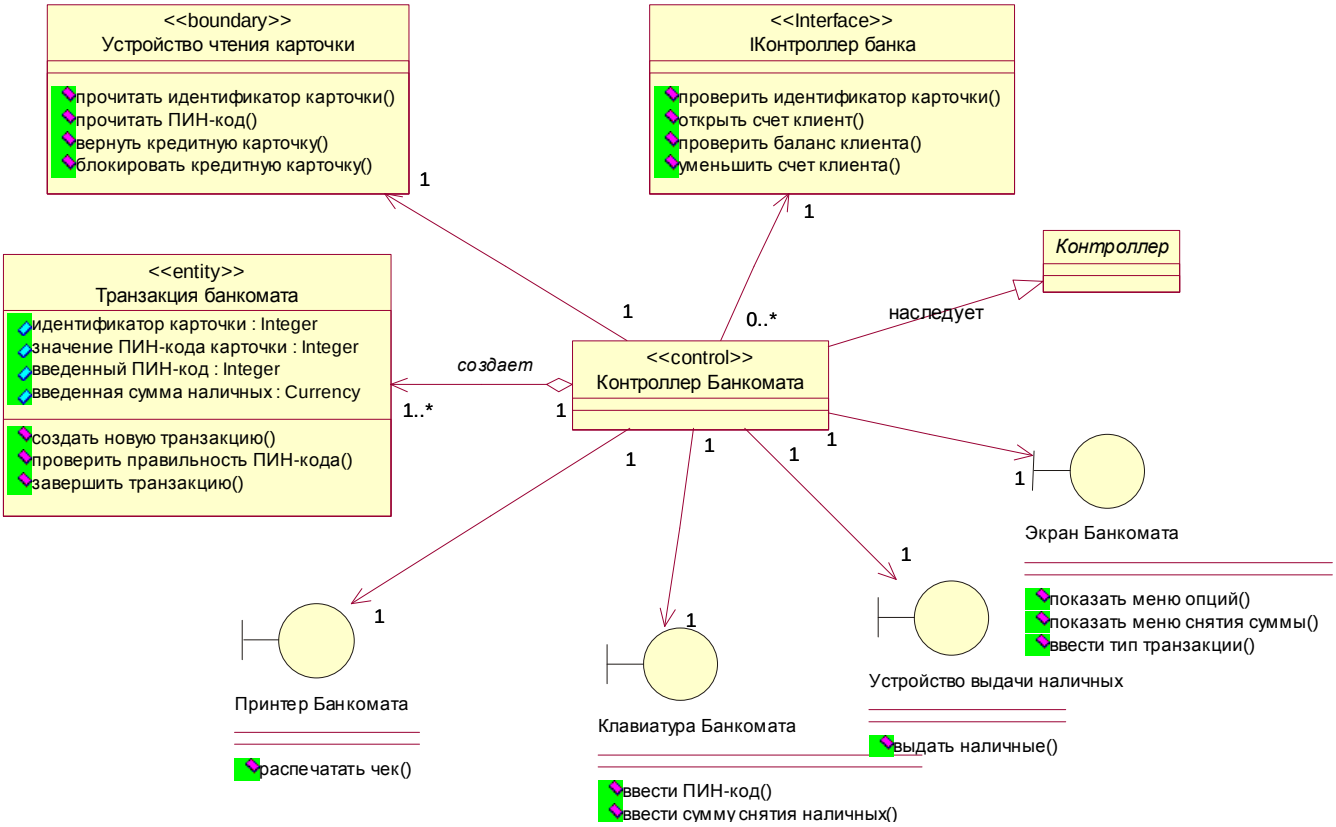

Рис. 22. Диаграмма классов для варианта использования
«Снятие наличных по кредитной карточке»

2.5 Диаграммы состояний

На **Рис. 23** приводится пример диаграммы состояний для банковского счета (account). Можно также наблюдать процесс перехода счета из одного состояния в другое. Например, если клиент требует закрыть счет, он переходит в состояние «закрыт», требование клиента называется событием (event), именно такие события и вызывают переход из одного состояния в другое. Если клиент снимает деньги со счета, он может перейти в состояние «Превышение кредита». Это происходит только в том случае, если баланс по счету меньше нуля, что отражено условием [отрицательный баланс] на нашей диаграмме. Заключенное в квадратных скобках условие (guard condition) определяет, когда может произойти переход из одного состояния в другое. На диаграмме имеются два специальных состояния - начальное (start) и конечное (stop). На диаграмме состояний может быть одно и только одно начальное состояние. В тоже время может быть столько конечных состояний, сколько вам к нужно, или их может не быть вообще. Когда объект находится в каком-то конкретном состоянии; могут выполняться различные процессы. В нашем примере при превышении кредита клиенту посылается соответствующее сообщение. Процессы, происходящие в этот момент, когда объект находится в определенном состоянии, называются действиями (actions).

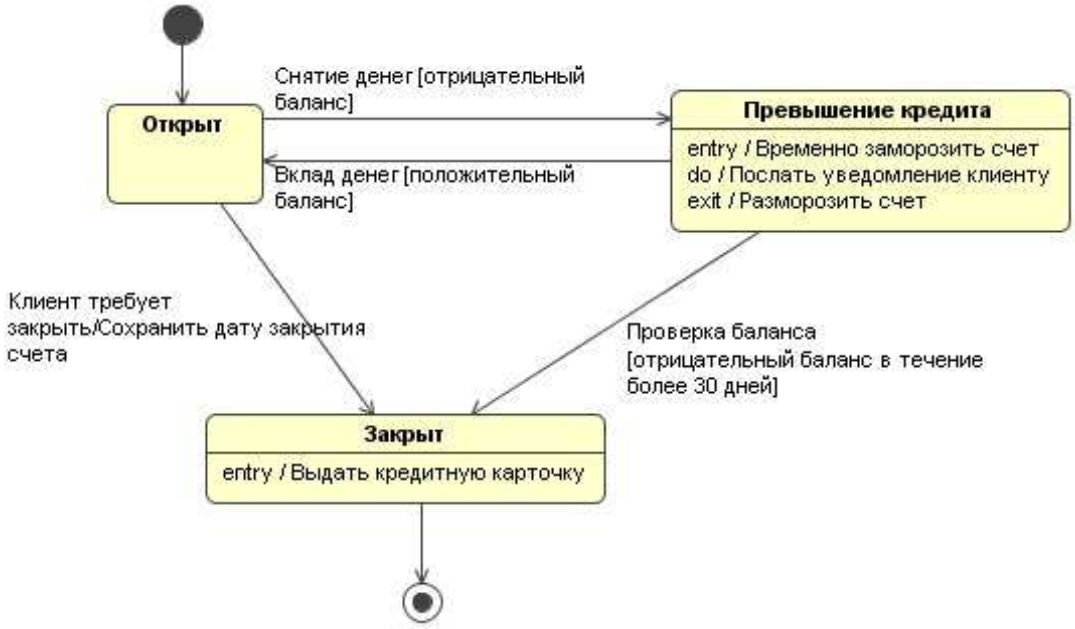

Рис. 23. Диаграмма состояний для класса Account (банковский счет)

На **Рис. 24** представлена диаграмма состояний для моделирования поведения банкомата

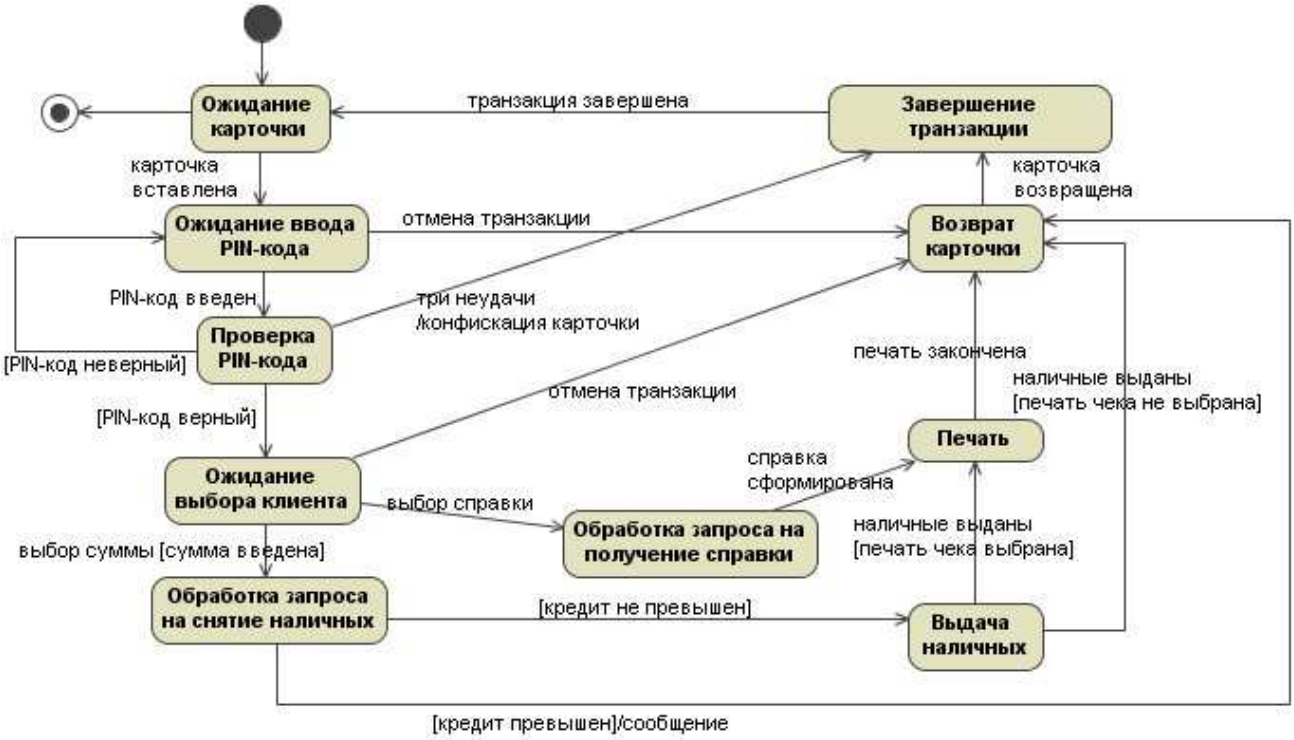

Рис. 24. Диаграмма состояний для моделирования поведения банкомата

2.6 Диаграммы деятельностей

Диаграммы деятельностей особенно полезны в описании поведения, включающего большое количество параллельных процессов. Самым большим достоинством диаграмм деятельностей является поддержка параллелизма. Самый большой их недостаток заключается в том, что связи между действиями и объектами просматриваются не слишком четко.

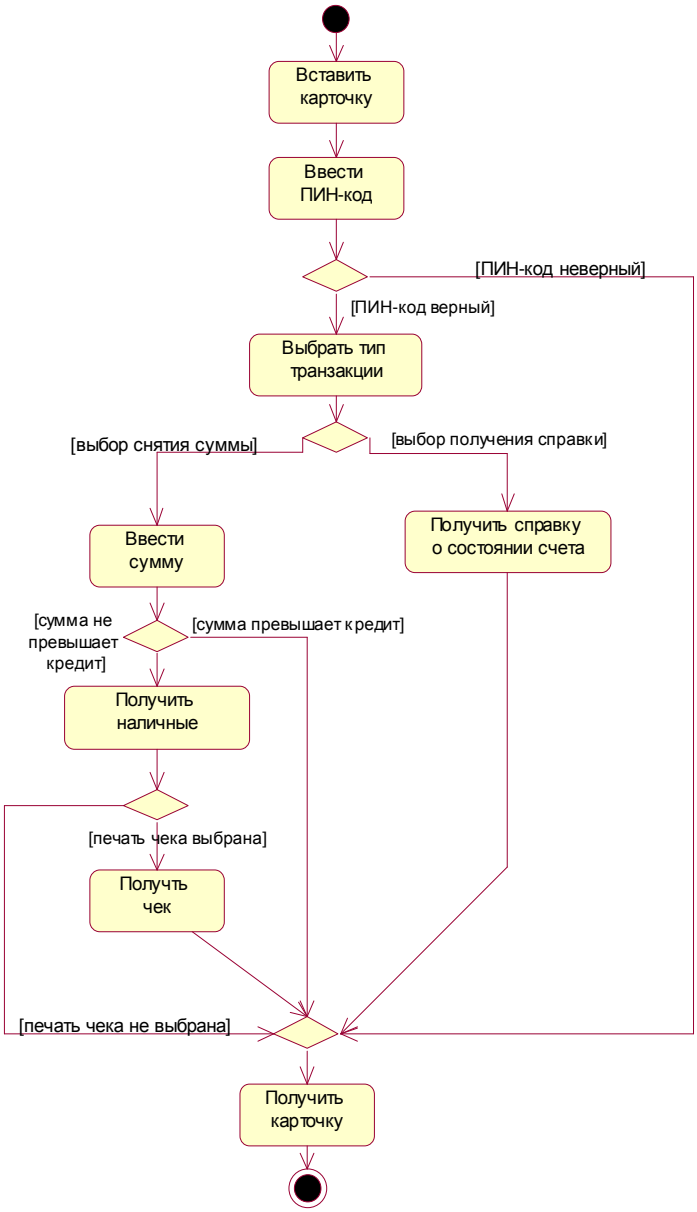

Рис. 25. Диаграмма деятельностей

2.7 Диаграмма компонентов

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы

У системы может быть несколько диаграмм компонентов в зависимости от числа подсистем или исполняемых файлов

На **Рис. 26** изображена одна из диаграмм компонентов для банковской системы.

На этой диаграмме показаны компоненты для клиентской части системы. В данном случае система разрабатывается на языке C++. У каждого класса имеется свой собственный заголовочный файл и файл с расширением. CPP, так что каждый класс преобразуется в свои собственные компоненты на диаграмме. Например, класс ATM Screen преобразуется в компонент ATM Screen диаграммы. Он преобразуется также и во второй компонент ATM Screen. Вместе эти два компонента представляют тело и заголовок класса ATM Screen. Полностью закрашенный компонент называется спецификацией пакета (package specification) и соответствует файлу тела класса ATM Screen на языке C++ (файл с расширением .CPP). Незакрашенный компонент также называется спецификацией пакета, но соответствует заголовочному файлу класса языка C++ (файл с расширением .H).

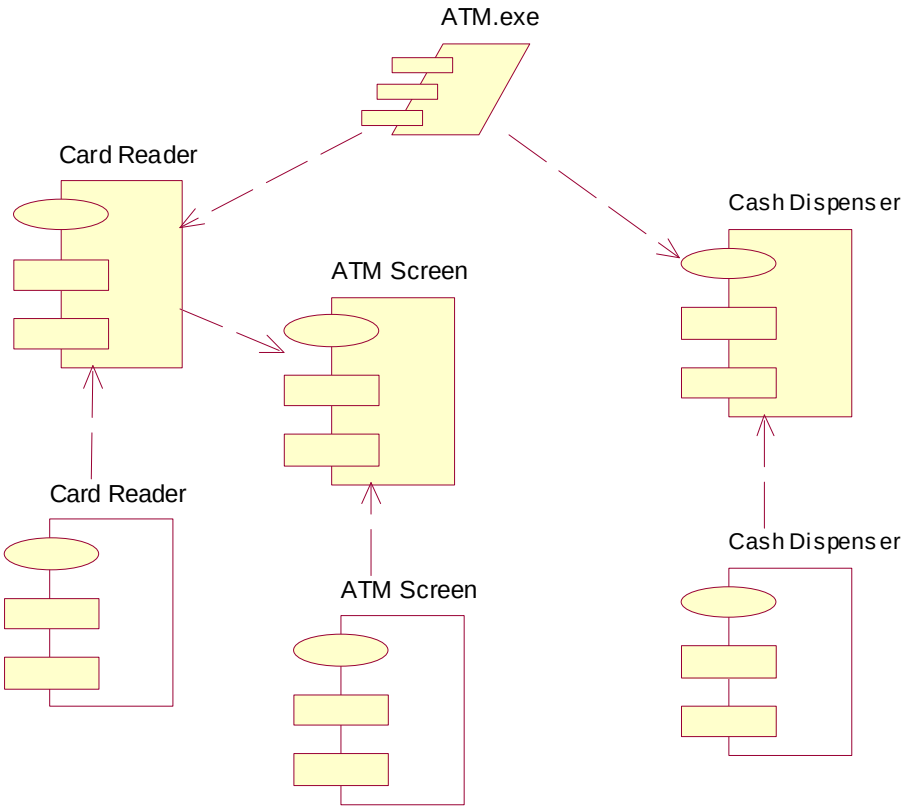

Рис. 26. Диаграмма компонентов для клиентской части системы

Компонент ATM.exe является спецификацией задачи и представляет поток обработки информации (thread of processing). В данном случае поток обработки является исполняемой программой.

Компоненты соединены штриховой линией, что соответствует зависимостям между ними. Например, класс Card Reader зависит от класса ATM Screen. Это означает, что для того, чтобы класс Card Reader мог быть скомпилирован, класс ATM Screen должен уже существовать. После компиляции всех классов может быть создан исполняемый файл ATMClient.exe.

Пример банковской системы содержит два потока обработки, и таким образом получаются два исполняемых файла. Один из них – это клиентская часть системы, она содержит компоненты Cash Dispenser, Card Reader и ATM Screen. Второй файл – это сервер, включающий в себя компонент Account. Диаграмма компонентов для сервера показана на **Рис. 27**.

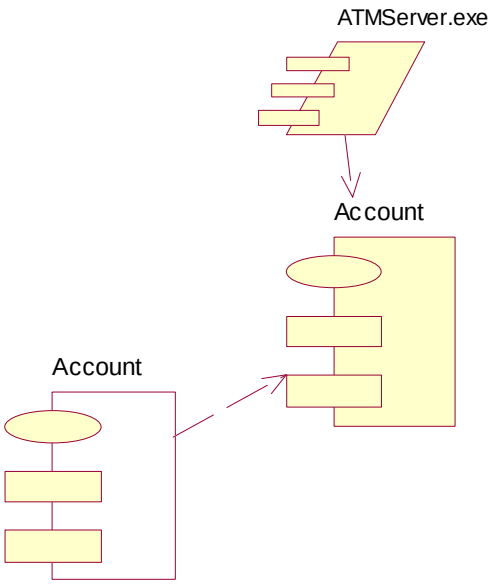

Рис. 27. Диаграмма компонентов для сервера

2.8 Диаграммы размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Она показывает размещение объектов и компонентов в распределенной системе.

Каждый узел на диаграмме размещения представляет собой некоторый тип вычислительного устройства - в большинстве случаев часть аппаратуры. Эта аппаратура может быть простым устройством или датчиком, а может быть и мэйнфреймом.

Диаграмма размещения показывает физическое расположение сети и местонахождение в ней различных компонентов. В нашем примере банковская система состоит из большого количества подсистем, выполняемых на отдельных физических устройствах, или узлах (node). Диаграмма размещения для банковской системы показана на **Рис. 28**.

Из данной диаграммы можно узнать о физическом размещении системы. Клиентские программы будут работать в нескольких местах на различных сайтах. Через закрытые сети будет осуществляться их сообщение с региональным сервером системы, с работающим программным обеспечением. В свою очередь, региональный сервер посредством локальной сети будет сообщаться, с сервером банковской базы данных, работающим под управлением Oracle. Наконец, с региональным сервером соединен принтер.

Диаграмма размещения используется менеджером проекта, пользователями, архитектором системы и эксплуатационным персоналом, чтобы понять физическое размещение системы и расположение ее отдельных подсистем.

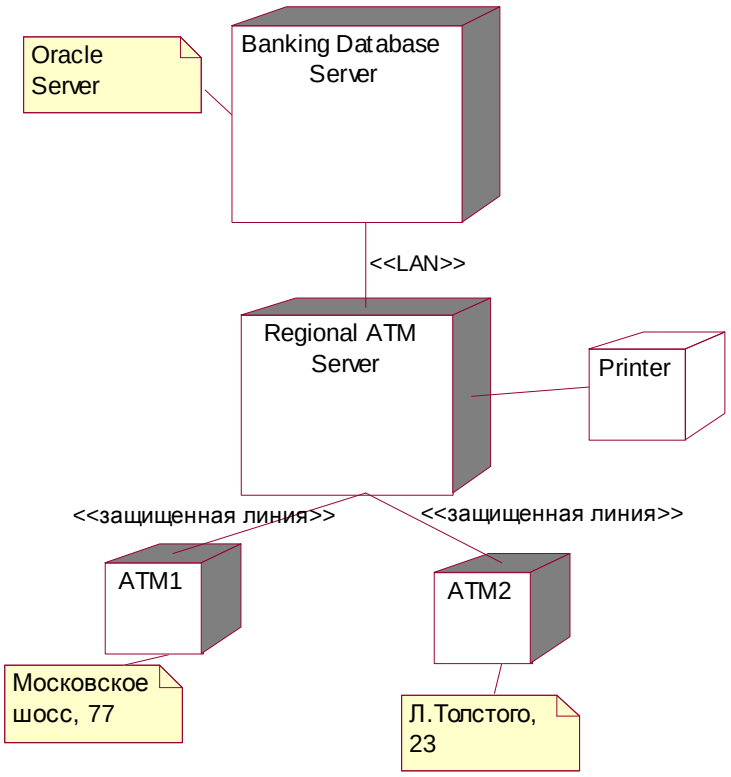

Рис. 28. Диаграмма размещения для банковской системы

3. СТРУКТУРА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ К КУРСОВОМУ ПРОЕКТУ

Курсовой проект выполняется по одному из вариантов, приведенных в следующем разделе. Номер варианта выбирается по сумме двух последних цифр зачетной книжки. Пояснительная записка должна отражать ход выполнения курсового проекта.

Пояснительная записка должна иметь следующую структуру:

Титульный лист
Лист для рецензии
Содержание
Задание с указанием номера варианта
Введение
1 Постановка задачи
2 Разработка проекта информационной системы с помощью объектно-ориентированного подхода (UML-диаграммы)
- 2.1. Диаграмма вариантов использования
- 2.2. Диаграмма классов
- 2.3. Диаграмма состояний
- 2.4. Диаграмма последовательности
- 2.5. Диаграмма кооперации
- 2.6. Диаграмма деятельностей
- 2.7. Диаграмма компонентов
- 2.8. Диаграмма размещения

3. Разработка пользовательского интерфейса
- 31. Описание функциональностей системы

3.2. Разработка экранных форм

Заключение

Список использованных источников

Введение. Введение должно содержать общие сведения о курсовом проекте. В нем необходимо отразить актуальность выбранной темы, цель и задачи, решаемые в проекте, объект (экономико-информационные процессы, происходящие на данном предприятии). Также необходимо дать обоснование необходимости автоматизации данной задачи (т.е. почему указанная задача должна быть автоматизирована?).

Объем введения должен быть не более 2 страниц.

1. Постановка задачи. В данном разделе необходимо провести анализ предметной области и осветить следующие пункты:

- наименование задачи, место ее решения;
- цель решения;
- назначение (для каких подразделений и пользователей предназначена);
- периодичность решения и требования к срокам решения;
- источники и способы поступления данных;
- потребители результатной информации и способы ее отправки;
- информационная связь с другими задачами;
- перечень исходной информации;
- перечень выходной информации;
- алгоритм решения задачи (показать последовательность действий информационной системы при обработке документов).

2. Разработка проекта ИС с помощью объектно-ориентированного подхода. В данном разделе разрабатываются UML-диаграмм. Пример UML-диаграмм приведен в предыдущем разделе.

3. Разработка пользовательского интерфейса. При описании функциональностей системы необходимо привести список пользователей системы и указать те задачи, решение которых осуществляется с помощью разработанной информационной системы. Также в данном разделе пояснительной записки необходимо привести примеры экранных форм (интерфейс) АРМ каждой группы пользователей.

4. ВАРИАНТЫ ЗАДАНИЙ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

Вариант 1 – Деканат

Задача – информационная поддержка деятельности деканата вуза.

- ведение расписания сессии, хранение результатов сессии;
- составление зачётных и экзаменационных ведомостей;
- составление расписаний экзаменов по группам, кафедрам, для отдельных преподавателей;
- проверка корректности расписания экзаменов (уникальность комбинации "время – дата – аудитория"; между экзаменами в одной группе должно пройти не менее трёх дней);
- подсчёт по результатам зачётов и экзаменов итоговых значений (количество оценок '5', '4', '3', '2', количество неявок, средний балл по группе);
- получение списка экзаменов на текущую дату.

Вариант 2 – Книжный магазин

Необходимо построить базу данных, располагая которой пользователь может получить справочную информацию о работе книжного магазина:

- список книг, продаваемых в данном книжном магазине;

– количество экземпляров книги;
– жанры книг (поэзия, проза, фантастика, учебная литература и т.д.);
– перечень издательств, поставляющих книги данному магазину;

Пользователю на основе данных из базы данных необходимо:

– сформировать выходной документ "Величина продаж по каждому жанру книг";
– определить самую продаваемую книгу;
– перечень издательств, специализирующихся на конкретных жанрах;
– определить расчетным путем общую выручку магазина за отчетный период и т.д.

Вариант 3 – Расписание движения поездов

Система должна хранить информацию о проходящих через станцию Самара поездах и выдавать информацию по номеру поезда и пункту назначения (дата прибытия, дата отправления). Учесть, что некоторые поезда находятся в движении только в определенные периоды, например, летом и в определенные дни.

Система должна также выдавать сведения о поездах, на которых можно без пересадок доехать до указанной станции (номер поезда, день и время прибытия и отправления, время в пути до указанной станции, день и время прибытия на станцию назначения, стоимость проезда в вагонах различных категорий, стоимость постельного белья).

Вариант 4 – Кафедра

Задача – информационная поддержка учебного процесса и организационной деятельности на кафедре вуза. БД должна содержать учебный план, расписание занятий, списки групп, выпускаемых кафедрой, и списки аспирантов (с руководителями и темами исследований). БД должна обеспечивать составление:

– расписания занятий на семестр (по группам);
– учебного плана (по семестрам) для каждого курса;
– расписания занятий для преподавателей;
– списка телефонов сотрудников;
– нагрузки по часам для преподавателей;
– списка научных кадров по научным направлениям;
– списков студентов-дипломников (по группам и по преподавателям).

Вариант 5 – Библиотека

Задача – информационная поддержка деятельности научно-технической библиотеки.

БД должна включать два раздела: "Научная литература" и "Журнальные публикации". БД должна обеспечивать:

– ведение автоматизированного учёта выдачи/приёма литературы;
– ведение очередей на литературу (по заказам);
– учёт рейтинга изданий (количество читателей и дата последней выдачи);
– поиск литературы по требуемым разделу, теме, автору, ключевому слову (с заданием интересующего периода);
– составление списков должников по годам.

Вариант 6 – Больница

Задача – информационная поддержка деятельности регистратуры больницы. БД должна осуществлять:

– учёт поступления пациентов (по отделениям);
– учёт проведённого лечения;
– учёт платных услуг с выдачей счетов на оплату;
– ведение архива выписанных пациентов.

Необходимо предусмотреть определение (по отделениям):

– пропускной способности больницы;
– среднего времени пребывания больных в стационаре;
– наличия свободных мест в палатах (отдельно для мужчин и для женщин);
– количества прооперированных пациентов (из них – с осложнениями и умерших);
– смертности.

Вариант 7 – Магазин (выбрать конкретный профиль)

Задача – информационная поддержка деятельности магазина выбранного профиля. БД должна осуществлять:

– учёт поставщиков и поставок;
– учёт продаж по отделам;
– подсчёт остатков товаров (по отделам);
– оформление заказов на товары, запасы которых подходят к концу;
– подведение финансовых итогов дня (по отделам и в целом по магазину);
– анализ результативности работы продавцов (для премирования);
– анализ объёмов продаж по дням недели и по месяцам.

Вариант 8 – БД адвоката

Задача – информационная поддержка деятельности адвокатской конторы. БД должна осуществлять:

– ведение списка адвокатов;
– ведение списка клиентов;
– ведение архива законченных дел.

Необходимо предусмотреть:

– получение списка текущих клиентов для конкретного адвоката;
– определение эффективности защиты (максимальный срок минус полученный срок) с учётом оправданий, условных сроков и штрафов;
– определение неэффективности защиты (полученный срок минус минимальный срок);
– подсчёт суммы гонораров (по отдельных делам) в текущем году;
– получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

Вариант 9 – БД риэлторской компании

Задача – информационная поддержка деятельности фирмы, занимающейся продажей и арендой жилых и нежилых помещений. БД должна:

– осуществлять ведение списков жилых и нежилых помещений, предназначенных для аренды и/или продажи;
– поддерживать архив проданных и сданных в аренду помещений;
– производить поиск вариантов в соответствии с требованиями клиента.

Необходимо предусмотреть получение разнообразной статистики:

– наличие помещений разных типов;
– изменение цен на рынке;
– уровни спроса и предложения;
– средние показатели (среднее время нахождения помещения в БД (по типам помещений), среднюю стоимость аренды/продажи помещений и т.п.

Вариант 10. – Отель

Создаваемая информационная система предназначена для администрации отеля, которая на основании информации о номерах занимается размещением клиентов в соответствии с их запросами. При выбытии клиента информация о номере, в котором он проживал, должна

обновляться, а информация о клиенте должна удаляться из рабочих таблиц (карточки регистрации клиентов и карточки учета) и помещаться в архивную таблицу.

БД должна осуществлять:

- ведение списка постояльцев;
- учёт забронированных мест (с учетом класса комнаты);
- ведение архива выбывших постояльцев за последний год.

Необходимо предусмотреть:

- получение списка свободных номеров (по количеству мест и классу);
- получение списка номеров (мест), освобождающихся сегодня и завтра;
- выдачу информации по конкретному номеру;
- автоматизацию выдачи счетов на оплату номера и услуг;
- получение списка забронированных номеров;
- проверку наличия брони по имени клиента и/или названию организации.

Вариант 11 – Справочная служба аптек

Информационная система предназначена как информационной поддержки работы фармацевтов и для клиентов.

В системе должен вестись учет лекарственных препаратов (цена, наличие как в самой аптеке, так и на складе). Для каждого препарата должны быть указаны: состав, фармакотерапевтическая группа, условия отпуска. Также для лекарственных препаратов должна указываться форма выпуска (таблетки, настойка, мазь, гель и т.п.), условия хранения, срок годности.

Система должна:

- отслеживать выдачу лекарственных препаратов по рецепту;
- отслеживать поступление лекарственных препаратов с указанием цены, наименования производителей;
- выдавать отчет о проданных лекарствах за отчетный период для ведения статистического анализа.

Вариант 12. – Спортивный клуб

Информационная система содержит информацию о деятельности спортивного клуба. БД должна осуществлять:

- ведение списков спортсменов и тренеров;
- учёт проводимых соревнований (с ведением их архива);
- учёт травм, полученных спортсменами.

Необходимо предусмотреть:

- возможность перехода спортсмена от одного тренера к другому;
- составление рейтингов спортсменов;
- составление рейтингов тренеров;
- выдачу информации по соревнованиям;
- выдачу информации по конкретному спортсмену;
- подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

Вариант 13 – Отдел кадров ВУЗа

Задача – информационная поддержка деятельности отдела кадров.

Различают три группы сотрудников: а) администрация; б) преподавательский и инженерно-технический состав (по кафедрам); в) технический персонал. БД должна содержать штатное расписание по отделам (кафедрам) с указанием количества ставок по должностям, включать архив сотрудников и учитывать сотрудников, находящихся в отпуске по уходу за ребенком.

БД должна предоставлять возможность составления должностных (штатных) расписаний по кафедрам и отделам и следующих списков:

- вакансий (с учётом сотрудников, находящихся в отпуске по уходу за ребенком, т.е. с указанием даты, до которой ставка свободна);
- пенсионеров;
- людей предпенсионного возраста (не более 2-х лет до пенсии);
- бездетных сотрудников;
- юбиляров текущего года;
- многодетных сотрудников (трое и более детей);
- ветеранов (работающих в институте не менее тридцати лет);
- сотрудников, работающих более чем на одной ставке.

Вариант 14 – Санаторий

Система предназначена для администрации санатория. Система должна содержать информацию о клиентах санатория (как прошлых, так и настоящих): ФИО, дату приезда, дату выезда, процедуры, длительность курсов лечения, курирующем враче.

Также система должна содержать информацию о стоимости процедур, длительности курсов лечения.

Предусмотреть скидки постоянным клиентам, а также сезонные скидки.

Вариант 15 – Подписка

Система предназначена для городского отделения связи и должна позволять:

- получать сведения о каждом издании (название, индекс, вид, периодичность, цена подписки с учетом доставки);
- для еженедельников и журналов хранить даты поступления всех
- номеров;
- находить сведения о всех подписчиках любого издания (выдавать доставочные листы);
- выдавать сводную информацию о количестве комплектов каждого
- выписанного издания;
- находить сведения о подписке для конкретного подписчика.

Вариант 16 – Стоматологическая поликлиника

Поликлиника ведет прием и учет пациентов, учет их посещений (визитов) и учет обслуживания пациентов специалистами (врачами) поликлиники. Существует необходимость в хранении информации обо всех посещениях поликлиники пациентами и о том, на приеме у каких специалистов они находились. Обслуживание пациентов ведется по предварительной записи.

Необходимо обеспечить ввод, хранение и, возможно, редактирование данных. В определенных случаях необходимо выполнять удаление данных. Например, можно удалить информацию обо всех визитах некоторого пациента, если после его последнего визита прошел определенный срок (например, 3 года), а данные о самом пациенте перенести в архив (или также удалить).

Также система должна содержать график работы каждого врача, его специализацию и категорию. Необходимо предусмотреть поиск сведений о пациентах как по фамилии, так и по номеру истории болезни. За каждое посещение пациенту выписывается счет, который он должен оплатить.

Кроме задач, перечисленных выше, информационная система должна решать следующие задачи:

- подсчет выручки каждого специалиста за определенный период (день, месяц, квартал);

- подсчет выручки поликлиники в целом за определенный период (день, месяц, квартал);
- подсчет оплаченной суммы за лекарства за определенный период (день, месяц, квартал);
- подсчет количества посещений поликлиники за месяц в целом и по каждой группе специалистов.

Вариант 17 – Ателье мод

Ателье мод выполняет заказы клиентов на индивидуальный пошив одежды. В ателье существует каталог моделей и каталог тканей. По каталогу моделей клиент выбирает модель, а по каталогу тканей – ткань, из которой будет выполнена модель, и заказывает ее пошив в ателье.

Заказ каждого клиента содержит: Ф.И.О. клиента, информацию о модели, информацию о ткани, Ф.И.О. закройщика (исполнителя заказа), дату приема заказа, дату примерки, отметку о выполнении заказа, дату выполнения заказа.

В каталоге моделей каждая модель имеет уникальный номер, для каждой модели указывается рекомендуемая ткань, необходимый расход ткани для данной модели с учетом ширины ткани, цена готовой модели, включающая цену ткани и стоимость пошива изделия.

В каталоге тканей каждая ткань имеет уникальный номер, название, а также указываются ее ширина и цена за 1 метр.

В ателье есть еще и склад тканей. В книге учета тканей на складе для каждой ткани указывается общий метраж, который изменяется, если принимается заказ на изготовление модели из данной ткани.

Вариант 18 – Оптовый склад

Склад осуществляет продажу товаров оптом. Любая фирма, занимающаяся продажей товаров в розницу, закупает необходимые ей товары на складе, который служит посредником между производителями и продавцами.

На склад товар поступает от некоторой фирмы-поставщика, в свою очередь склад продает товар фирме-покупателю, заключая с ним сделку о продаже товара.

Необходимо вести учет поставщиков, покупателей, продаж, движения товара на складе. Кроме того, можно делать выводы о работе склада, спросе на определенные товары, выгодности работы с некоторыми поставщиками и покупателями.

Вариант 19 – Торгово-закупочное предприятие

Торгово-закупочное предприятие имеет склад, содержащий определенные виды товаров, например, продовольственные товары. Предприятие имеет штат сотрудников, являющихся агентами-реализаторами. Предприятие выдает агенту товар, устанавливая цену его продажи. Агент-реализатор оплачивает выданный товар не сразу, а по мере его реализации, оформляя приходные кассовые ордера. С каждой единицы проданного товара агент получает оплату, установленную предприятием.

Необходимо вести учет движения товаров как на складе, так и у агентов-реализаторов. Кроме того, предприятие проводит операции: по новым поступлениям товара, по выдаче товара агенту, по расчету с агентом за реализованный товар.

Вариант 20 – Автосалон

Существует фирма, торгующая автомобилями. Автомобиль выступает в качестве товара и как товар имеет определенные характеристики. Кроме того, на каждый автомобиль имеются исчерпывающие технические данные. Фирма имеет своих клиентов – покупателей автомобилей, сведения о которых хранит.

Необходимо обеспечить ввод, редактирование и просмотр данных в удобной для пользователя форме.

Предполагается также решение следующих задач:

- выдать информацию о наличии автомобилей определенной марки и модели;
- выдать технические данные заданной модели;
- выдать информацию обо всех проданных моделях некоторой марки, значение которой вводится в качестве параметра;
- посчитать сумму продаж моделей каждой марки и общую сумму продаж;
- выдать полную или частичную информацию о клиентах фирмы;
- выдать списки клиентов и автомобилей по виду оплаты.

Возможны постановка и решение других задач.

Вариант 21 – Продажа подержанных автомобилей

Фирма по продаже подержанных автомобилей работает с физическими лицами – клиентами фирмы, имеющими подержанный автомобиль или автомобили и желающими продать их через фирму. Непосредственной продажей автомобилей занимаются сотрудники фирмы – дилеры. На каждый предлагаемый в продажу автомобиль фирма заключает с клиентом договор, содержащий данные о клиенте, необходимые сведения об автомобиле, а также данные о дилере, обслуживающем этот договор.

В создаваемой информационной системе необходимо обеспечить ввод и редактирование данных. Кроме того, необходимо выдавать информацию о клиентах и предлагаемых ими автомобилях, а также информацию о деятельности дилеров (с перечислением договоров) и клиентах, которые они обслуживают. Могут быть выполнены разнообразные запросы, например:

- посчитать количество договоров, заключенных с каждым клиентом;
- посчитать количество договоров, обслуживаемых каждым дилером;
- выдать некоторую информацию (например: данные дилера, дата заключения договора, данные клиента, отметка о продаже) обо всех договорах, договорах за некоторый промежуток времени или договорах, удовлетворяющих определенному условию.

Вариант 22 – Такси

Система предназначена для автоматизации деятельности компании такси.

Звонки от клиентов поступают в службу такси и принимаются диспетчером, обладающим данными о наличии свободных машин в различных районах города. Диспетчер на основе имеющейся информации сообщает клиенту примерное время ожидания машины и, если возможно оперативно узнать расстояние перевозки, тариф на данное направление.

В системе необходимо осуществить прием заказа от клиента, назначить водителя на конкретный заказ. Также в системе должно быть предусмотрено ведение сведений о имеющемся парке автомобилей

Вариант 23 – Пассажирское автопредприятие

Муниципальное автопредприятие осуществляет пассажирские перевозки на внутригородских маршрутах. Автопредприятие имеет парк автобусов, которые работают на определенных маршрутах. Работу автопредприятия обеспечивает персонал предприятия, который можно разделить по категориям занимаемых должностей надминистрацию, инженерно-технический персонал и персонал, обслуживающий маршруты (водители, кондукторы). Выезжая на маршрут, водитель автобуса получает маршрутный лист (или путевой лист), содержащий данные об автобусе, маршруте, режиме работы, водителе, кондукторе.

В учетных данных персонала должны вестись данные для оплаты труда, если предполагается автоматизация начисления зарплаты. В маршрутных листах можно ввести плановую и фактическую выручки за смену соответственно.

БД должна обеспечивать выполнение запросов:

- выдать список сотрудников администрации с указанием должности;

- на определенную дату для всех номеров маршрутов выдать информацию о количестве автобусов, обслуживающих каждый маршрут;
- по каждому номеру маршрута и дате (параметры запроса) выдать информацию об автобусах, обслуживающих маршрут: бортовой номер, марка, гос. номер автобуса;
- по итогам работы за месяц посчитать количество рейсов, выполненных каждым автобусом или на каждом маршруте;
- по итогам работы за месяц посчитать количество смен, отработанных каждым водителем и кондуктором.

Вариант 24 – Междугородные пассажирские перевозки

Рассмотрим автовокзал, который занимается обслуживанием и учетом пассажиров на междугородных автобусных маршрутах. На автовокзале имеется расписание движения автобусов, содержащее информацию о маршрутах и рейсах. Кроме того, на автовокзале имеется справочное бюро, в котором можно получить информацию о наличии мест на определенный рейс конкретной даты. И, наконец, на автовокзале есть кассы, в которых пассажир может приобрести билет. Кассы начинают предварительную продажу билетов за определенный промежуток времени до дня отправления автобуса (например, за 10 дней).

Необходимо построить такую базу данных, в которой хранится информация как о технических характеристиках маршрутов, содержащаяся в расписании, так и информация о наличии мест на рейсы, и информация о пассажирах, купивших билеты на определенный рейс.

Также в система должна выдавать ответы на следующие запросы:

- наличие свободных мест на рейс;
- продажу билетов в оба конца;
- поиск места на рейс в соответствии с требованиями заказчика;
- количество пассажиров уже выполненного рейса, доходность рейса;
- список всех пассажиров определенного рейса (выполненного или того, на который идет продажа билетов);
- определить, покупал ли билет человек с заданной фамилией и, если покупал, то на какой рейс;
- определение количества перевезенных пассажиров и объем перевозок (в денежном выражении) по дням, по месяцам в целом по всем направлениям или по определенному маршруту.

Вариант 25 – Агентство по продаже авиабилетов

Агентство занимается продажей авиабилетов на различные рейсы, ведет учет проданных билетов и учет пассажиров, купивших билеты. Особенность данной задачи состоит в том, что информация в базе данных может использоваться как пассажирами (например, для получения сведений о расписании и наличии свободных мест на рейс), так и служащими агентства: кассирами и диспетчерами (администраторами).

В системе должен осуществляться поиск следующей информации:

- расписание рейсов (номер рейса, тип самолета, пункт отправления, пункт назначения, дата вылета, время вылета, время полета, цена билета);
- информация о свободных местах на рейс (номер рейса, дата вылета, общее количество мест, количество свободных мест);
- информация о пассажирах, заказавших билет (фамилия, имя, отчество, предъявленный документ, его серия и номер, номер рейса, дата вылета).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вендров А.М. Практикум по проектирование программного обеспечения экономических информационных систем: Учеб. пособие для вузов.- М.: Финансы и статистика, 2002.- 192 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем – М.: Финансы и статистика, 2005.- 544 с.
3. Леоненков А.В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. – Издательства: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2006. – 320 с.: ил.
4. Якобсон А. Унифицированный процесс разработки программного обеспечения/ А. Якобсон, Г. Буч, Дж. Рамбо; пер. с англ. В. Горбункова.- СПб.: Питер, 2002.- 496 с.: ил