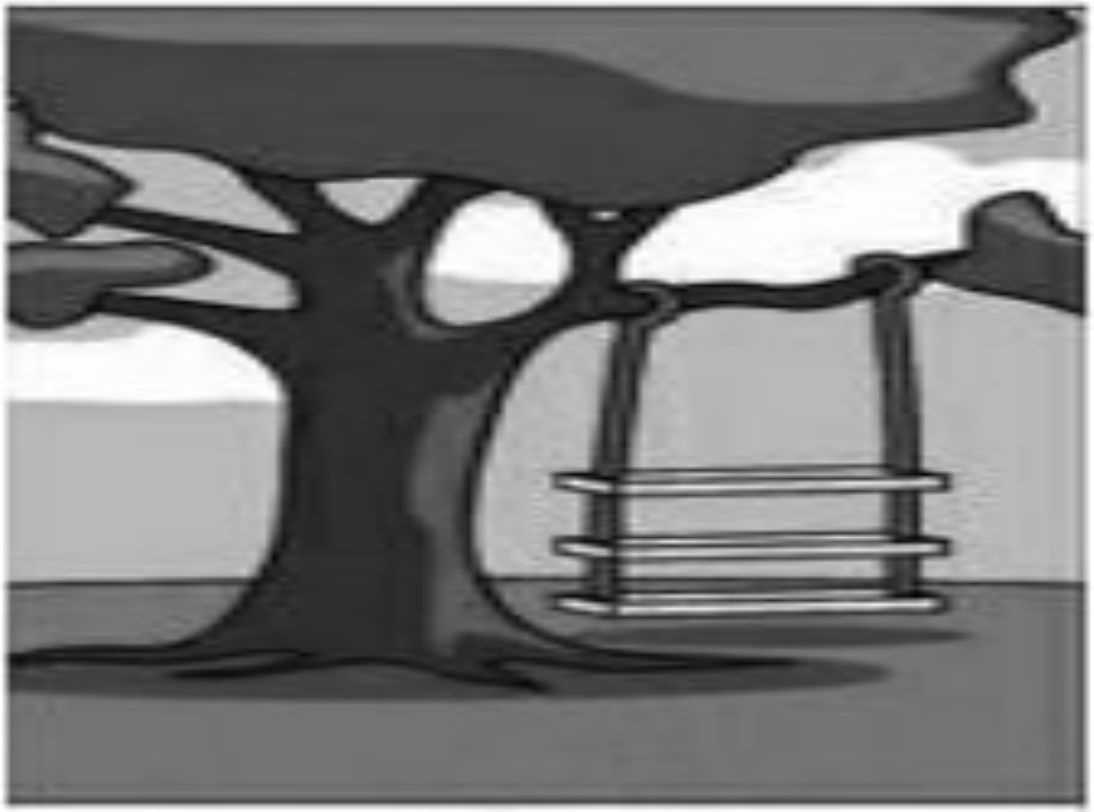
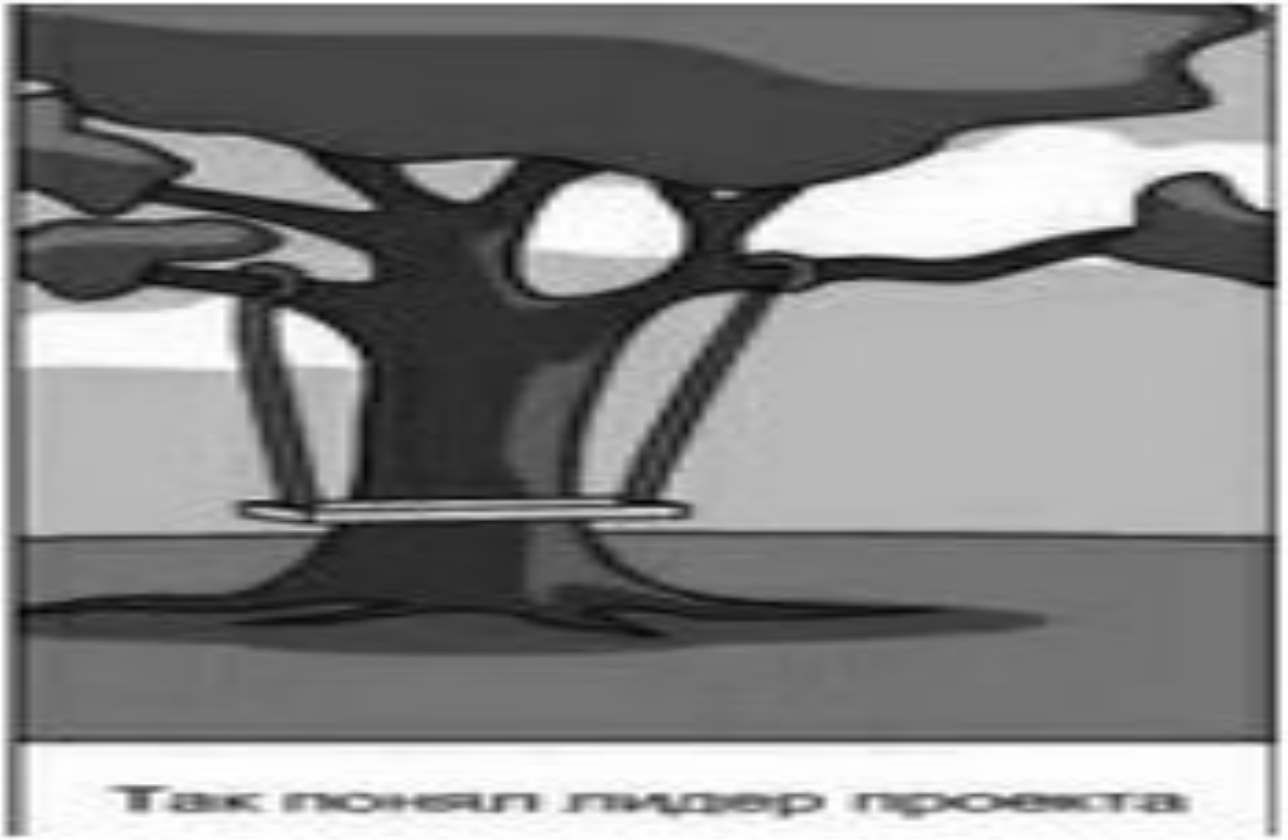
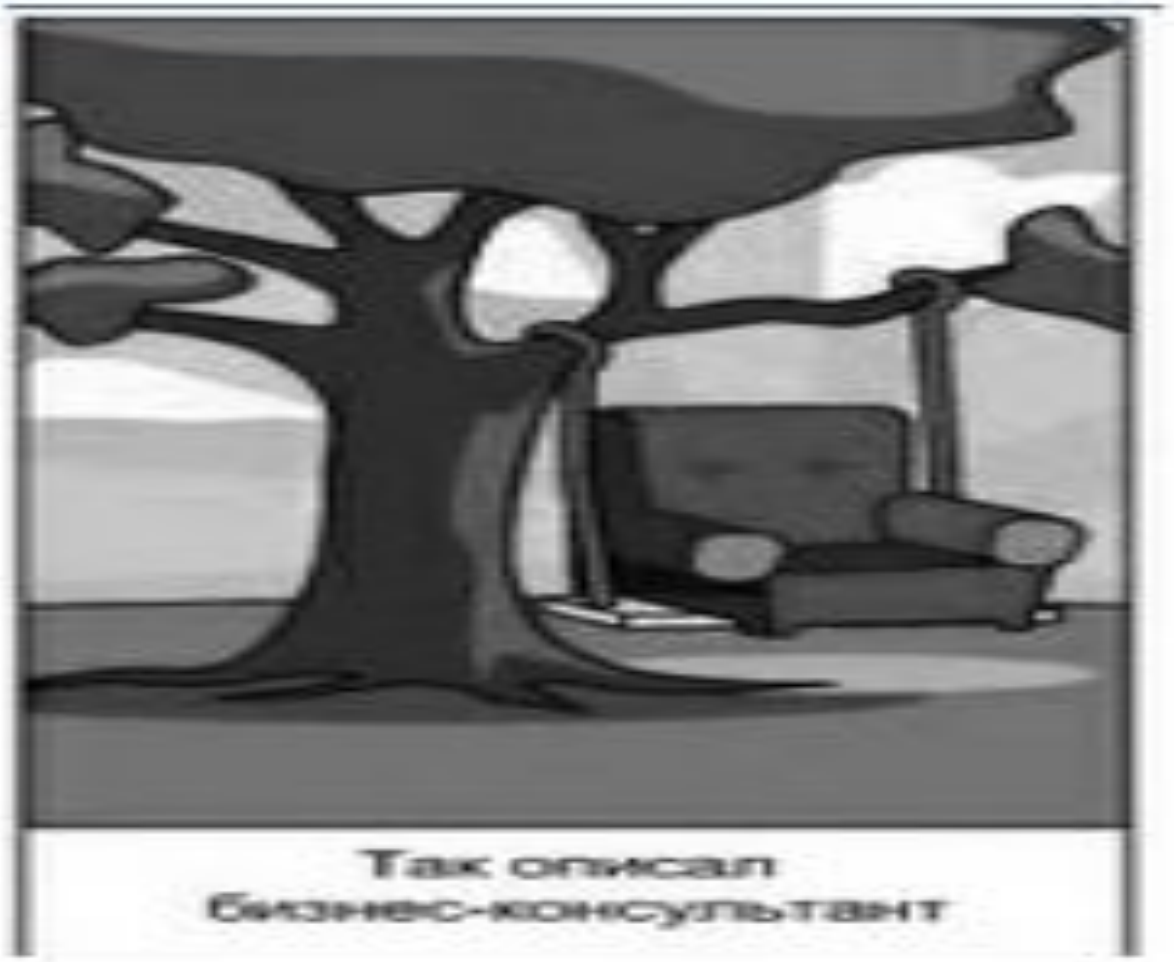
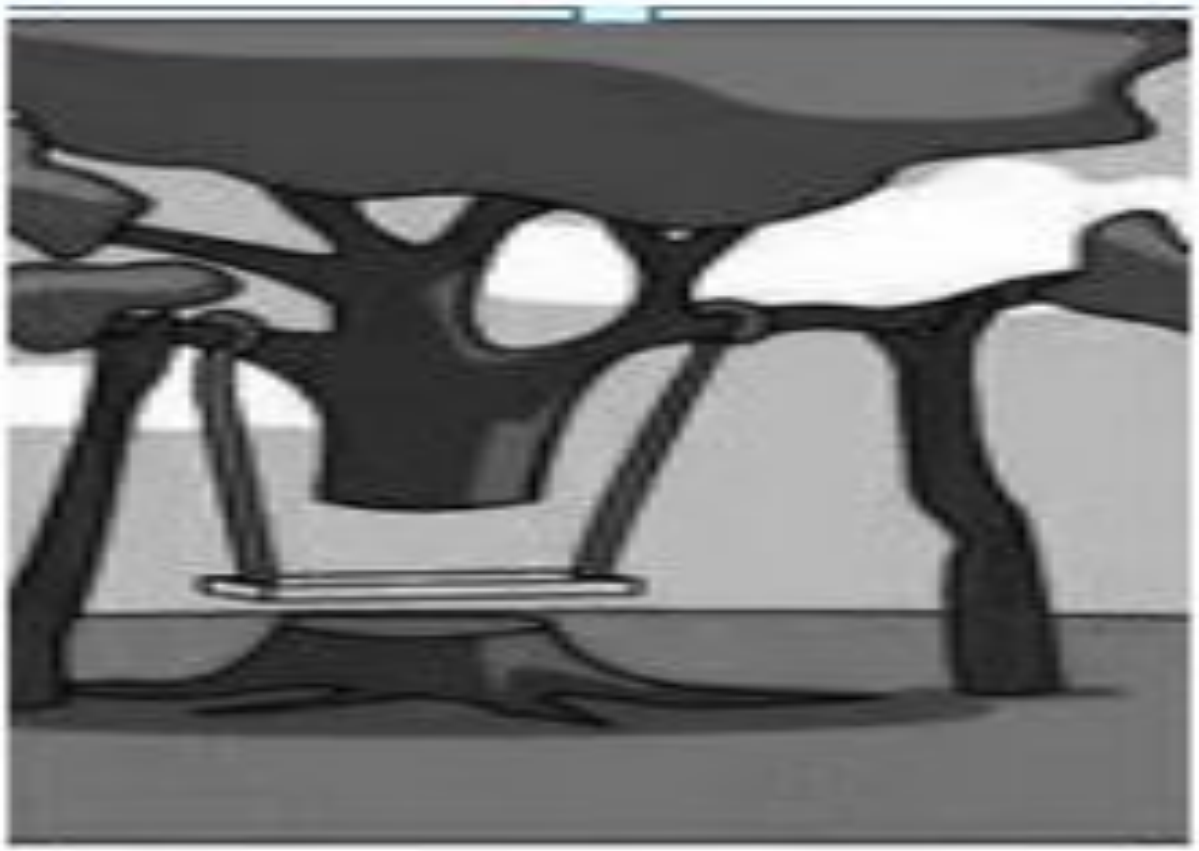
УНИФИЦИРОВАННЫЙ ЯЗЫК ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ (UML)



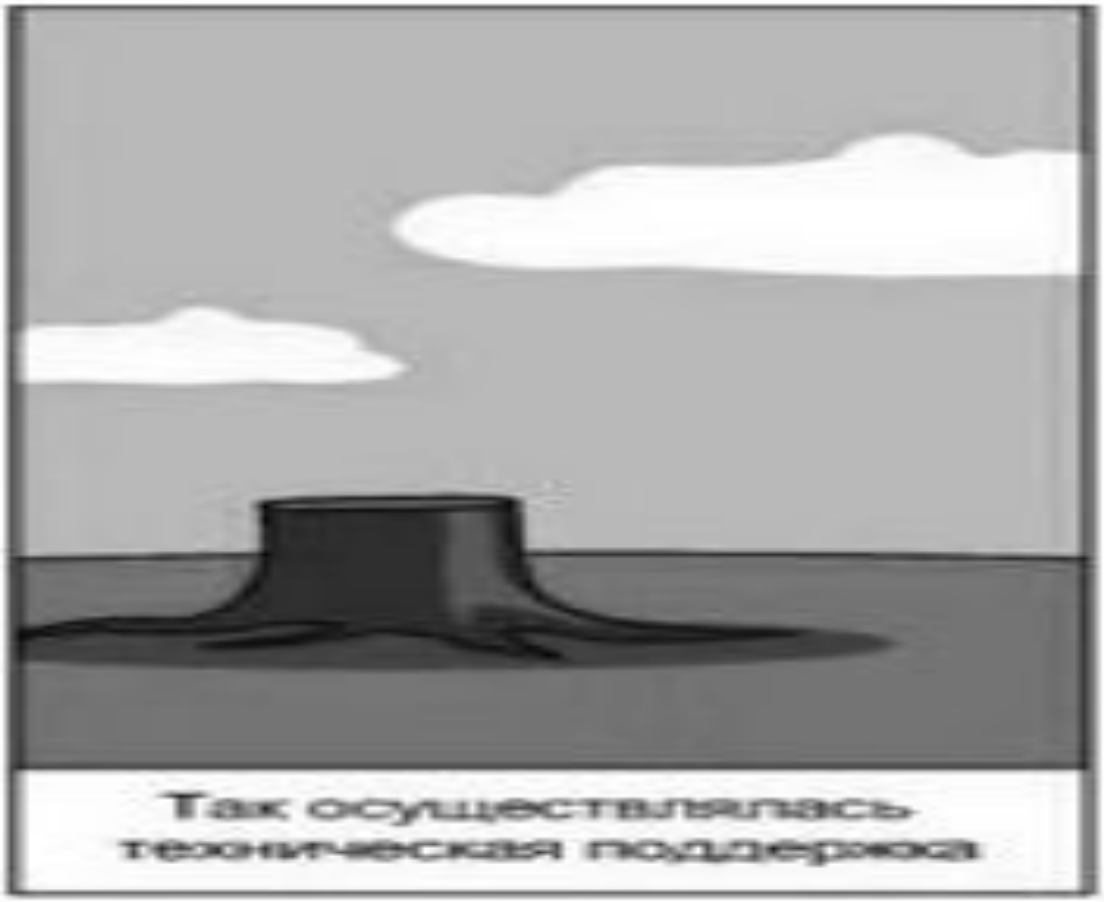
Так объяснил заказчик

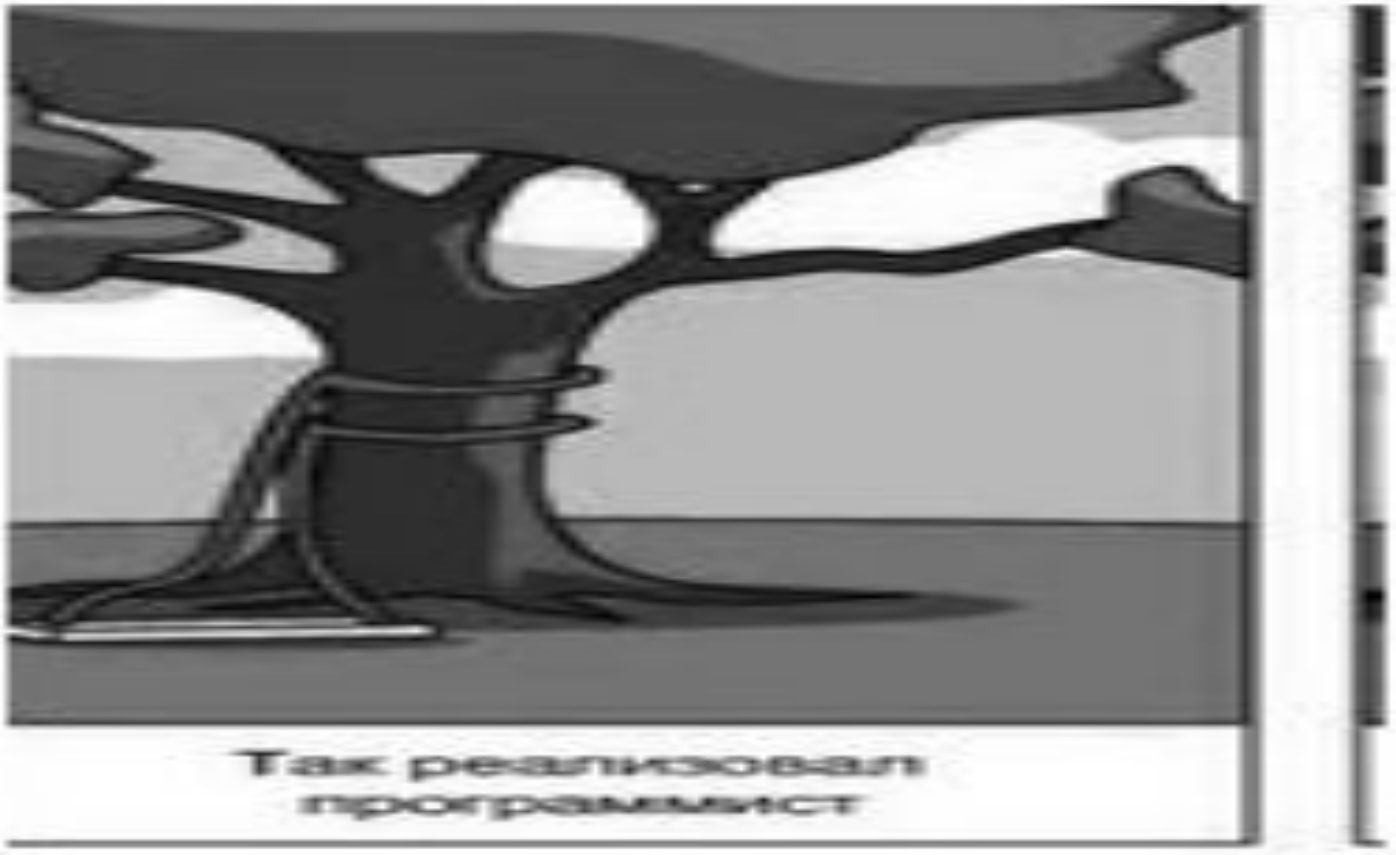

Так понял лидер проекта

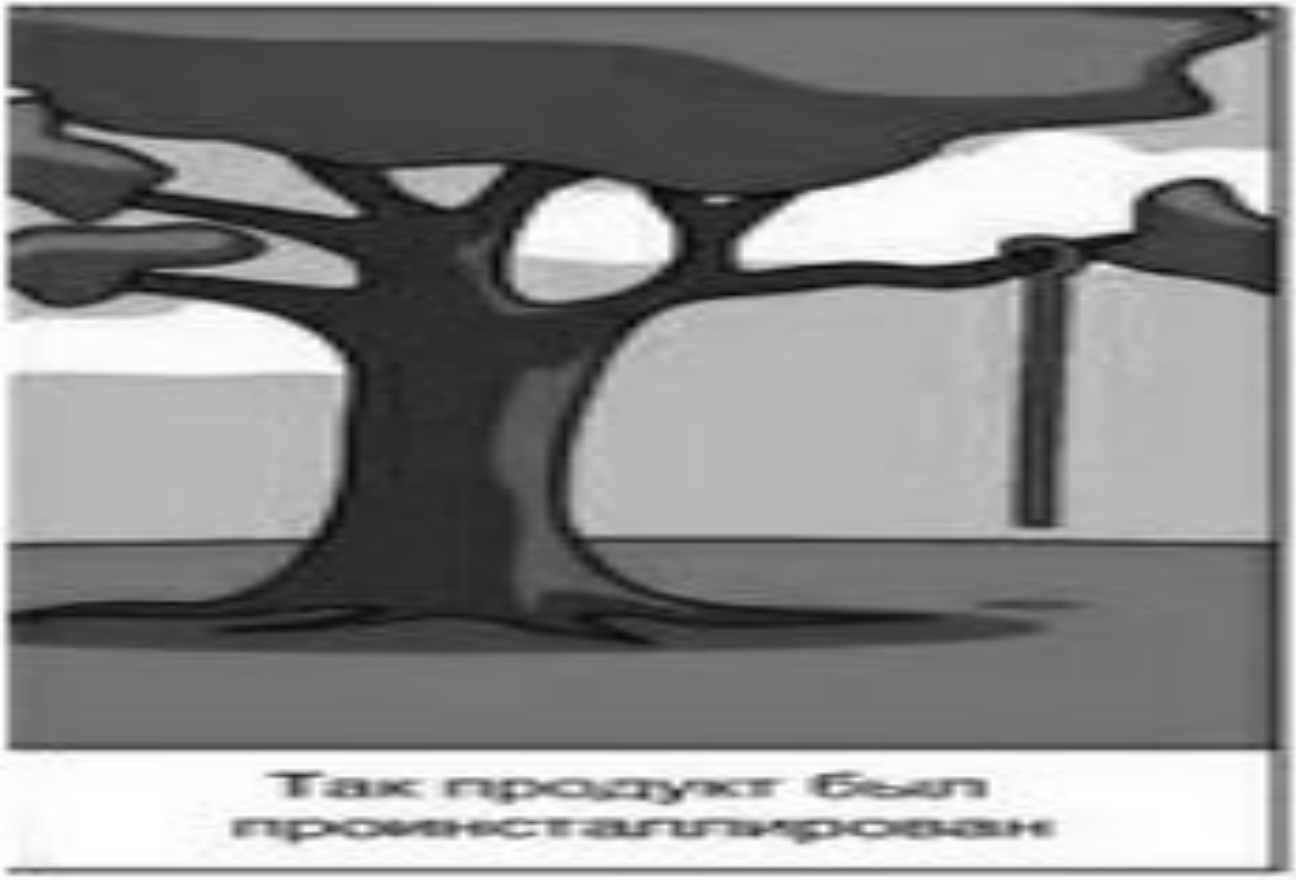

Так описал
бизнес-консультант



Так спроектировал аналитик

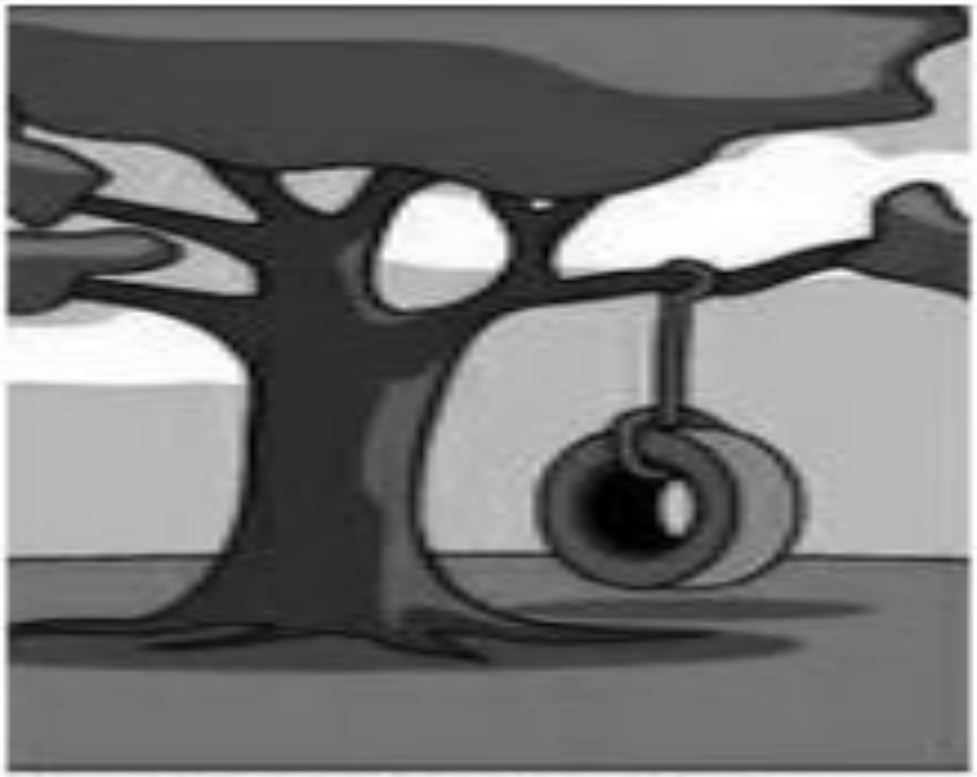

Так осуществлялась
техническая поддержка


Так реализовал
программист


Так продукт был
проинсталлирован

Так проект был документирован

Такой счет был выставлен заказчику



А вот чего на самом деле хотелось заказчику

- Из приведенного примера видна основная проблема программной инженерии вызванной отсутствием четкой спецификации создаваемого продукта.
- Разрешить эту проблему можно только наличием единого, унифицированного средства создания спецификаций, простого и понятного для всех заинтересованных лиц

Введение

- В последнее десятилетие в компьютерном мире наметилась тенденция моделирования сложных систем визуальными (наглядными) моделями.
- Причем в новых методах проектирования сложных компьютерных систем, например ООП и ООАП, наглядные модели очень часто связываются с такими зрительными образами как "взгляды", направленные на сложную систему с различных точек зрения. Набор из нескольких наглядных моделей (модельных взглядов) создает в сознании специалистов интегральный образ сложной компьютерной системы, которую они совместно проектируют.
- Вместе с тем, наглядные модели служат эффективным средством документирования компьютерных систем и их программных обеспечений, а также языком общения между программистами, системными аналитиками и заказчиками систем.

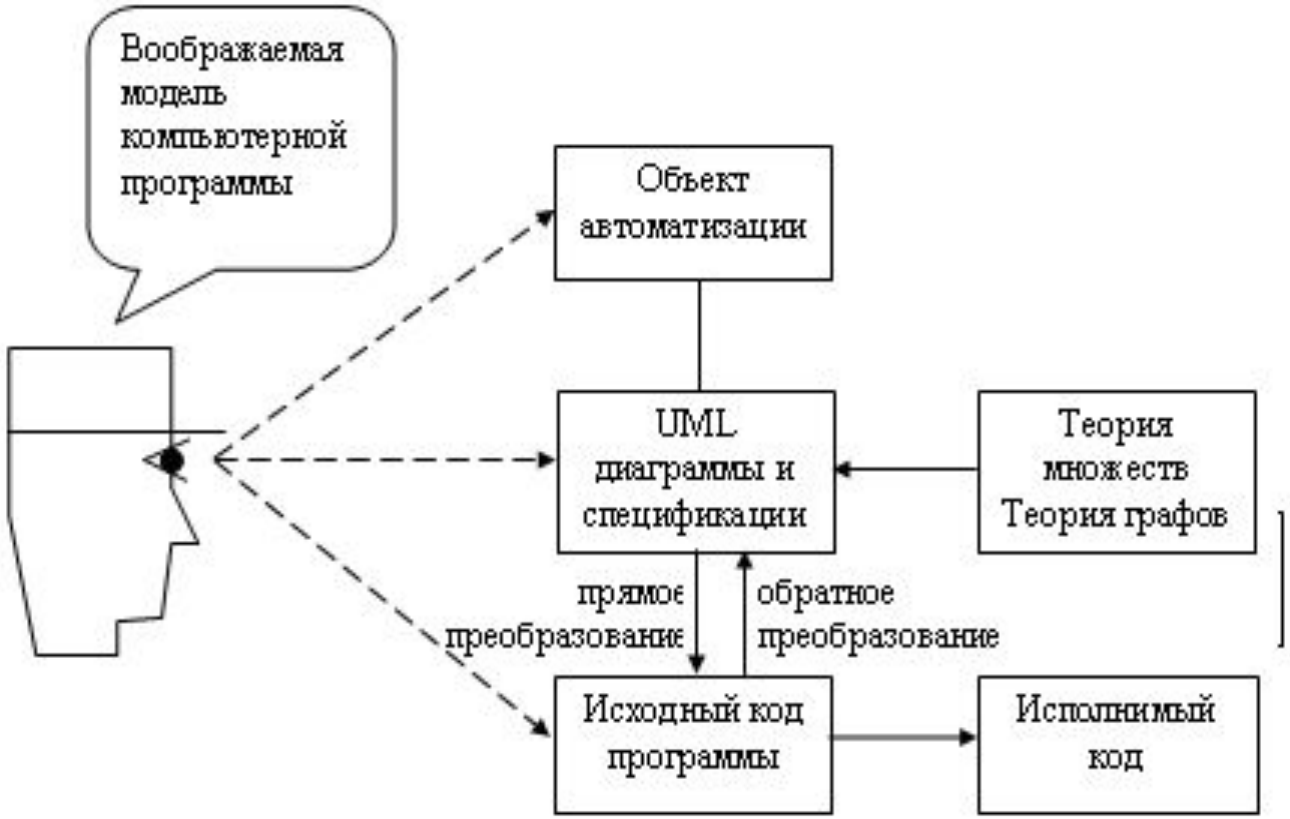
Направления визуального моделирования

- Функциональное моделирование, связано с использованием стандартов IDEF0, IDEF1X, IDEF3 ,DFD, STD и др. с использованием CASE инструментария All Function Process Modeler (BPWin)
- Объектное моделирование , связано с использованием графических схем языка UML с использованием инструментариев Rational Rose и MS Visio 2007/2010

История развития языка UML

- берет начало с октября 1994 года, когда Гради Буч и Джеймс Румбах из Rational Software Corporation начали работу по разработке языка объединяющего ООП с методиками IDEF Начиная работу Г. Буч, Дж. Румбах и А. Джекобсон сформулировали следующие требования к языку моделирования.
- Новый язык должен:
- Позволять моделировать не только программное обеспечение, но и более широкие классы систем и бизнес-приложений, с использованием объектно-ориентированных понятий.
- Явным образом обеспечивать взаимосвязь между базовыми понятиями для моделей концептуального и физического уровней.
- Обеспечивать масштабируемость моделей, что является важной особенностью сложных многоцелевых систем.
- Быть понятен аналитикам и программистам, а также должен поддерживаться специальными инструментальными средствами, реализованными на различных компьютерных платформах
- Работа была закончена в 1995 году результаты отображены книге Г.Буча, Д. Рембо, А. Джекобсона "Руководство пользователя UML"

Схема разработки при использовании UML



Унифицированный язык визуального моделирования (UML)

- ***Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML)*** явился средством для структурного и функционального моделирования. Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML* жизненный цикл информационных систем, и, одновременно, *UML* является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Общие характеристики ***UML***

- ***UML*** представляет собой *объектно-ориентированный* язык моделирования, обладающий следующими основными характеристиками:
- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;
- содержит механизмы расширения и специализации базовых концепций языка.

Виды диаграмм UML

- Диаграмма - это графическое представление совокупности классов, чаще всего изображаемое в виде связного графа, состоящего из вершин (сущностей) и ребер (отношений). С помощью диаграмм можно визуализировать систему с различных точек зрения. Поскольку сложное целое нельзя понять, глядя на него лишь с одной стороны, в UML определено много разных диаграмм, которые позволяют сосредоточиться на различных аспектах моделируемой системы. Всего средствами UML можно представить 9 типов диаграмм.

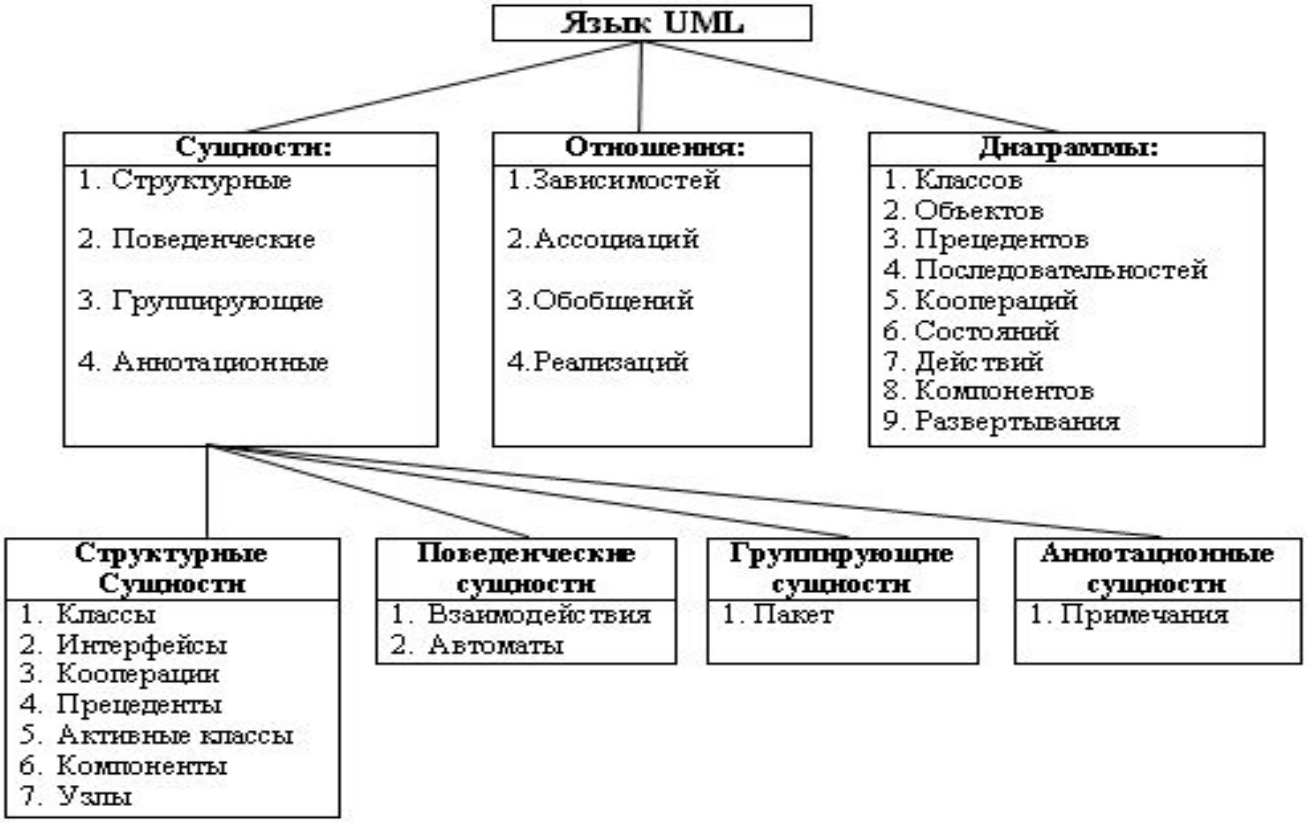
Диаграммы UML

- диаграммы использования;
- диаграммы состояний;
- диаграммы деятельности.
- диаграммы компонентов;
- диаграммы классов;
- диаграммы объектов;
- диаграммы развертывания.
- диаграммы последовательности;
- диаграммы кооперации;

Словарь языка UML

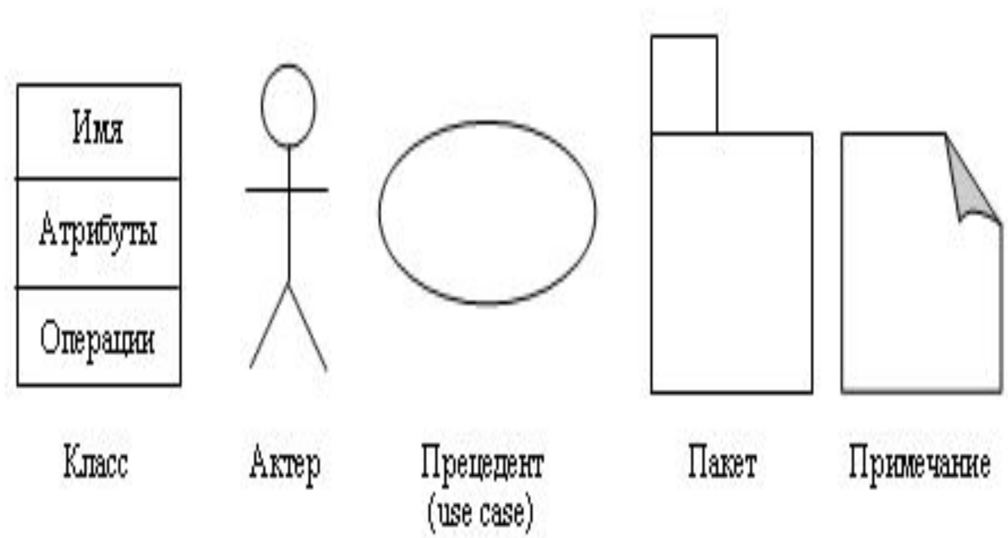
- включает три вида строительных блоков:
- сущности;
- отношения;
- диаграммы.
- Сущности - это абстракции, являющиеся основными элементами модели.
- Отношения связывают различные сущности;
- Диаграммы группируют представляющие интерес совокупности сущностей.

Схема словаря языка



Сущности(Entity)

- Сущности являются основными объектно-ориентированными блоками языка. С их помощью можно создавать корректные модели.
- Язык UM
поведен
- **“Структ**
пиктогра
интерфе
классам
- **Поведе**
- **Диаграм**
второго
только о



,
еют

- **Аннотац** амм,
называемых примечаниями

Сущности языка

- **В UML имеется четыре типа сущностей:**
- структурные;
- поведенческие;
- группирующие;
- аннотационные

Структурные сущности

- Структурные сущности - это имена существительные в моделях на языке UML Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует 6 разновидностей структурных сущностей.

Виды структурных сущностей.

- Класс
- Интерфейс
- Кооперация
- Прецедент (Use case)
- Компонент
- Узел

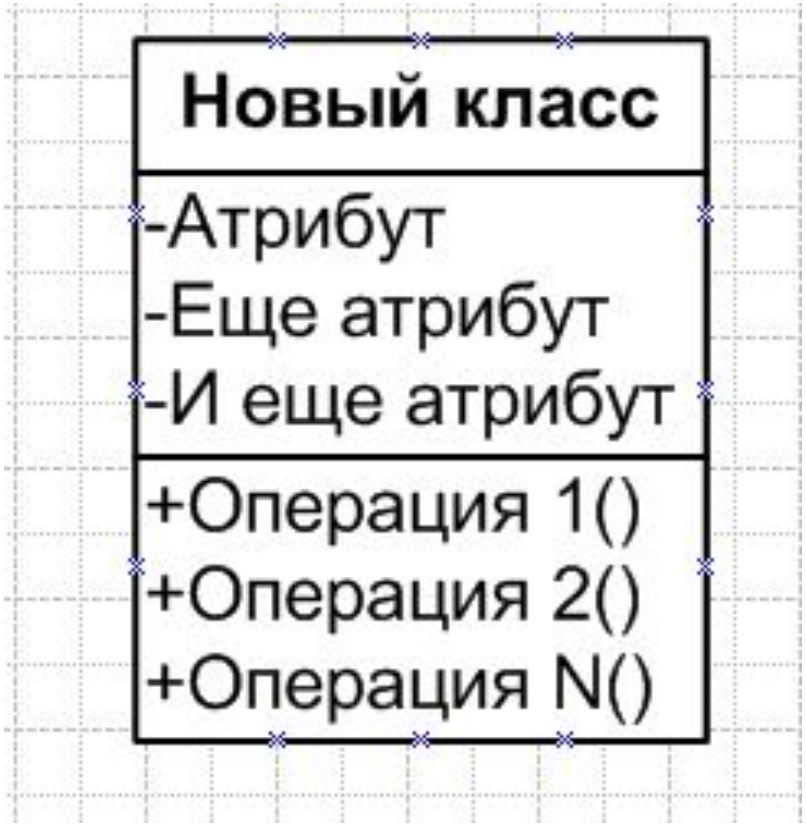
Понятие класса

- классы - это строительные блоки любой объектно-ориентированной системы. Они представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. При проектировании объектно-ориентированных систем диаграммы классов обязательны.
- Классы используются в процессе анализа предметной области для составления словаря предметной области разрабатываемой системы. Это могут быть как абстрактные понятия предметной области, так и классы, на которые опирается разработка и которые описывают программные или аппаратные сущности.

Синтаксис и семантика основных объектов UML

- ***Классы*** — это базовые элементы **UML** *Классы* представляют собой описание совокупностей однородных объектов с присущими им свойствами — атрибутами, операциями, отношениями и семантикой. В рамках модели каждому *классу* присваивается уникальное имя, отличающее его от других *классов*. Если используется составное имя (в начале имени добавляется имя пакета, куда входит *класс*), то имя *класса* должно быть уникальным в пакете.
- **Атрибут** — это свойство *класса*, которое может принимать множество значений. Множество допустимых значений атрибута образует домен. Атрибут имеет имя и отражает некоторое свойство моделируемой сущности, общее для всех объектов данного *класса*. *Класс* может иметь произвольное количество атрибутов.
- **Операция** — реализация функции, которую можно запросить у любого объекта *класса*. Операция показывает, что можно сделать с объектом. Исполнение операции часто связано с обработкой и изменением значений атрибутов объекта, а также изменением состояния объекта.

пример класса



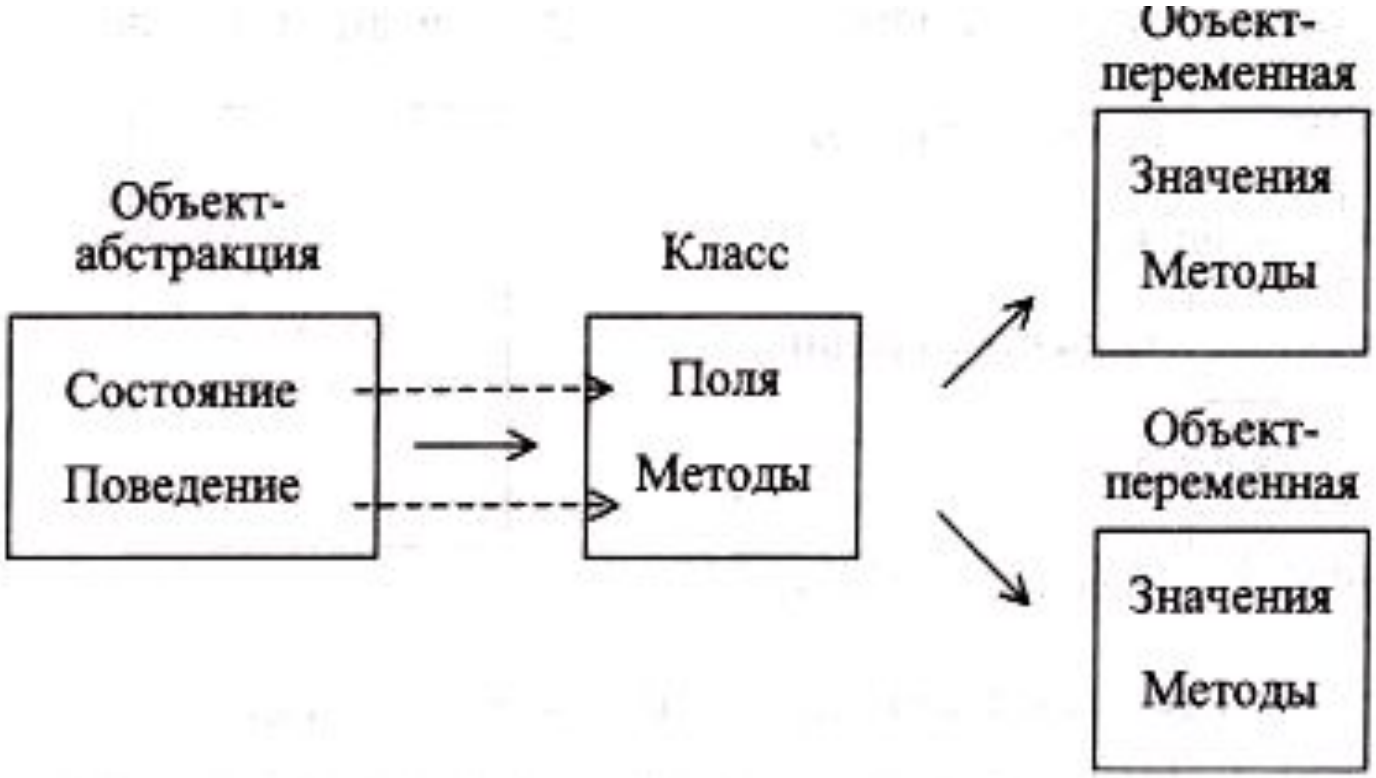
Имя класса

- должно быть уникальным в пределах пакета, который может содержать одну или несколько диаграмм классов. Имя указывается в самой верхней секции прямоугольника, поэтому она часто называется секцией имени класса. В дополнение к общему правилу именования элементов языка UML, имя класса записывается по центру секции имени полужирным шрифтом и должно начинаться с заглавной буквы. Рекомендуется в качестве имен классов использовать существительные, записанные по практическим соображениям без пробелов. В секции имени класса могут также находиться стереотипы или ссылки на стандартные шаблоны, от которых образован данный класс и, соответственно, от которых он наследует атрибуты и операции.

Виды классов

- Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML различают **конкретные** и **абстрактные** классы.
- **Конкретный класс** (concrete class) — класс, на основе которого могут быть непосредственно созданы экземпляры или объекты.
- **Абстрактный класс** (abstract class) — класс, который не имеет экземпляров или объектов. Для обозначения имени абстрактного класса используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом.

Соответствие объекта-абстракции классу и объектам-переменным в ООП



Атрибут (attribute)

- — содержательная характеристика класса, описывающая множество значений, которые могут принимать отдельные объекты этого класса. Атрибут класса служит для представления отдельного свойства или признака, который является общим для всех объектов данного класса. Атрибуты класса записываются во второй сверху секции прямоугольника класса. Эту секцию часто называют секцией атрибутов.
- В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

Операции класса

- сервис, предоставляемый каждым экземпляром или объектом класса по требованию своих клиентов, в качестве которых могут выступать другие объекты, в том числе и экземпляры данного класса.
- Операции класса записываются в третьей сверху секции прямоугольника класса, которую часто называют секцией операций. Совокупность операций характеризует функциональный аспект поведения всех объектов данного класса. Запись операций класса в языке UML также стандартизована и подчиняется определенным синтаксическим правилам. При этом каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени операции, выражения типа возвращаемого операцией значения и, возможно, строка-свойство данной операции.

Синтаксис *UML* для свойств *классов*

- <признак видимости> <имя атрибута> : <тип данных>
- = <значение по умолчанию>
- <признак видимости> <имя операции> <(список аргументов)>
- Видимость свойства указывает на возможность его использования другими *классами*. Один *класс* может «видеть» другой, если тот находится в области действия первого и между ними существует явное или неявное отношение.
- В языке *UML* определены три уровня видимости:

Уровни видимости

- **public** (общий) — любой внешний *класс*, который «видит» данный, может пользоваться его общими свойствами. Обозначаются знаком «+» перед именем атрибута или операции;
- **protected** (защищенный) — только любой потомок данного *класса* может пользоваться его защищнными свойствами. Обозначаются знаком «#»;
- **private** (закрытый) — только данный *класс* может пользоваться этими свойствами. Обозначаются символом «-» .

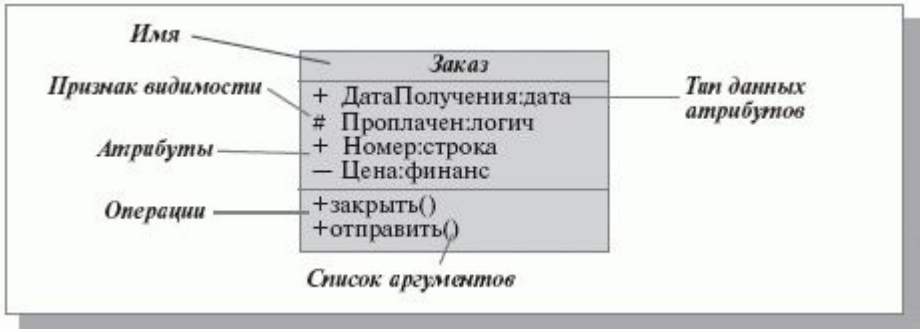
область действия *классов*

- важной характеристикой атрибутов и операций *классов* является область действия.
- Область действия свойства указывает, будет ли оно проявлять себя по-разному в каждом экземпляре *класса*, или одно и то же значение свойства будет совместно использоваться всеми экземплярами:
- instance (экземпляр) — у каждого экземпляра *класса* есть собственное значение данного свойства;
- classifier (классификатор) — все экземпляры совместно используют общее значение данного свойства (выделяется на диаграммах подчеркиванием).

кратность *класса*

- Возможное количество экземпляров *класса* называется его кратностью. В *UML* можно определять следующие разновидности *классов*:
- не содержащие ни одного экземпляра — тогда *класс* становится служебным (Abstract);
- содержащие ровно один экземпляр (Singleton);
- содержащие заданное число экземпляров;
- содержащие произвольное число экземпляров.

Пример класса



Определение класса в ООП

- Формат описания класса выглядит следующим образом:
- ***Type*** <имя объявляемого класса> = ***class(***<имя родителя>)
- ***private*** <скрытые элементы класса>
- ***protected*** <защищенные элементы класса>
- ***public*** <общедоступные элементы класса>
- ***published*** <опубликованные элементы класса>
- ***automated*** <элементы, реализующие OLE-механизм>

ДИАГРАММЫ UML

диаграммы UML

- Диаграмма - это графическое представление совокупности классов, чаще всего изображаемое в виде связного графа, состоящего из вершин (сущностей) и ребер (отношений).
- С помощью диаграмм можно визуализировать процессы и системы с различных точек зрения. Поскольку сложное целое нельзя понять, глядя на него лишь с одной стороны, в UML определено много разных диаграмм, которые позволяют сосредоточиться на различных аспектах моделируемых процессов. Всего средствами UML можно представить 9 типов диаграмм.

Диаграммы UML

1. диаграммы использования;
2. диаграммы состояний;
3. диаграммы деятельности.
4. диаграммы компонентов;
5. диаграммы классов;
6. диаграммы объектов;
7. диаграммы развертывания.
8. диаграммы последовательности;
9. диаграммы кооперации;

Диаграммы классов

- *Классы* в *UML* изображаются на ***диаграммах классов***, которые позволяют описать процессы и системы в статическом состоянии — определить типы объектов системы и различного рода статические связи между ними.
- Между *классами* возможны различные отношения:
- **зависимости**, которые описывают существующие между *классами* отношения использования;
- **обобщения**, связывающие обобщенные *классы* со специализированными;
- **ассоциации**, отражающие структурные отношения между объектами *классов*.

Диаграмма классов

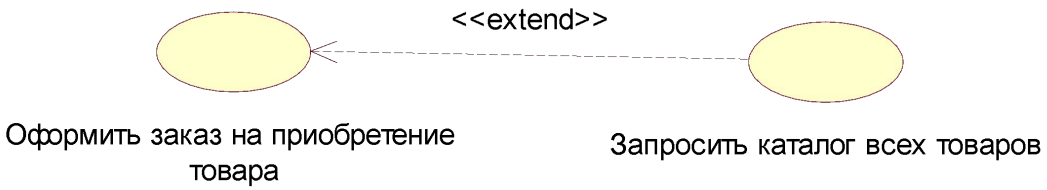
- - это набор статических, декларативных элементов модели. Диаграммы классов могут применяться и при прямом проектировании, то есть в процессе разработки новой системы, и при обратном проектировании - описании существующих и используемых систем. Информация с диаграммы классов напрямую отображается в исходный код приложения - в большинстве существующих инструментов UML-моделирования возможна кодогенерация для определенного языка программирования (обычно Java или C++). Таким образом, диаграмма классов - конечный результат проектирования и отправная точка процесса разработки.

Диаграммы классов

- *Классы* в *UML* изображаются на ***диаграммах классов***, которые позволяют описать систему в статическом состоянии — определить типы объектов системы и различного рода статические связи между ними.
- Между *классами* возможны различные отношения:
- зависимости, которые описывают существующие между *классами* отношения использования;
- обобщения, связывающие обобщенные *классы* со специализированными;
- ассоциации, отражающие структурные отношения между объектами *классов*.

отношение зависимость

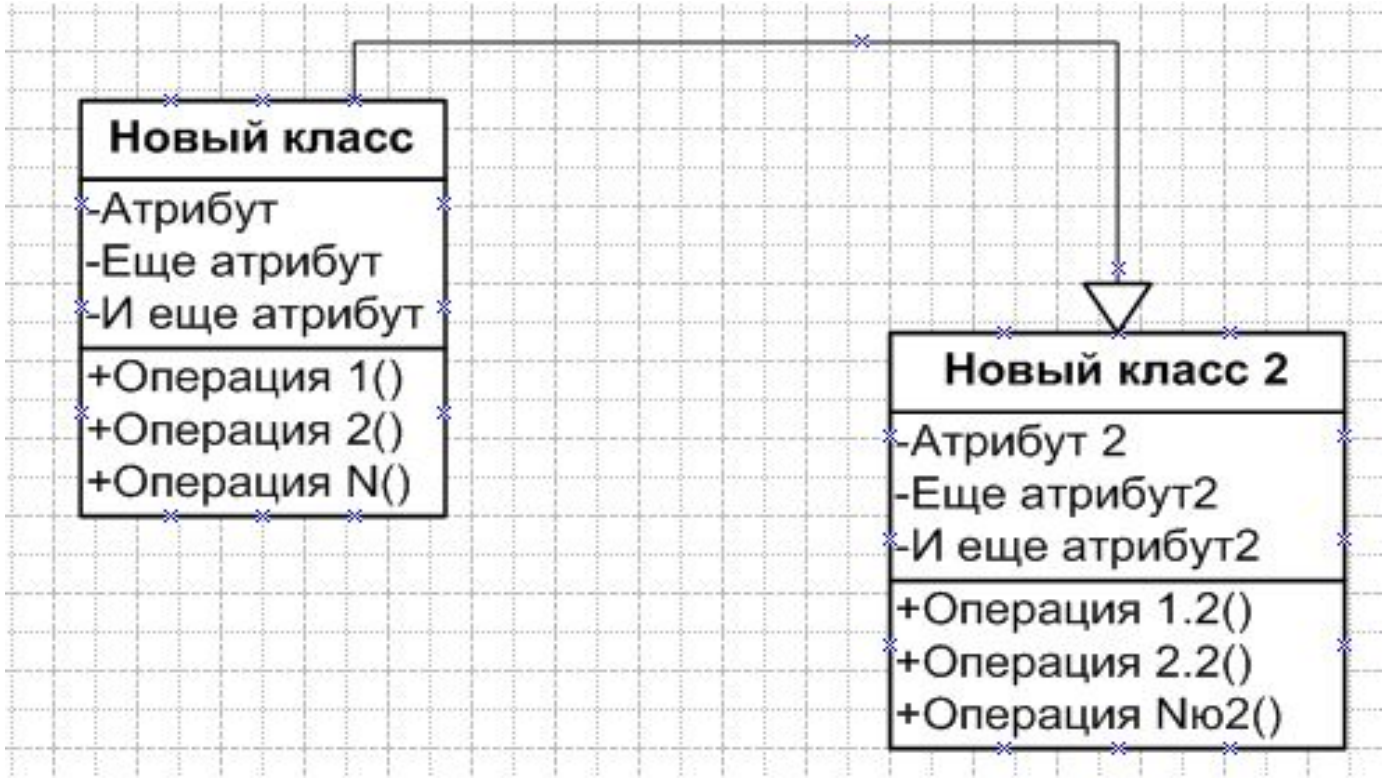
- Зависимостью называется отношение использования, согласно которому изменение в спецификации одного элемента (например, *класса* «товар») может повлиять на использующий его элемент (*класс* «строка заказа»). Часто зависимости показывают, что один *класс* использует другой в качестве аргумента.



отношение обобщение

- **Обобщение** — это отношение между общей сущностью (родителем — *класс* «клиент») и ее конкретным воплощением (потомком — *классы* «корпоративный клиент» или «частный клиент»). Объекты *класса*-потомка могут использоваться всюду, где встречаются объекты *класса*-родителя, но не наоборот. При этом он наследует свойства родителя (его атрибуты и операции). Операция потомка с той же сигнатурой, что и у родителя, замещает операцию родителя; это свойство называют полиморфизмом. *Класс*, у которого нет родителей, но есть потомки, называется корневым. *Класс*, у которого нет потомков, называется листовым.

Пример отношения обобщения



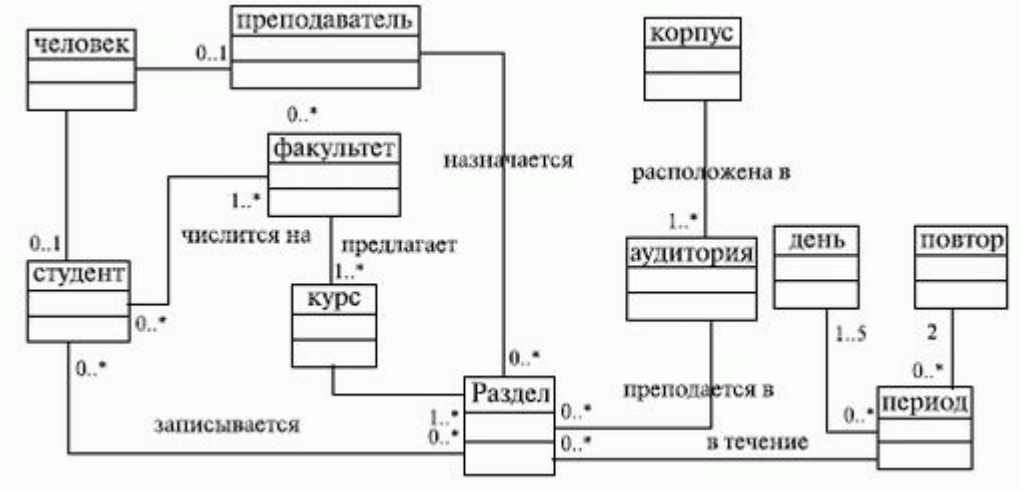
отношение ассоциация

- **Ассоциация** — это отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа («клиент» может сделать «заказ»). Если между двумя *классами* определена ассоциация, то можно перемещаться от объектов одного *класса* к объектам другого. При необходимости направление навигации может задаваться стрелкой. Допускается задание ассоциаций на одном *классе*. В этом случае оба конца ассоциации относятся к одному и тому же *классу*. Это означает, что с объектом некоторого *класса* можно связать другие объекты из того же *класса*.
- Ассоциации может быть присвоено имя, описывающее семантику отношений. Каждая ассоциация имеет две роли, которые могут быть отражены на диаграмме Роль ассоциации обладает свойством множественности, которое показывает, сколько соответствующих объектов может участвовать в данной связи.

Пример отношения «Ассоциация»

Отображение связей между классами

Пример диаграммы классов



Диаграммы использования (прецедентов)

- **Цель диаграммы**
- определение границы и контекста моделируемой предметной области на ранних этапах проектирования;
- формирование общих требований к поведению проектируемой системы;
- разработка концептуальной модели системы для ее последующей детализации;
- подготовка документации для взаимодействия с заказчиками и пользователями систем

Диаграммы использования описывают функциональность ИС, которая будет видна пользователям системы. «Каждая функциональность» изображается в виде «прецедентов использования» (use case) или просто прецедентов.

- Прецедент — это типичное взаимодействие пользователя с системой, которое при этом:

1. описывает видимую пользователем функцию,
2. может представлять различные уровни детализации,
3. обеспечивает достижение конкретной цели, важной для пользователя.

Обозначение Прецедента

- обозначается на диаграмме овалом, связанным с пользователями, которых принято называть действующими лицами (actors).
- Действующие лица используют систему (или используются системой) в данном прецеденте.
- Действующее лицо выполняет некоторую роль в данном прецеденте. На диаграмме изображается только одно действующее лицо, однако реальных пользователей, выступающих в данной роли по отношению к ИС, может быть много. Список всех прецедентов фактически определяет функциональные требования к ИС, которые лежат в основе разработки технического задания на создание системы.

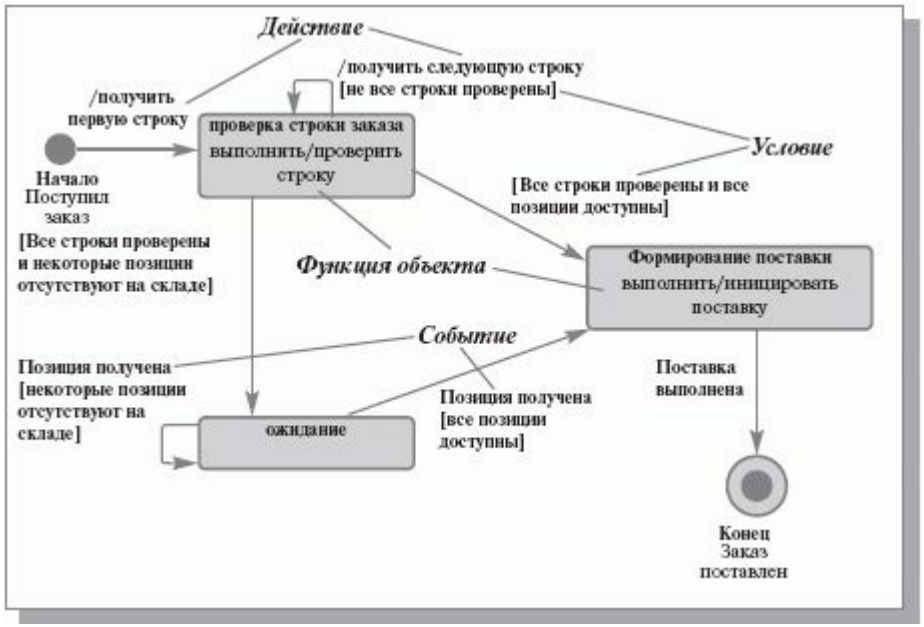
Связи на диаграммах прецедентов

Диаграммы состояний

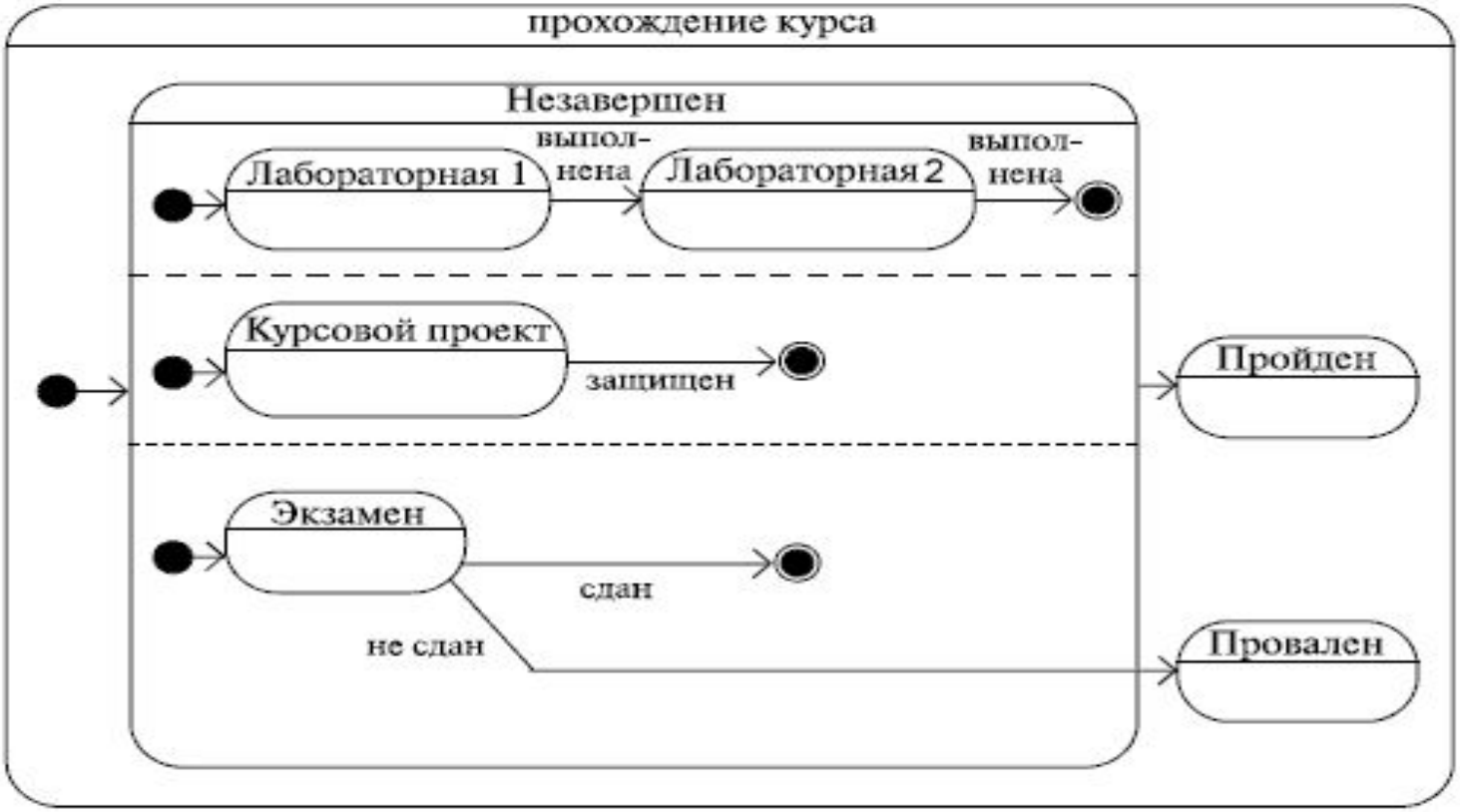
- **используются для описания поведения сложных систем.**
- Они определяют все возможные состояния, в которых может находиться объект, а также процесс смены состояний объекта в результате некоторых событий. Эти диаграммы обычно используются для описания поведения одного объекта в нескольких прецедентах.
- **Прямоугольниками** представляются состояния, через которые проходит объект во время своего поведения. **Состояниям** соответствуют определенные значения атрибутов объектов. **Стрелки** представляют переходы от одного состояния к другому, которые вызываются выполнением некоторых функций объекта.
- Имеется два вида псевдо-состояний: **начальное состояние**, в котором находится только что созданный объект, и **конечное состояние**, которое объект не покидает, как только туда перешел.
- Переходы имеют метки, которые синтаксически состоят из трех необязательных частей

  <Событие> <[Условие]> < / Действие>
- На диаграммах также отображаются функции, которые выполняются объектом в определенном состоянии. Синтаксис метки деятельности:
- выполнить/< деятельность >.

Пример диаграммы состояний



Пример 2 диаграмма состояний с несколькими точками входа и выхода



Диаграммы деятельности

- Диаграмма деятельности — это частный случай *диаграммы состояний*. На диаграмме деятельности представлены переходы потока управления от одной деятельности к другой внутри системы.
- Этот вид диаграмм обычно используется для описания поведения, включающего в себя множество параллельных процессов. ***Диаграммы деятельности*** удобно применять для визуализации алгоритмов, по которым работают операции классов.

  Основными элементами диаграмм деятельности являются овалы, изображающие действия объекта;
- линейки синхронизации, указывающие на необходимость завершить или начать несколько действий (модель логического условия «И»);
- ромбы, отражающие принятие решений по выбору одного из маршрутов выполнения процесса (модель логического условия «ИЛИ»);
- стрелки — отражают последовательность действий, могут иметь метки условий.

Диаграммы деятельности

- На диаграмме деятельности могут быть представлены действия, соответствующие нескольким вариантам использования. На таких диаграммах появляется множество начальных точек, поскольку они отражают теперь реакцию системы на множество внешних событий.
- Таким образом, диаграммы деятельности позволяют получить полную картину поведения системы и легко оценивать влияние изменений в отдельных вариантах использования на конечное поведение системы.
- Любая деятельность может быть подвергнута дальнейшей декомпозиции и представлена в виде отдельной диаграммы деятельности или спецификации (словесного описания).

Пример «обработка заказа»

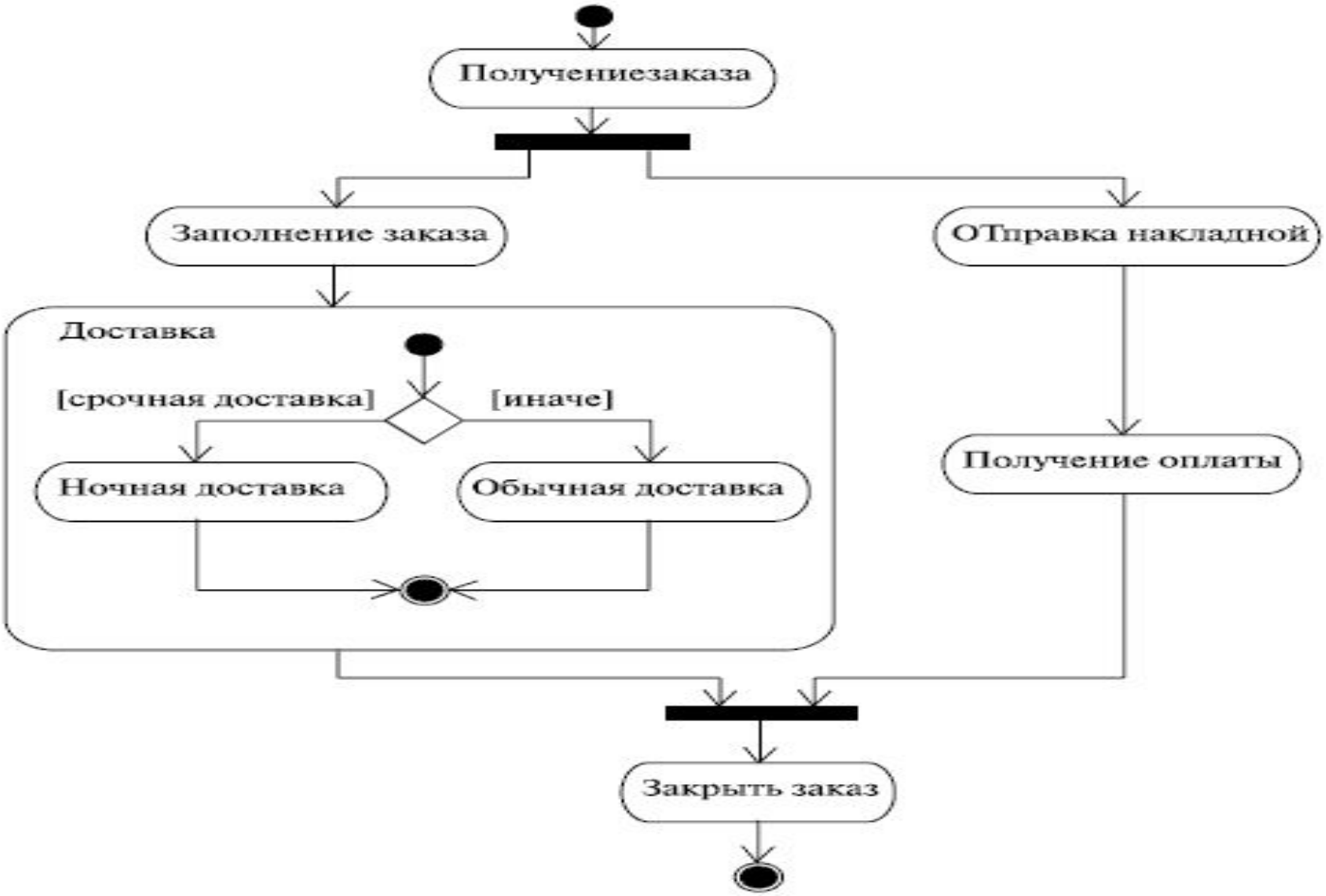
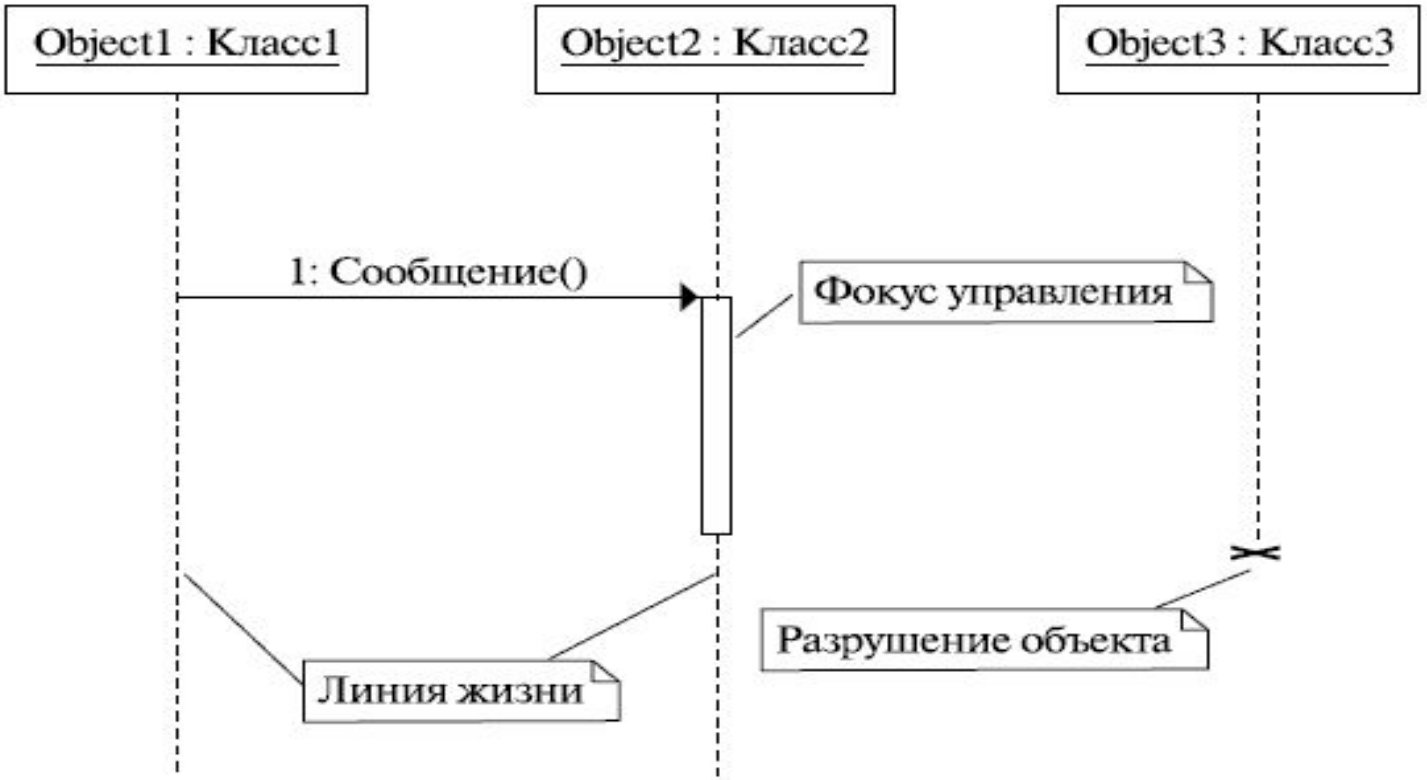


Диаграмма последовательностей (sequence diagram)

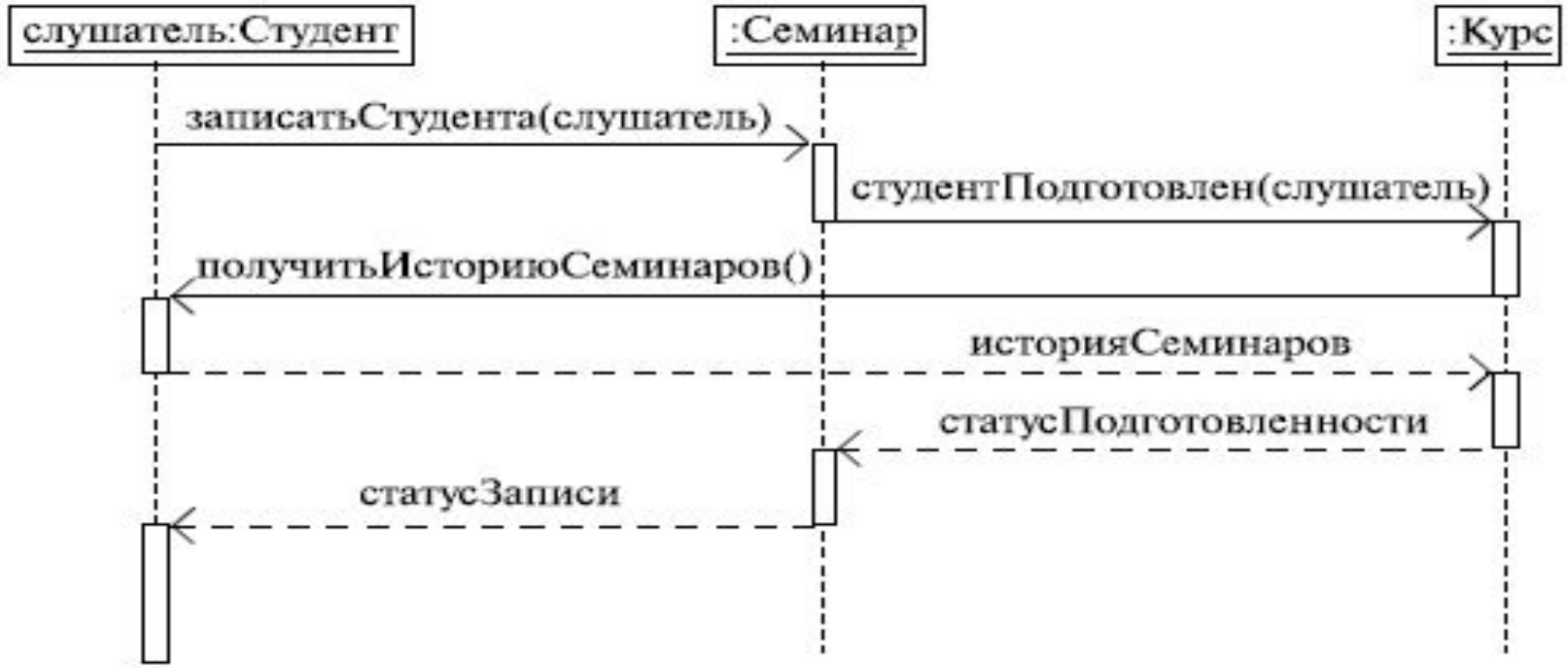
- Диаграмма последовательностей относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени, т.е. диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами.

- Диаграммы последовательностей можно (и нужно!) использовать для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. При проектировании ИС их часто используют для описания сценария диалога.
- Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.
- Обозначения используются на диаграмме последовательностей
- Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов), сообщения (вызовы методов) - линиями со стрелками, возвращаемые результаты - пунктирными линиями со стрелками. Прямоугольники на вертикальных линиях под каждым из объектов показывают "время жизни" (фокус) объектов.

Структура диаграммы последовательностей



Пример диаграммы последовательностей.



Диаграммы компонентов

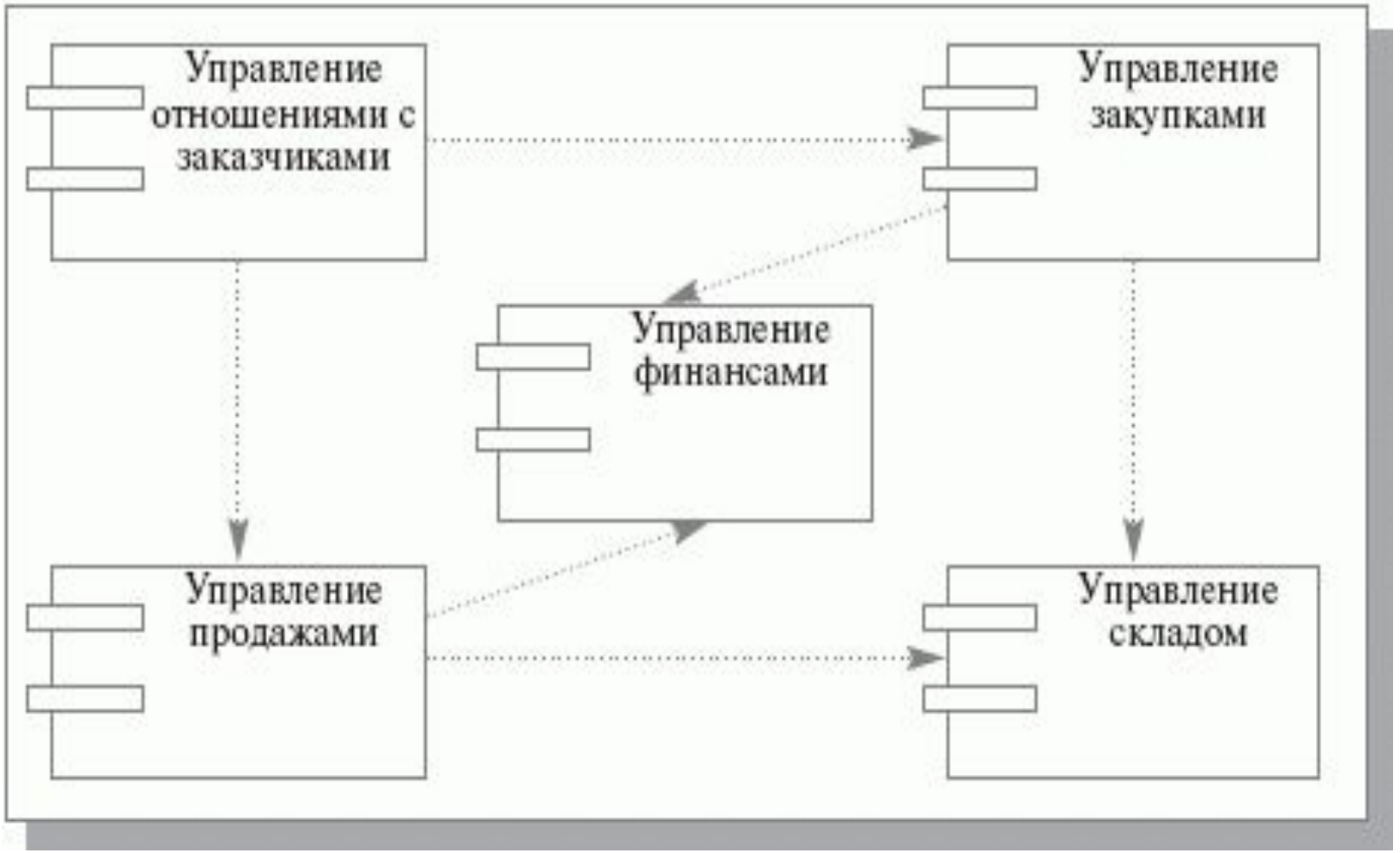
- ***Диаграммы компонентов*** позволяют изобразить модель системы на физическом уровне.
- Элементами диаграммы являются компоненты — физические замещаемые модули системы. Каждый компонент является полностью независимым элементом системы. Разновидностью компонентов являются узлы. Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки.

  Узлы делятся на два типа:
- устройства — узлы системы, в которых данные не обрабатываются.
- процессоры — узлы системы, осуществляющие обработку данных.

Диаграммы компонентов

- Компонентом может быть любой достаточно крупный модульный объект, такой как таблица или экстент базы данных, подсистема, бинарный исполняемый файл, готовая к использованию система или приложение.
- *Основное назначение диаграмм компонентов — разделение системы на элементы, которые имеют стабильный интерфейс и образуют единое целое. Это позволяет создать ядро системы, которое не будет меняться в ответ на изменения, происходящие на уровне подсистем.*

Пример корпоративной ДК



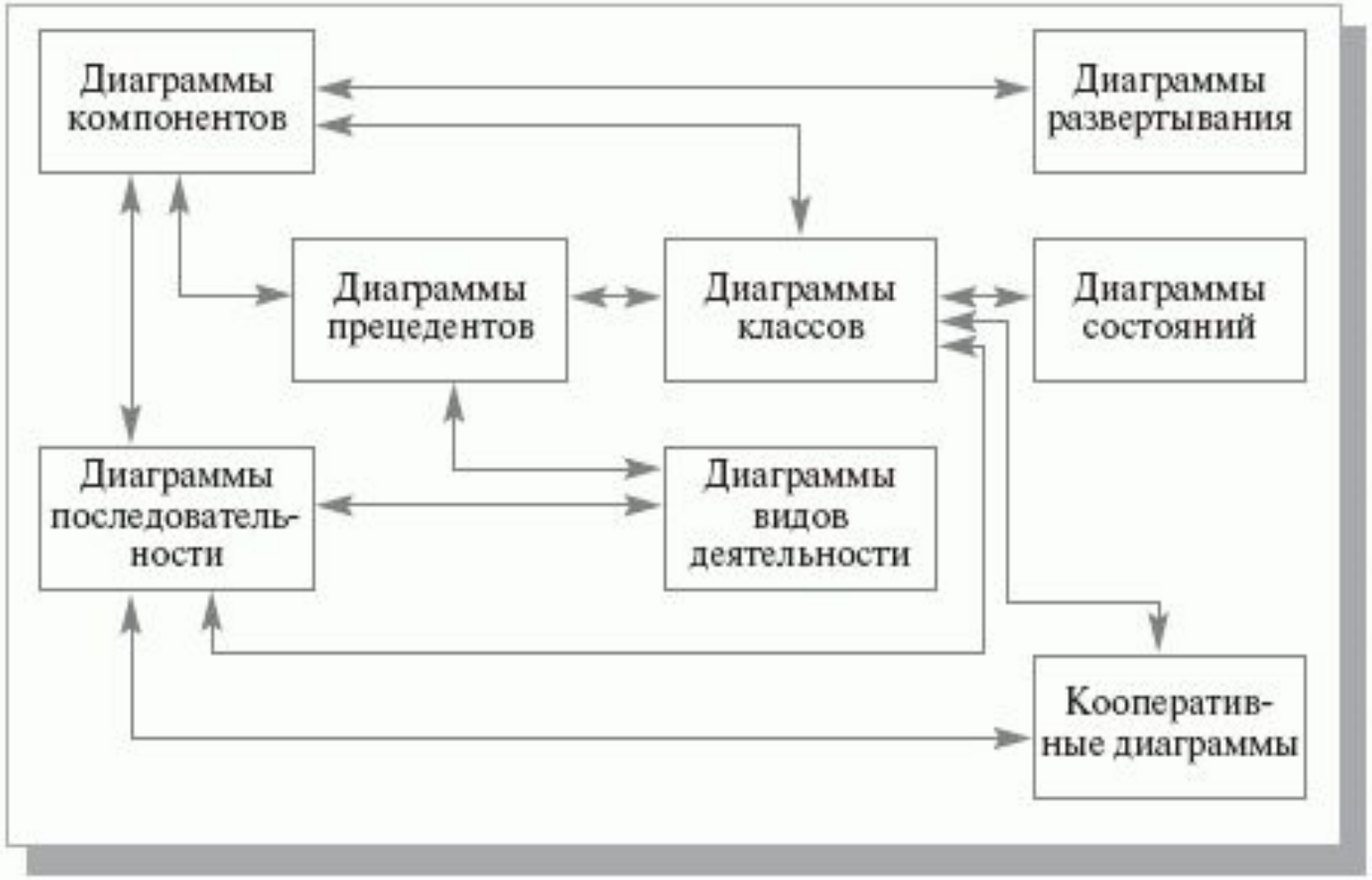
Пакеты UML

- Пакеты представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить диаграммы различного типа и назначения. В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя и иногда — описание содержимого.

Диаграмма пакетов

- содержит пакеты *классов* и зависимости между ними. Зависимость между двумя пакетами имеет место в том случае, если изменения в определении одного элемента влекут за собой изменения в другом.
- По отношению к пакетам можно использовать механизм обобщения основных типов UML-диаграмм, используемых в проектировании информационных систем.
- Взаимосвязи между диаграммами. Поддержка UML итеративного процесса проектирования ИС На этапе создания концептуальной модели для описания бизнес-деятельности используются модели бизнес-прецедентов и диаграммы видов деятельности, для описания бизнес-объектов – модели бизнес-объектов и диаграммы последовательностей.

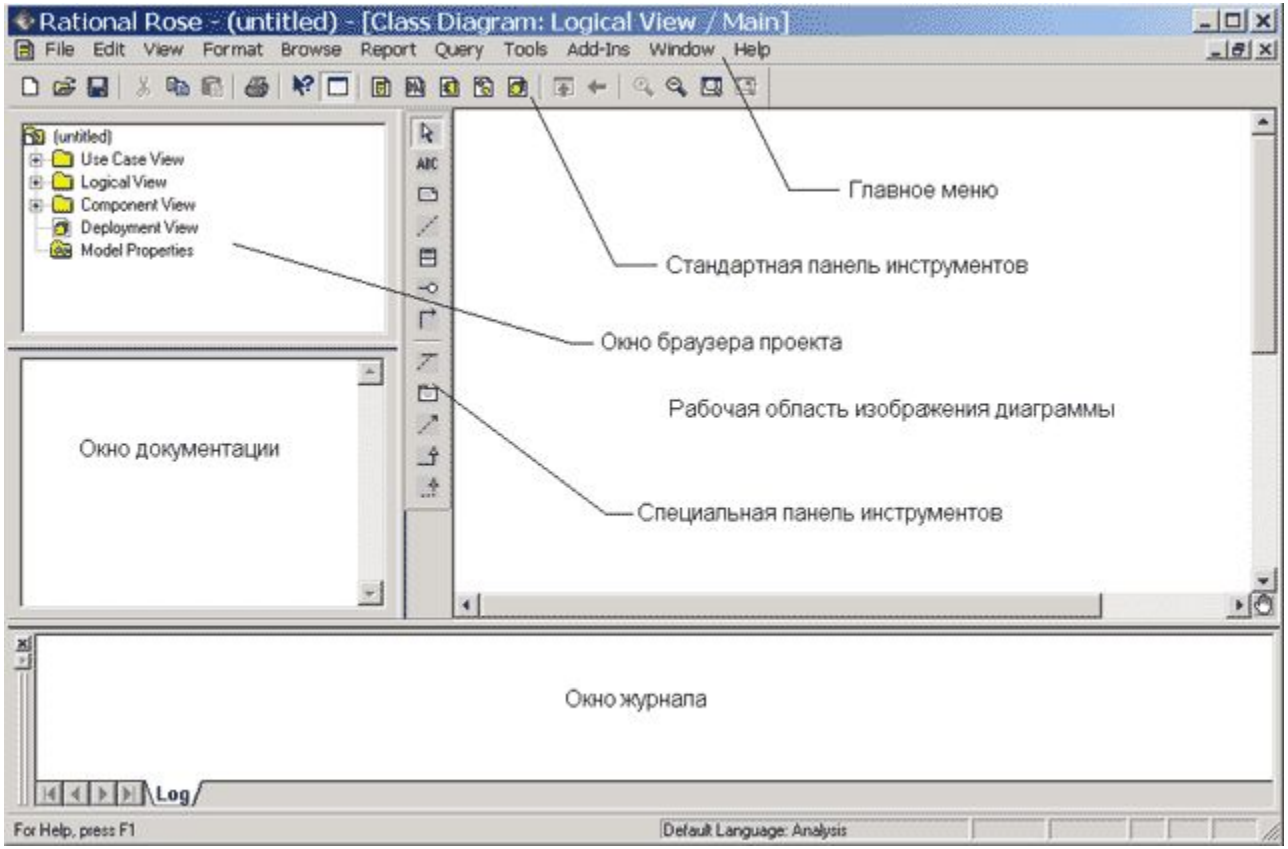
Пример диаграммы пакетов



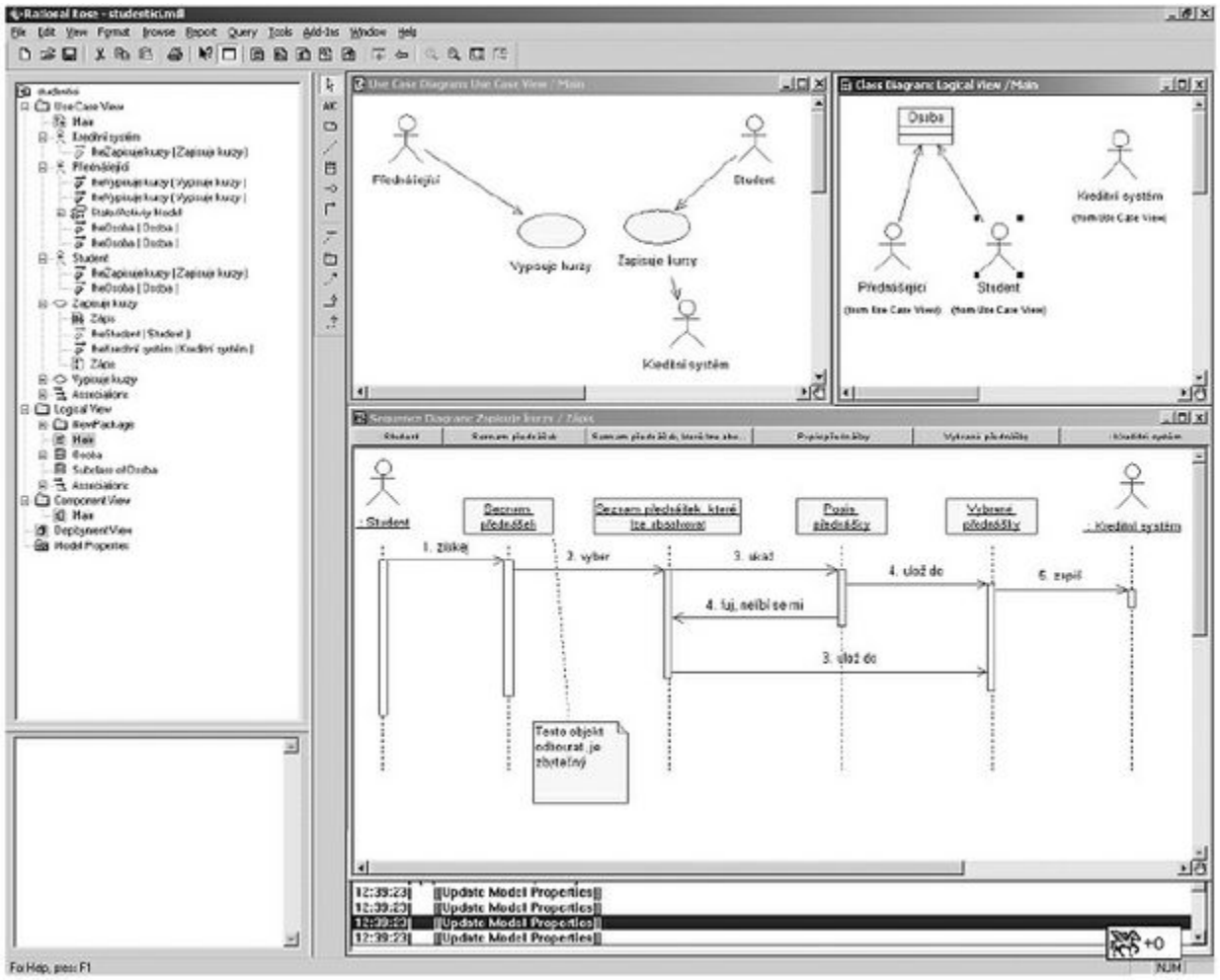
Инструментарий Rational Rose

- - это объектно-ориентированное средство автоматизированного проектирования ПС. В его основе лежит CASE-технология, комплексный подход и использование единой унифицированной нотации на всех этапах жизненного цикла создания ПС.
- Для работы с Rational Rose необходим UML - графический язык описания архитектуры системы. Графические возможности продукта позволяют решать задачи, связанные с проектированием, на различных уровнях абстракции: от общей модели процессов предприятия до конкретной модели класса в создаваемом программном обеспечении (ПО). В среде Rational Rose проектировщик и программист работают в тандеме. Первый создает логическую модель системы, а второй дополняет ее моделями классов на конкретном языке программирования.
- В настоящее время ПП обеспечивает генерацию кода по модели на ряде языков программирования: Microsoft Visual **C++,** Ada, Java, Visual Basic, CORBA, XML, COM, Oracle. Кроме того, разрабатываются специальные мосты к не входящим в стандартную поставку языкам, например к Delphi.

Окно


Rational Rose - (untitled) - [Class Diagram: Logical View / Main]
File Edit View Format Browse Report Query Tools Add-Ins Window Help
(untitled)
Use Case View
Logical View
Component View
Deployment View
Model Properties
Главное меню
Стандартная панель инструментов
Окно браузера проекта
Окно документации
Рабочая область изображения диаграммы
Специальная панель инструментов
Окно журнала
Log
For Help, press F1
Default Language: Analysis

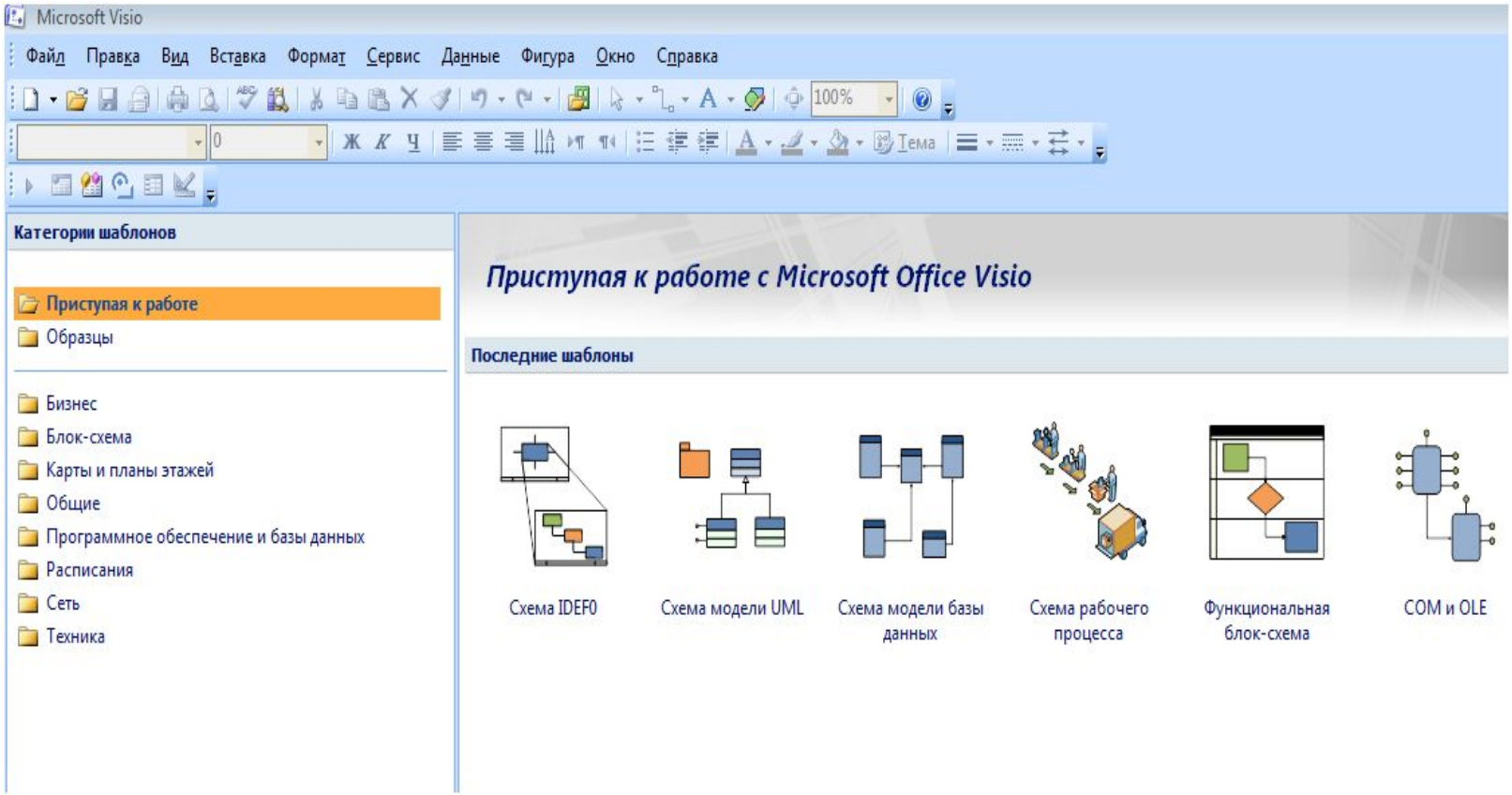
Пример использования при проектировании ИС



Инструментарий Microsoft Visio

- позволяет документировать бизнес-процессы в соответствии со стандартами ISO 9000, нотацией структурного (IDEFи DFD) и объектно-ориентированного проектирования (UML), представлять структуру реляционной базы БД (IDEF1X).
- Microsoft Visio поддерживает совместную работу пользователей с документами при помощи службы Windows SharePoint Services, позволяет экспортировать диаграммы совместных обсуждений в Microsoft Word, Microsoft Excel или XML. Возможна также интеграция Microsoft Visio с веб-службой XML и программным обеспечением Microsoft .NET, подключение диаграмм к важным деловым данным для улучшенного восприятия данных и быстрого принятия решений и т.п.
- В настоящее время используются версии 2003- 20010 данной программы. В ряде случаев простая программа MS Visio может заменить дорогостоящие графические процессоры и системы визуального моделирования деловых процессов.

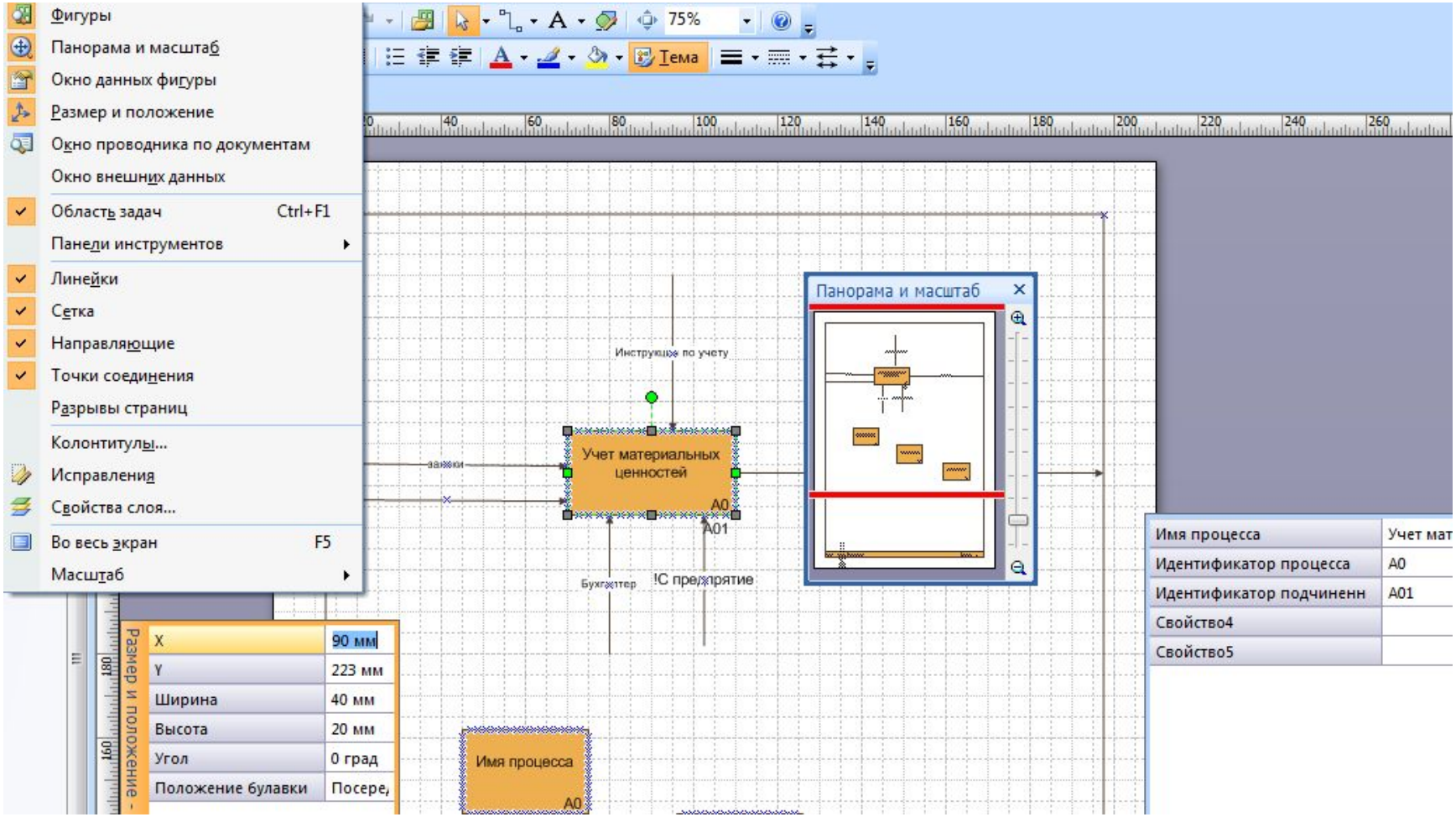
Окно


Microsoft Visio
Файл Правка Вид Вставка Формат Сервис Данные Фигура Окно Справка
100%
Тема
Категории шаблонов
Приступая к работе
Образцы
Бизнес
Блок-схема
Карты и планы этажей
Общие
Программное обеспечение и базы данных
Расписания
Сеть
Техника
Приступая к работе с Microsoft Office Visio
Последние шаблоны
Схема IDEF0
Схема модели UML
Схема модели базы данных
Схема рабочего процесса
Функциональная блок-схема
COM и OLE

Интерфейс MS Visio 2007

- В центральной части расположена область вставки, которая может содержать несколько страниц. Слева – окно «Фигуры» (Shapes), в котором отображаются выбранные пользователем трафареты с наборами фигур.
- Специальные окна (см. команды режима Вид) можно располагать в произвольном месте на экране. В центральной части экрана выводятся страницы документа Visio, имеющие ярлыки с названиями страниц различных типов (например, IDEF0, Page-2 и т.п.). В правой части экрана выводится Панель задач, с помощью которой выполняются определенные виды работ с документом, например, создание нового документа, работа с системой помощи и т.п.

Рабочее окно



Трафареты для построения объектных моделей