Государственное бюджетное профессиональное
образовательное учреждение
Владимирской области
«Владимирский авиамеханический колледж»

Презентация по дисциплине

МДК 02.02 Инструментальные средства разработки ПО
для специальности 090207 Информационные системы и программирование

Язык UML
Лекция 2

Преподаватель Воронова С.А.

При моделировании реальных систем, независимо от предметной области, создаются все те же типы диаграмм, поскольку они соответствуют наиболее часто встречающимся представлениям модели. Как правило, при рассмотрении статических частей системы используются следующие четыре типа:

- диаграммы классов;
- диаграммы объектов;
- диаграммы компонентов;
- диаграммы развертывания.

Для работы с динамическими частями системы применяются пять типов:

- диаграммы прецедентов;
- диаграммы последовательности;
- диаграммы кооперации;
- диаграммы состояний;
- диаграммы деятельности.

Структурные диаграммы

Структурные диаграммы

В UML существует четыре структурных диаграммы для визуализации, специфицирования, конструирования и документирования статических аспектов системы, составляющих ее относительно прочный "костяк".

Подобно тому как статические аспекты дома показывают, что и каким образом будет размещено в здании (стены, двери, окна, трубы, электропроводки, вентиляционные отверстия и т.д.), так и статические аспекты программных систем отражают наличие и расположение классов, интерфейсов, кооперaций, компонентов, узлов и других сущностей.

4 типа структурных диаграмм:

- диаграммы классов - классам, интерфейсам и кооперациям;
- диаграммы объектов - объектам;
- диаграммы компонентов - компонентам;
- диаграммы развертывания - узлам.

1. Диаграмма классов

На ***диаграмме классов*** изображают множество классов, интерфейсов, кооперацій и их отношений. Это самый распространенный тип диаграмм, применяемый при моделировании объектно-ориентированных систем; он используется для иллюстрации статического вида системы с точки зрения проектирования. Диаграммы, на которых показаны активные классы, применяются для работы со статическим видом системы с точки зрения процессов.

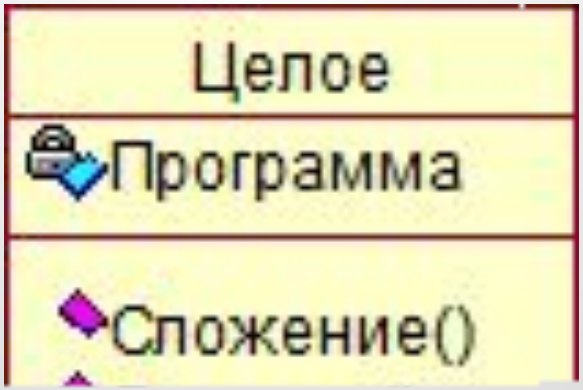
Диаграммы классов

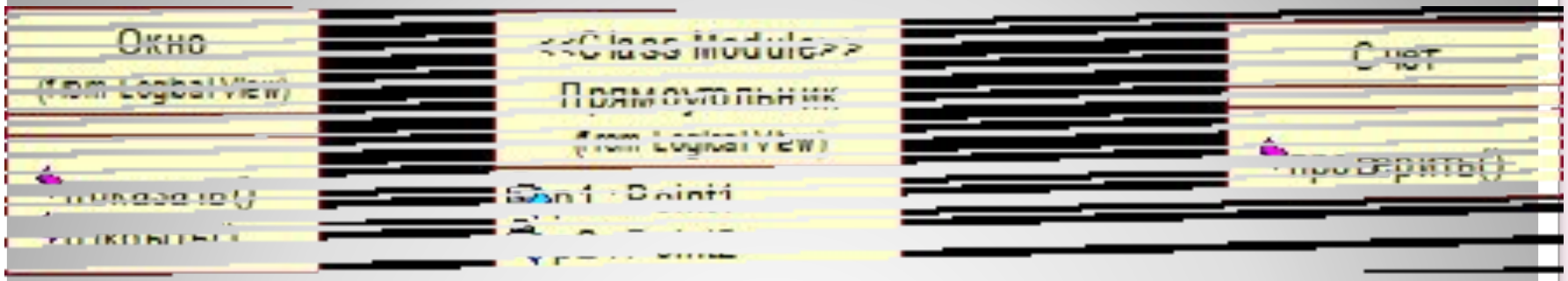
Диаграмма классов состоит из множества элементов, которые в совокупности отражают знания о предметной области. Эти знания интерпретируются в базовых понятиях языка UML, таких как классы, интерфейсы и отношения между ними и их составляющими компонентами.

Класс

Класс *(class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно разделен горизонтальными линиями на три раздела или секции. В этих разделах могут указываться имя класса, атрибуты (переменные) и операции (методы).*

- *Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса.*
- *Даже если секция атрибутов и операций является пустой, в обозначении класса она выделяется горизонтальной линией, чтобы сразу отличить класс от других элементов языка UML. В первом случае для класса "Прямоугольник" указаны только его атрибуты — точки на координатной плоскости, которые определяют его расположение. Для класса "Окно" указаны только его операции, секция атрибутов оставлена пустой.*



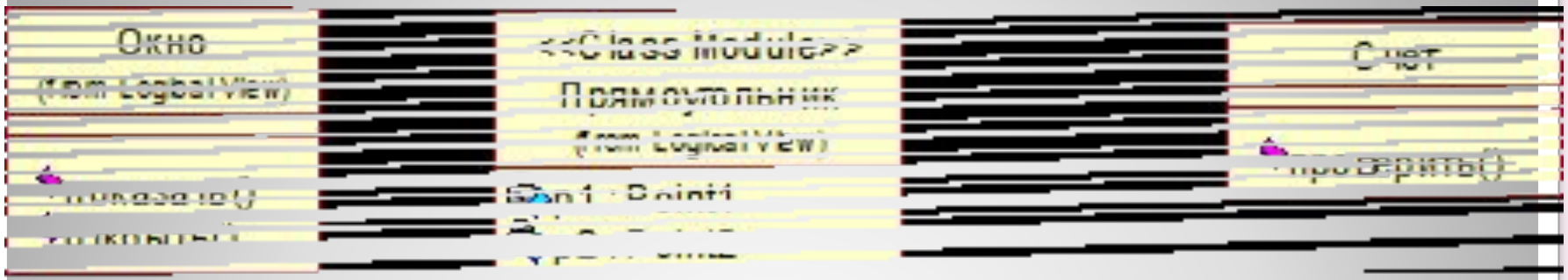


Имя класса *указывается в первой верхней секции прямоугольника. Рекомендуется в качестве имен классов использовать существительные, записанные по практическим соображениям без пробелов.*

Атрибуты класса

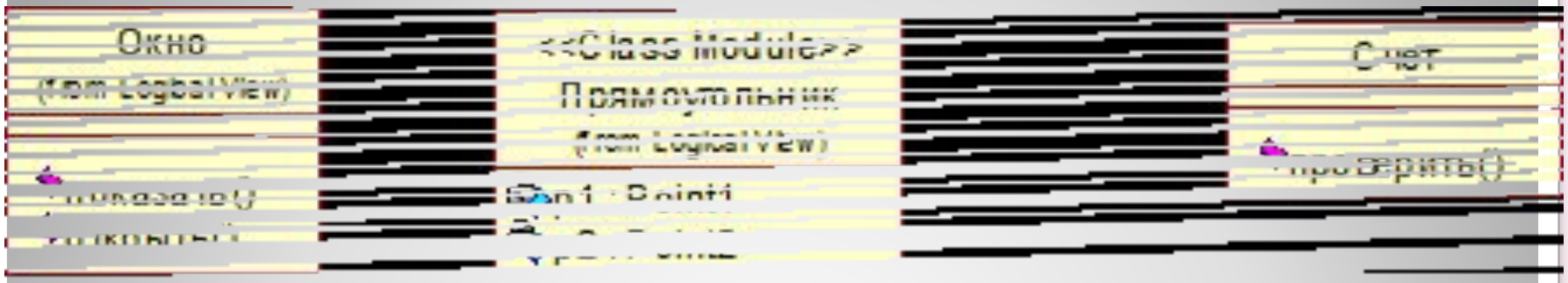
Во второй сверху секции прямоугольника класса записываются его атрибуты (attributes) или свойства. В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения.

Имя атрибута *представляет собой строку текста, которая используется в качестве идентификатора. Имя атрибута является единственным обязательным элементом синтаксического обозначения атрибута.*



Кратность *атрибута характеризует общее количество конкретных атрибутов данного типа, входящих в состав отдельного класса. В общем случае кратность записывается в форме строки текста в квадратных скобках после имени соответствующего атрибута:* ***[нижняя_граница.. верхняя граница],***

где нижняя__граница и верхняя_граница являются положительными целыми числами



Операция

В нижней секции прямоугольника записываются операции или методы класса. ***Операция*** *(****operation****) представляет собой некоторый сервис, представляющий каждый экземпляр класса по определенному требованию, совокупность операций характеризует функциональный аспект поведения класса. Запись операций класса в языке UML также стандартизована и подчиняется определенным синтаксическим правилам.*

Имя операции *представляет собой строку текста, которая используется в качестве идентификатора соответствующей операции и обычно является глаголом.*

Отношения между классами:

Отношение зависимости (dependency relationship).

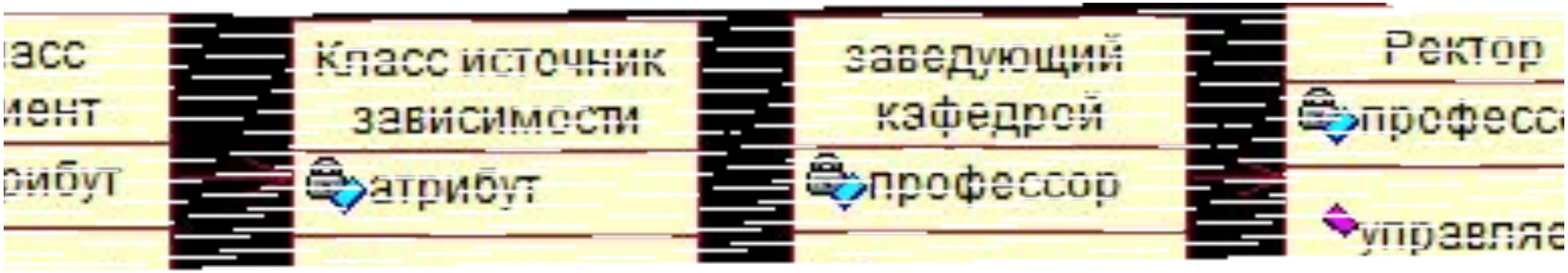
Отношение ассоциации (association relationship).

Отношение обобщения (generalization relationship).

Отношение реализации (realization relationship).

Отношение зависимости *в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели.*

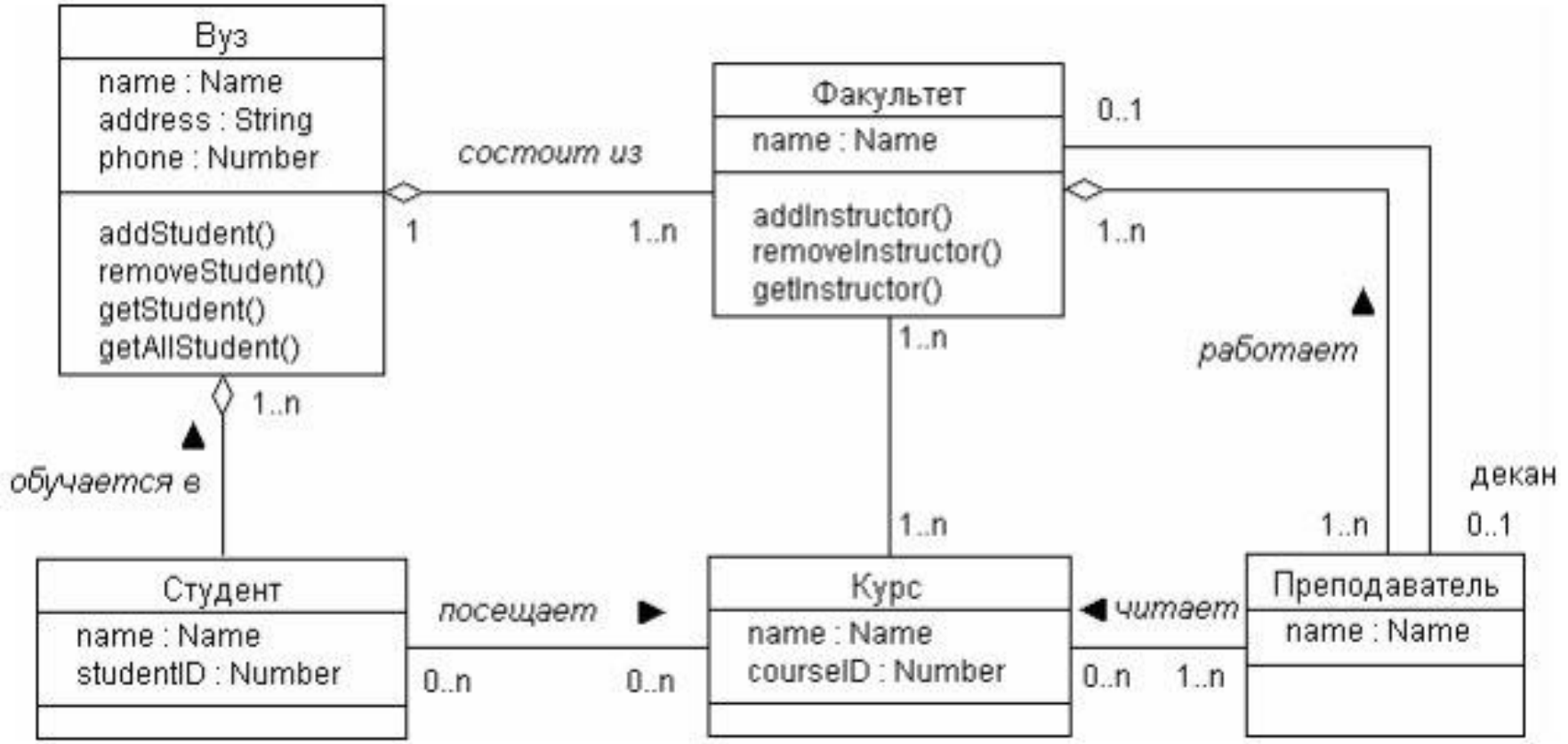
Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов ("à" или "ß"). При этом стрелка направлена от класса-клиента зависимости к независимому классу или классу-источнику. На данном рисунке изображены два класса: Класс_А и Класс_В, при этом Класс_В является источником некоторой зависимости, а Класс_А - клиентом этой зависимости.


Класс источник зависимости
атрибут
заведующий кафедрой
профессор
Ректор

Отношение ассоциации *соответствует наличию некоторого отношения между классами. Данное отношение обозначается сплошной линией с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации. В качестве дополнительных специальных символов могут использоваться имя ассоциации, а также имена и кратность классов-ролей ассоциации. Имя ассоциации является необязательным элементом ее обозначения. Если оно задано, то записывается с заглавной (большой) буквы рядом с линией соответствующей ассоциации.*

Отношение агрегации *имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.*
Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа "часть-целое". Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой незакрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой "целое". Остальные классы являются его "частями".

Отношение обобщения *является обычным таксономическим отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).*
При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка. На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс (класс-предок или суперкласс), а ее отсутствие — на более специальный класс (класс-потомок или подкласс).


Вуз
name : Name
address : String
phone : Number
addStudent()
removeStudent()
getStudent()
getAllStudent()
состоит из
1
1..n
Факультет
name : Name
addInstructor()
removeInstructor()
getInstructor()
0..1
1..n
1..n
работает
1..n
обучается в
декан
0..1
1..n
Студент
name : Name
studentID : Number
посещает
0..n
0..n
Курс
name : Name
courseID : Number
читает
0..n
1..n
Преподаватель
name : Name

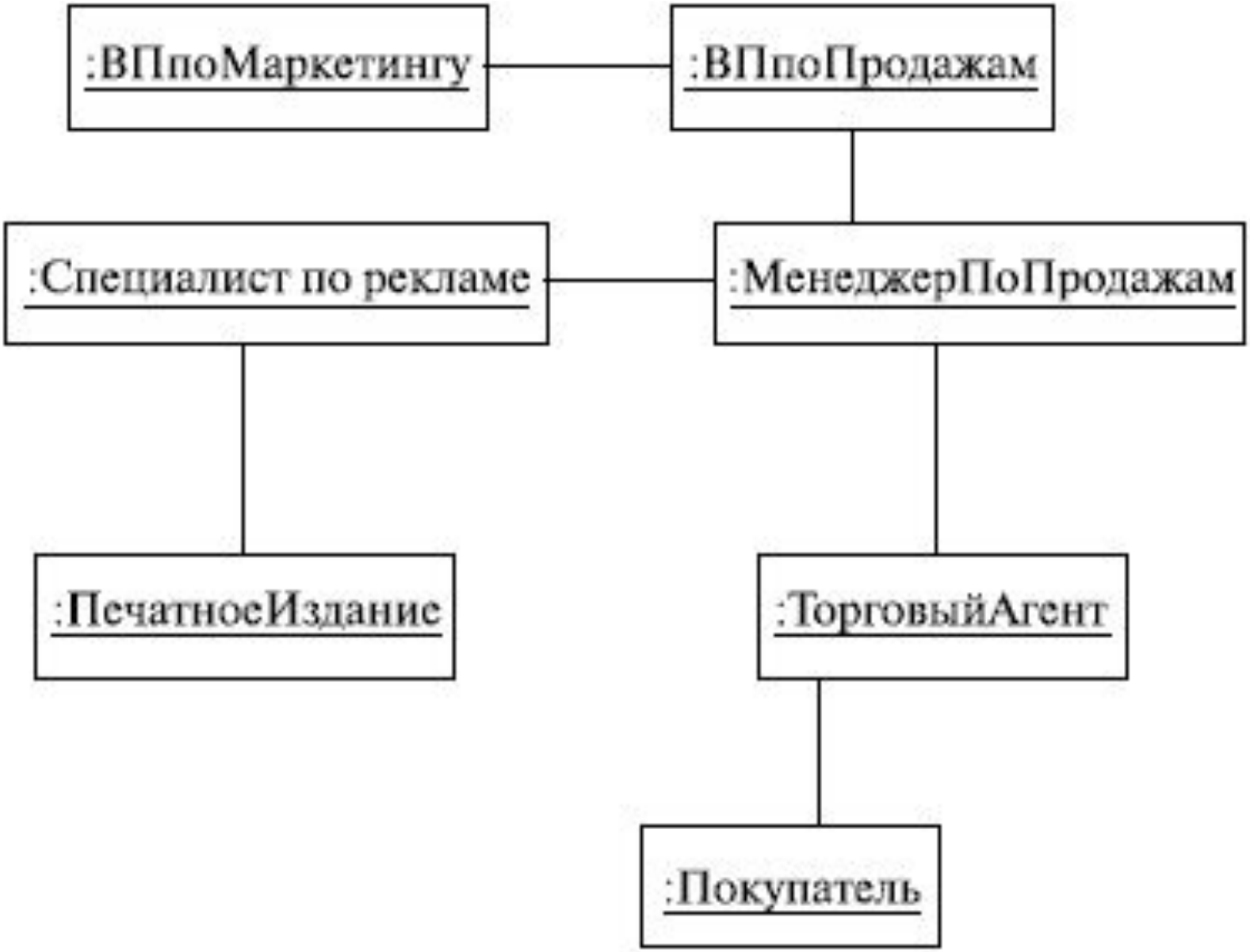
2. Диаграмма объектов

На ***диаграмме объектов*** показывают множество объектов и отношения между ними. Такие изображения используются для иллюстрации структуры данных, то есть статических "мгновенных снимков" экземпляров тех сущностей, которые представлены на диаграмме классов. Диаграммы объектов, так же как и диаграммы классов, относятся к статическому виду системы с точки зрения процессов, но заостряют внимание на реальных или модельных прецедентах.

Диаграмма объектов *- это, по существу, экземпляр диаграммы классов или статическая часть диаграммы взаимодействия. В любом случае она содержит прежде всего объекты и связи и акцентирует внимание на конкретных экземплярах или экземплярах-прототипах. Диаграммы компонентов и развертывания также могут содержать экземпляры, причем если они не содержат ничего другого (в частности, сообщений), то могут рассматриваться как частные случаи диаграммы объектов.*

Типичные примеры применения

С помощью диаграмм объектов, как и с помощью диаграмм классов, моделируют статический вид системы с точки зрения проектирования или процессов, но принимая во внимание реальные экземпляры или прототипы. Этот вид описывает главным образом функциональные требования к системе, то есть услуги, которые она должна предоставлять конечным пользователям. Диаграммы объектов позволяют моделировать статические структуры данных.


:ВПпоМаркетингу
:ВПпоПродажам
:Специалист по рекламе
:МенеджерПоПродажам
:ПечатноеИздание
:ТорговыйАгент
:Покупатель

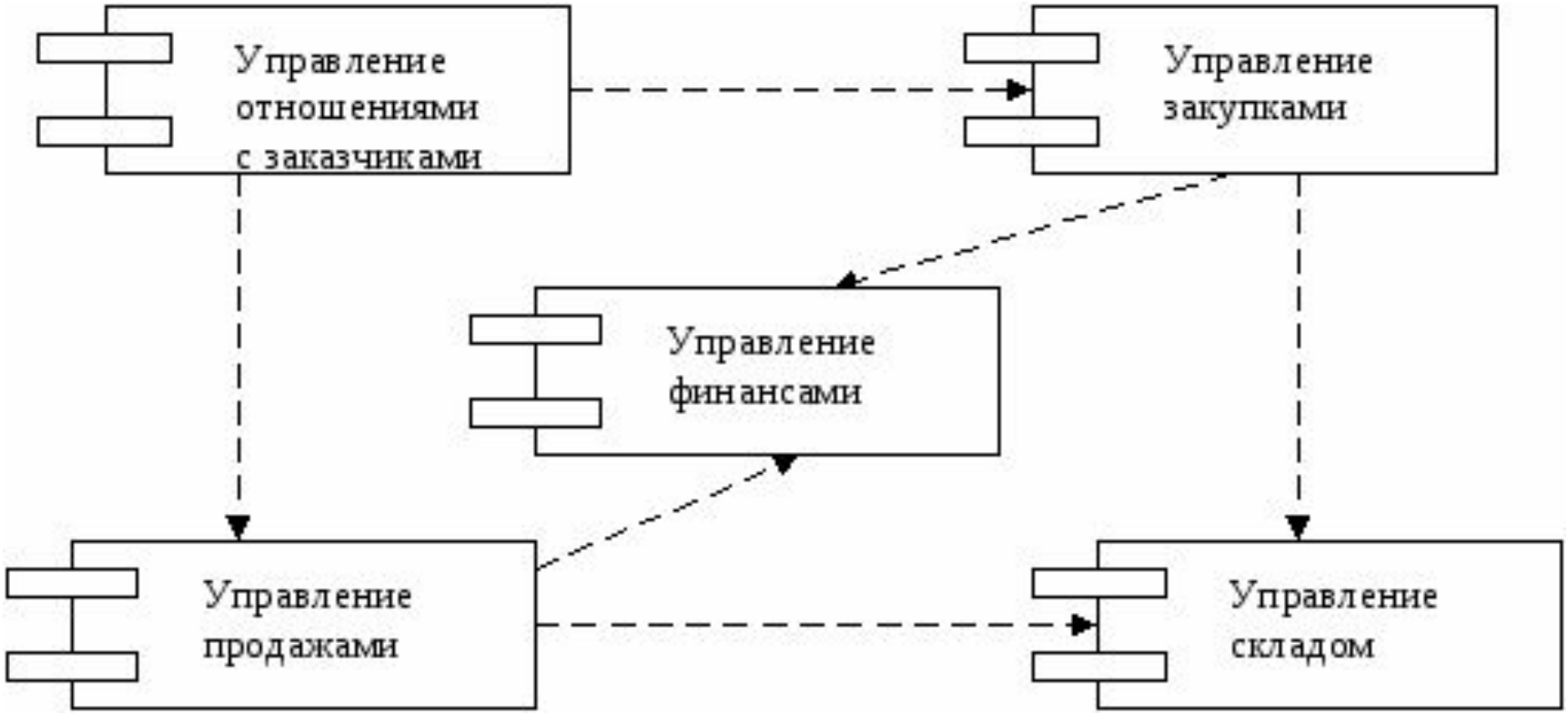
3. Диаграмма компонентов

На ***диаграммах компонентов*** показаны множества компонентов и отношения между ними. С их помощью иллюстрируют статический вид системы с точки зрения реализации.

Диаграммы компонентов соотносятся с диаграммами классов, так как обычно компонент отображается на один или несколько классов, интерфейсов или кооперацией.

Типичные примеры применения

Диаграммы компонентов используются для моделирования статического вида системы с точки зрения реализации. Этот вид в первую очередь связан с управлением конфигурацией частей системы, составленной из компонентов, которые можно соединять между собой различными способами.


Управление отношениями с заказчиками
Управление закупками
Управление финансами
Управление продажами
Управление складом

4. Диаграмма развертывания

На ***диаграммах развертывания*** представлены узлы и отношения между ними. С помощью таких изображений иллюстрируют статический вид системы с точки зрения развертывания. Они соотносятся с диаграммами компонентов, так как узел обычно содержит один или несколько компонентов.

Типичное применение

Диаграммы развертывания используются для моделирования статического вида системы с точки зрения развертывания.

Это представление в первую очередь обращено на распределение, поставку и установку частей, из которых состоит физическая система.

Фрагмент диаграммы развертывания

Пример диаграммы развертывания

Диаграммы поведения

Диаграммы поведения в UML условно разделяются на пять типов в соответствии с основными способами моделирования динамики системы:

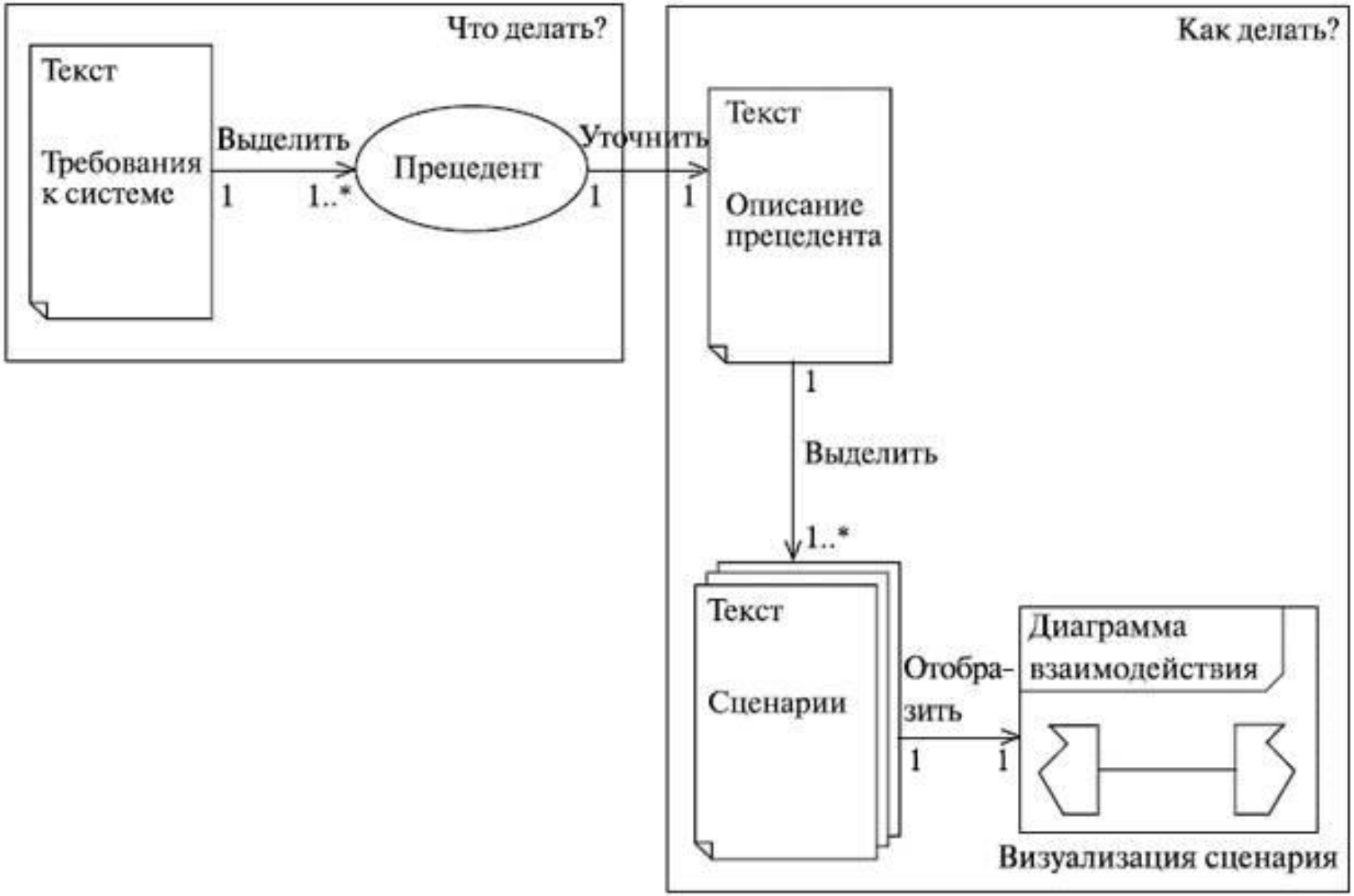
- диаграммы прецедентов описывают организацию поведения системы;
- диаграммы последовательностей акцентируют внимание на временной упорядоченности сообщений;
- диаграммы кооперации сфокусированы на структурной организации объектов, посылающих и получающих сообщения;
- диаграммы состояний описывают изменение состояния системы в ответ на события;
- диаграммы деятельности демонстрируют передачу управления от одной деятельности к другой.

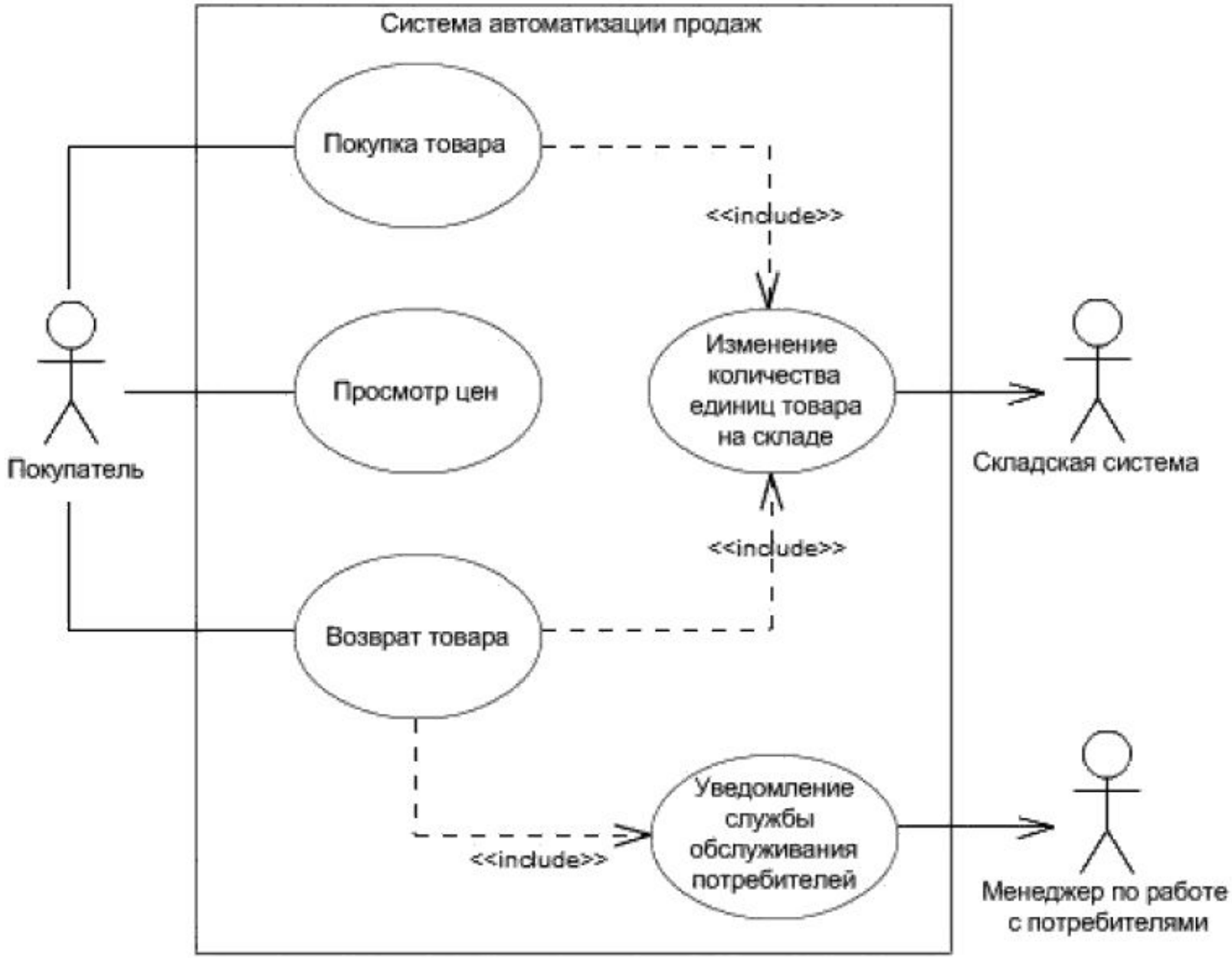
Диаграммы поведения

1. Диаграммы прецендентов

На *диаграммах прецедентов* показывается совокупность вариантов использования (прецедентов), актеров (частный случай классов) и отношений между ними.

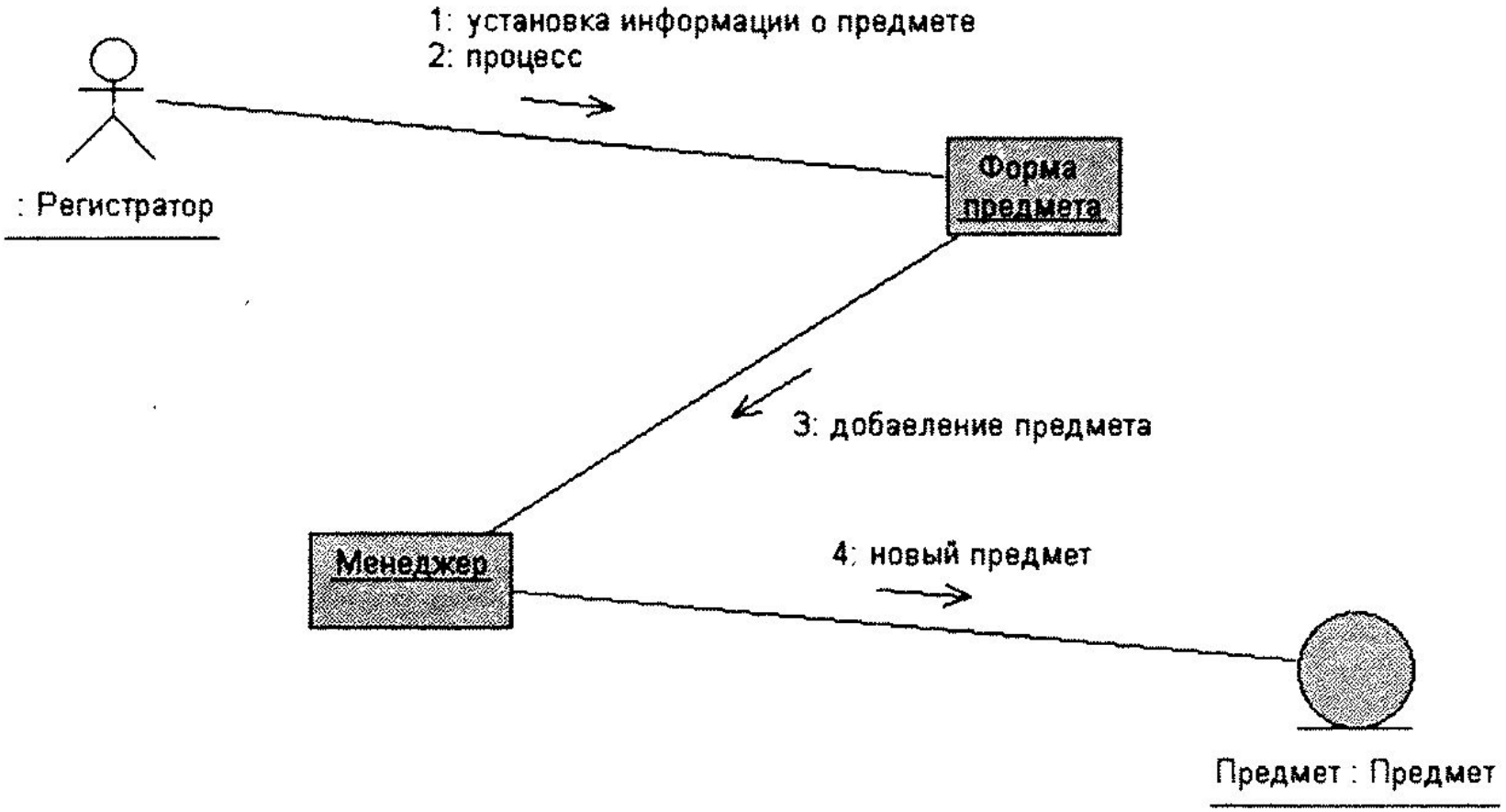
С помощью таких диаграмм иллюстрируют статический вид системы с точки зрения прецедентов, что особенно важно для ее организации и моделирования ее поведения.


Что делать?
Текст
Требования к системе
Выделить
1
1..*
Прецедент
Уточнить
1
Как делать?
Текст
Описание прецедента
1
Выделить
1..*
Текст
Сценарии
Отобра-зить
1
1
Диаграмма взаимодействия
Визуализация сценария


Система автоматизации продаж
Покупка товара
<<include>>
Просмотр цен
Изменение количества единиц товара на складе
Складская система
Покупатель
<<include>>
Возврат товара
Уведомление службы обслуживания потребителей
<<include>>
Менеджер по работе с потребителями

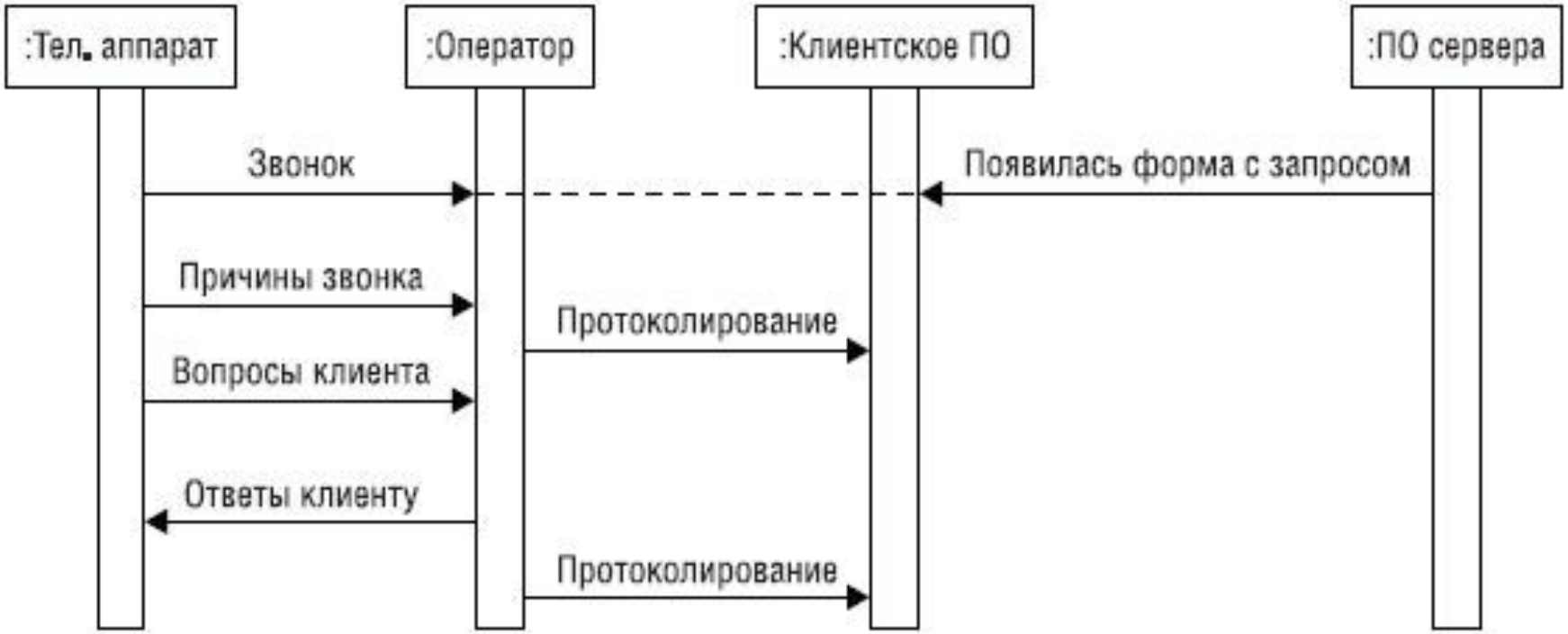
Диаграммы взаимодействий

Диаграммы взаимодействий - это общее наименование диаграмм последовательностей и кооперации. Любая диаграмма последовательностей или кооперации является диаграммой взаимодействия, а каждая диаграмма взаимодействия - это либо диаграмма последовательностей, либо диаграмма кооперации.


1: установка информации о предмете
2: процесс
: Регистратор
Форма предмета
3: добавление предмета
Менеджер
4: новый предмет
Предмет : Предмет

2. Диаграммы последовательностей

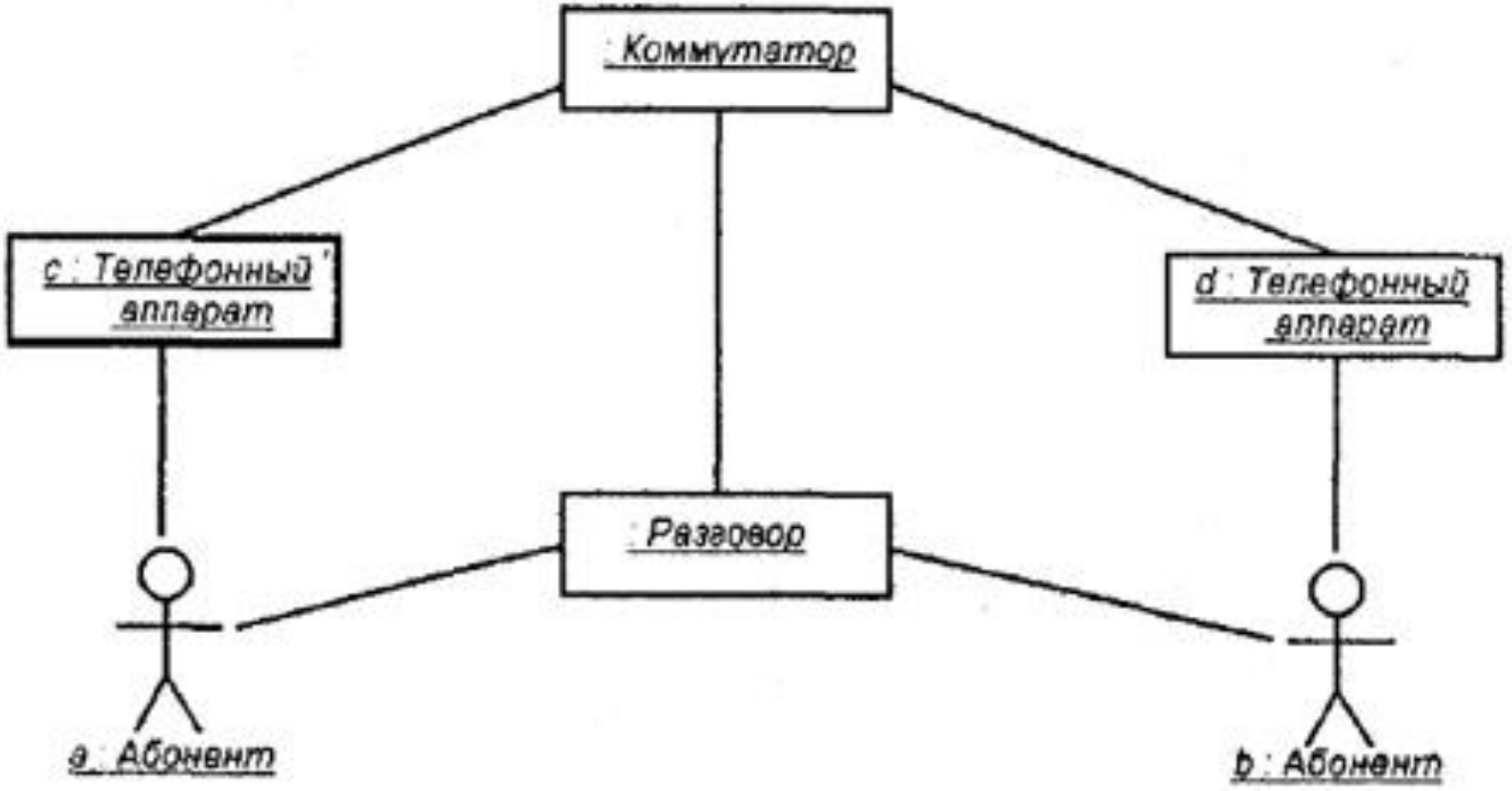
На ***диаграмме последовательностей*** основное внимание уделяется временной упорядоченности событий. На них изображают множество объектов и посланные или принятые ими сообщения. Объекты, как правило, представляют собой анонимные или именованные экземпляры классов, но могут быть также экземплярами других сущностей, таких как кооперации, компоненты или узлы. Диаграммы последовательностей относятся к динамическому виду системы.


:Тел. аппарат
:Оператор
:Клиентское ПО
:ПО сервера
Звонок
Появилась форма с запросом
Причины звонка
Протоколирование
Вопросы клиента
Ответы клиенту
Протоколирование

3. Диаграммы кооперации

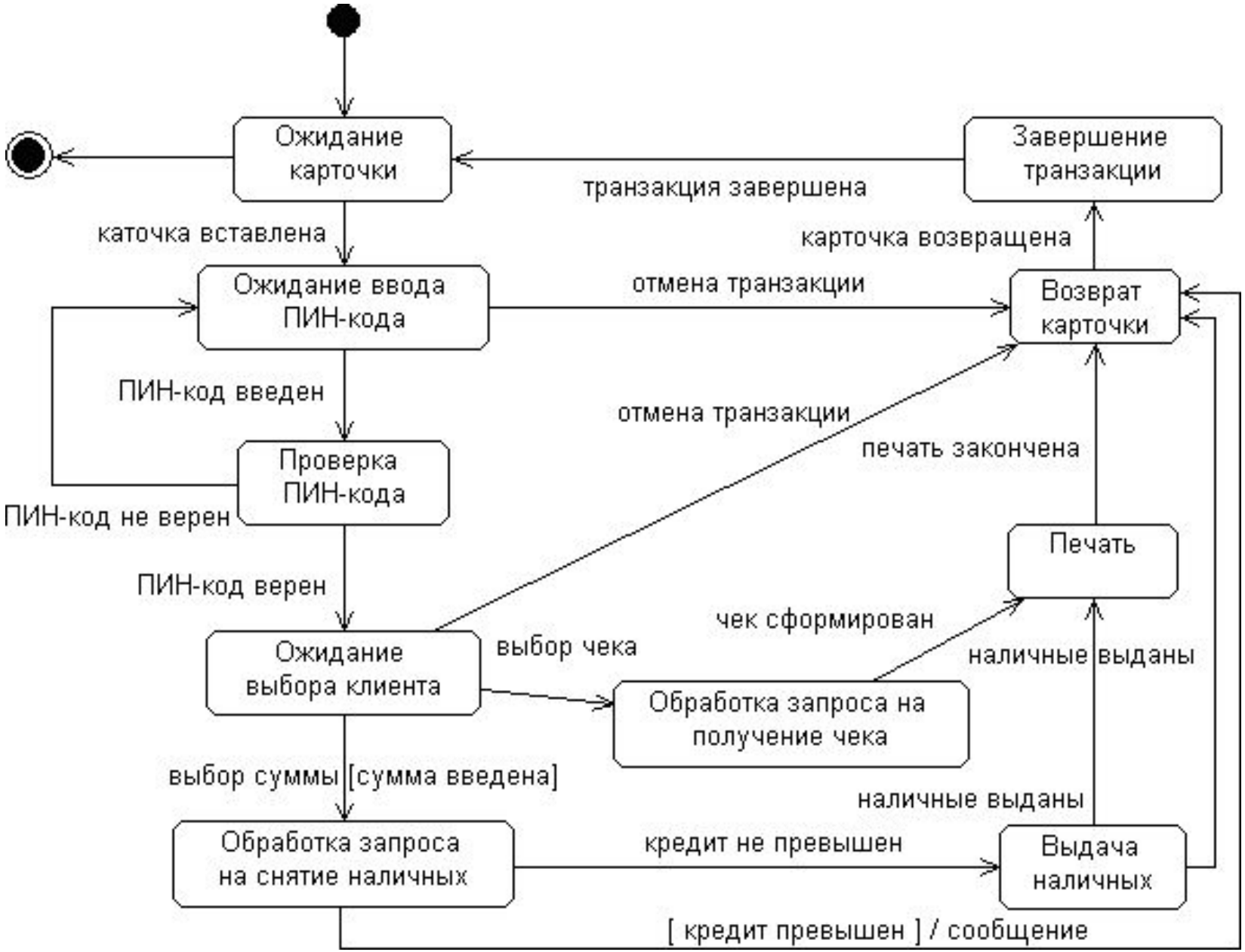
Диаграммы кооперации заостряют внимание на структурной организации объектов, принимающих или отправляющих сообщения. На диаграмме кооперации показано множество объектов, связи между ними и сообщения, которые они посылают или получают. Объекты обычно представляют собой анонимные или именованные экземпляры классов, но могут быть также экземплярами других сущностей, например кооперацией, компонентов и узлов. Диаграммы кооперации относятся к динамическому виду системы.

НАЧАЛЬНЫЙ ФРАГМЕНТ ДИАГРАММЫ КООПЕРАЦИИ



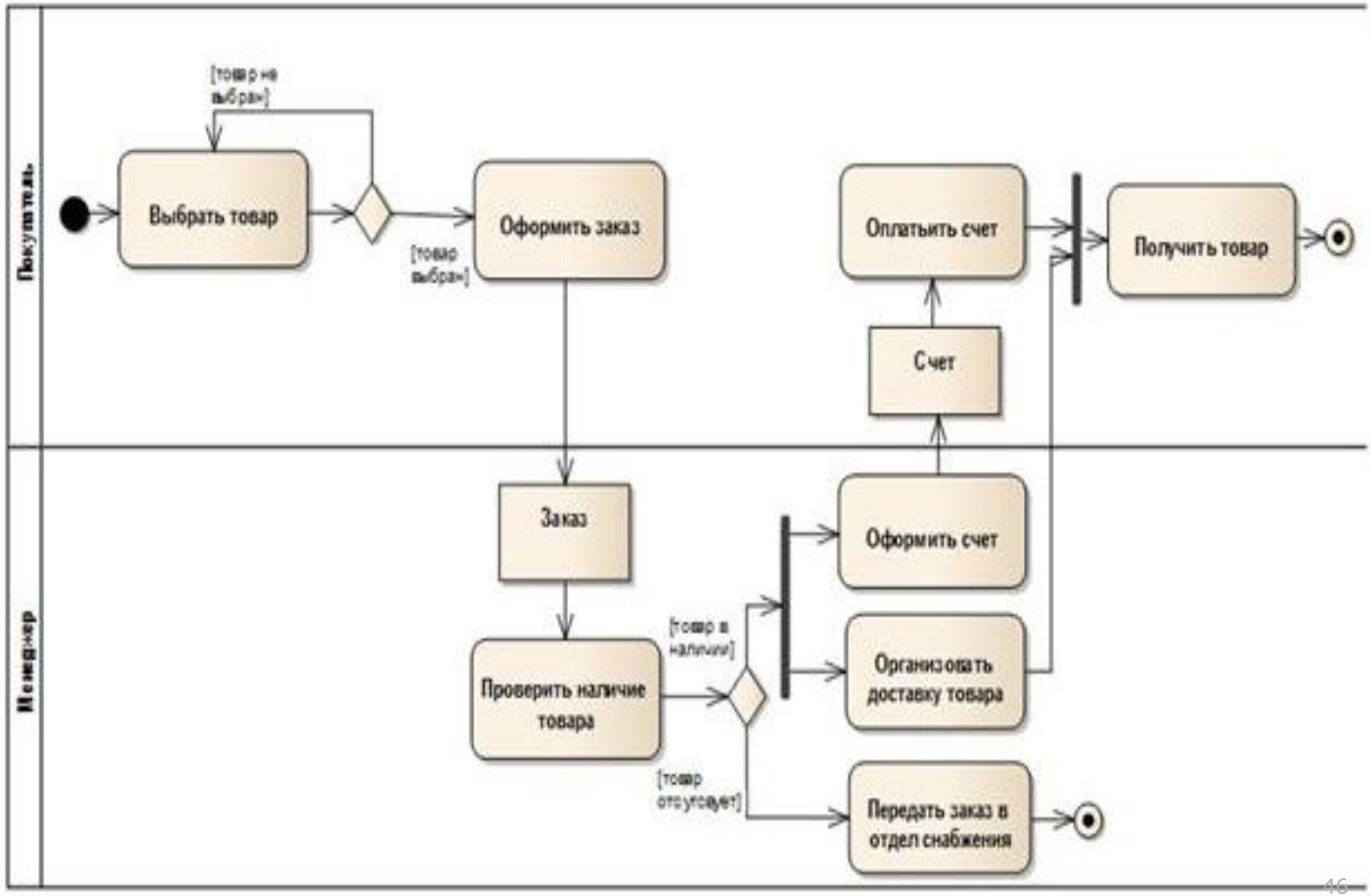
4. Диаграммы состояний

Диаграмма состояний показывает автомат, содержащий состояния, переходы, события и действия. Диаграммы такого рода относятся к динамическому виду системы и особенно важны при моделировании поведения интерфейса, класса или кооперации. Основное внимание в них уделяется порядку возникновения событий, связанных с объектом, что особенно полезно при моделировании реактивных систем.


Ожидание карточки
Завершение транзакции
транзакция завершена
каточка вставлена
карточка возвращена
Ожидание ввода ПИН-кода
отмена транзакции
Возврат карточки
ПИН-код введен
отмена транзакции
печать закончена
Проверка ПИН-кода
ПИН-код не верен
Печать
ПИН-код верен
чек сформирован
Ожидание выбора клиента
выбор чека
наличные выданы
Обработка запроса на получение чека
выбор суммы [сумма введена]
наличные выданы
Обработка запроса на снятие наличных
кредит не превышен
Выдача наличных
[кредит превышен] / сообщение

5. Диаграммы деятельности

На ***диаграммах деятельности*** изображают передачу управления от одной деятельности к другой внутри системы. На них показаны виды деятельности, последовательные или параллельные ветвления потока управления и объекты, которые воздействуют на что- то или сами подвергаются воздействию. Диаграммы деятельности относятся к динамическому представлению системы и особенно важны при моделировании ее функций. Они являются особой разновидностью диаграмм состояния. На диаграммах деятельности основное внимание уделено потоку управления между объектами.


Покупатель
Менеджер
[товар не выбран]
Выбрать товар
[товар выбран]
Оформить заказ
Оплатить счет
Получить товар
Счет
Заказ
Оформить счет
[товар в наличии]
Проверить наличие товара
Организовать доставку товара
[товар отсутствует]
Передать заказ в отдел снабжения

Пять основных диаграмм поведения в UML используются для визуализации, специфицирования, конструирования и документирования динамических аспектов системы.

Динамические аспекты программной системы охватывают такие ее элементы, как поток сообщений во времени и физическое перемещение компонентов по сети.

Рекомендации построения UML диаграмм

1. Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области. В процессе разработки диаграммы крайне важно учесть все сущности, важные в контексте данной диаграммы. Отсутствие существенных элементов на диаграмме служит признаком неполноты модели и может потребовать ее последующей доработки.

2. Все сущности на диаграмме должны принадлежать одному концептуальному уровню представления модели. Отдельные фрагменты диаграммы могут детализироваться на других диаграммах этого же типа, образуя вложенные или подчиненные диаграммы. Таким образом, модель системы на языке UML представляет собой пакет иерархически вложенных диаграмм, детализация которых должна быть достаточной для последующей генерации программного кода, реализующего проект соответствующей системы.

3. Необходимо стремиться к явному указанию свойств всех элементов диаграмм, несмотря на то, что язык UML допускает использование значений по умолчанию при отсутствии некоторых символов на диаграмме (к примеру, в случае неявного указания видимости атрибутов и операций классов).

4. Диаграммы не должны содержать противоречивой информации. Противоречивость модели приводит к неоднозначной ее интерпретации и может служить источником проблем при реализации (к примеру, наличие замкнутых путей при изображении отношений агрегирования или композиции, наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен).

5. Не следует перегружать диаграммы текстовой информацией - визуализация модели является наиболее эффективной, в случае если она содержит минимум пояснительного текста. Как правило, наличие больших фрагментов текста на диаграмме служит признаком неоднородности модели, когда в рамках одной модели представляется различная по характеру информация. Набор доступных разработчику UML-диаграмм способен удовлетворить самые детальные представления о системе - важно уметь правильно отображать те или иные сущности и аспекты моделирования в соответствующие элементы UML- диаграмм.

6. Состав диаграмм, используемых в конкретном программном проекте, не является строго фиксированным и зависит от специфики проекта: для простых приложений нет крайне важности строить все без исключения типы диаграмм, некоторые из них могут просто отсутствовать в проекте системы, и данный факт не будет считаться ошибкой разработчика. К примеру, модель системы может не содержать диаграмму развертывания для приложения, выполняемого локально на компьютере пользователя.

Процесс построения отдельных типов диаграмм имеет свои особенности, которые тесно связаны с семантикой элементов этих диаграмм. Сам процесс ООАП в контексте языка UML получил специальное название – рациональный унифицированный процесс
(Rational Unified Process, RUP).