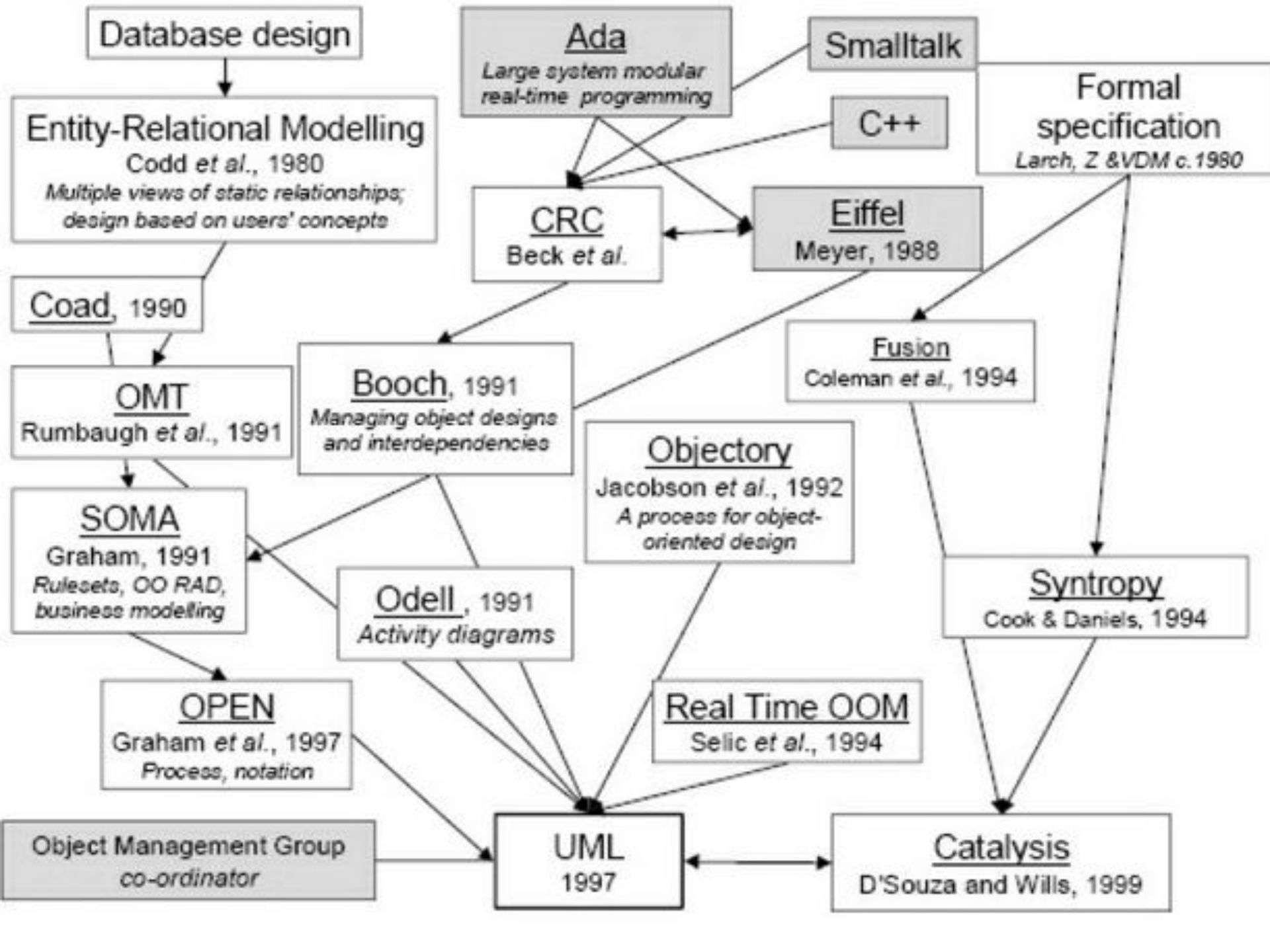
Что такое UML

Визуальное моделирование

UML (Unified Modeling Language) - унифицированный язык моделирования.

- **Язык** - система знаков, служащая:
- *средством человеческого общения и мыслительной деятельности;*
- *способом выражения самосознания личности;*
- *средством хранения и передачи информации.*

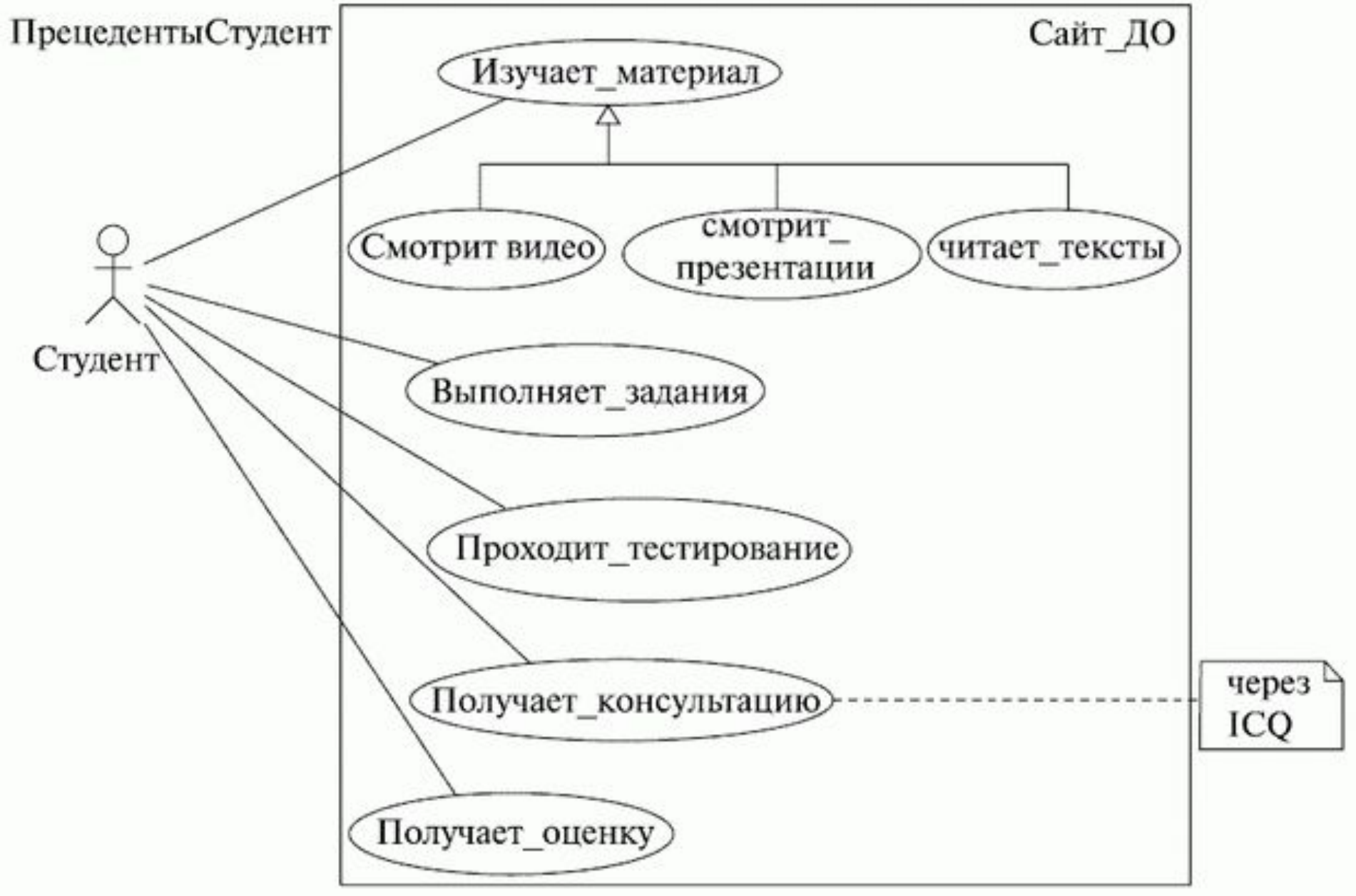

Database design
Ada
Large system modular real-time programming
Smalltalk
Formal specification
Larch, Z &VDM c.1980
C++
Entity-Relational Modelling
Codd et al., 1980
Multiple views of static relationships; design based on users' concepts
CRC
Beck et al.
Eiffel
Meyer, 1988
Coad, 1990
Fusion
Coleman et al., 1994
Booch, 1991
Managing object designs and interdependencies
OMT
Rumbaugh et al., 1991
Objectory
Jacobson et al., 1992
A process for object-oriented design
SOMA
Graham, 1991
Rulesets, OO RAD, business modelling
Odell, 1991
Activity diagrams
Syntropy
Cook & Daniels, 1994
OPEN
Graham et al., 1997
Process, notation
Real Time OOM
Selic et al., 1994
Object Management Group
co-ordinator
UML
1997
Catalysis
D'Souza and Wills, 1999

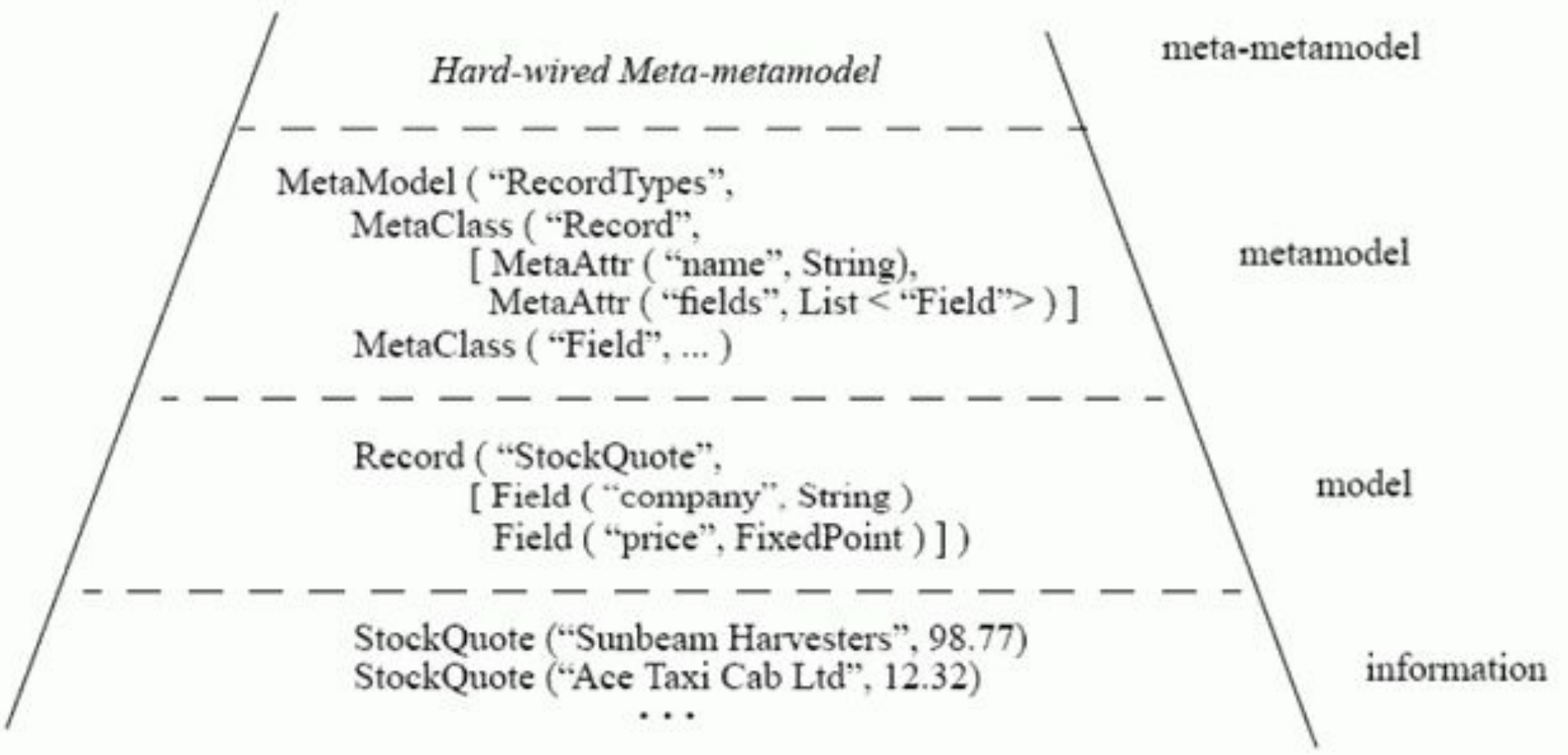

Мировые лидеры потребления кофе
Робуста
Арабика
Тропик Рака
Тропик Козерога
Производство кофе в мире:
Арабика
Мягкий вкус, тонкий аромат
70%
30%
Робуста
Менее ароматный, но содержит больше кофеина
1.
Зерна выкладываются на тонкий слой песка и сверху присыпаются еще одним слоем
2.
Через 10 дней появляются первые ростки
3.
Год спустя ростки высаживают на настоящую плантацию, но плодоносить дерево начинает только через 2 года
4.
Плоды собирают вручную, выбирая самые спелые ягоды
5.
В течение 15 дней ягоды сушат под лучами Солнца
6.
Ночью их собирают в кучи и укрывают тканью, чтобы они не впитывали влагу и не мокли под дождем
7.
Высушенные плоды отшелушивают, удаляя все внешние оболочки и очищают потоком сжатого воздуха
8.
Дорогие сорта кофе сортируют исключительно вручную. Кофе дешевых сортов сортируется механически
10.
обжарка
Степень:
Вкус:
1 — Кисловатый
2 — Кисловато-вкусный
3 — Горьковатый
4 — Горький
11.
помол
Самый лучший кофе — арабика, выращенная на плантациях Кении, Колумбии, Танзании
Кофе высокого качества — арабика с плантаций Центральной Америки и некоторых регионов Азии
Самые дешевые сорта — арабика с плантаций Бразилии и некоторых стран этого региона
Как правильно заваривать кофе в френч-прессе
1. Ополоснуть колбу горячей водой
2. Засыпать кофе 3 ст.л. на 200 мл воды, залить горячей водой
3. Перемешать
4. Настаивать в течение 4–6 минут
5. Отпустить фильтр вниз
Чем дольше настаивается кофе, тем больший кофеина в готовом напитке
Френч-пресс
Гейзерная кофеварка
Фильтровая кофеварка
Эспрессо-машина
Турка
COFFEE
Coffee

Итак, UML в первую очередь – это спецификации.

Спецификация - подробное описание системы, которое полностью определяет ее цель и функциональные возможности. Различают:

- *словесные спецификации на естественном языке;*
- *модельные спецификации;*
- *формальные спецификации.*


ПрецедентыСтудент
Сайт_ДО
Изучает_материал
Смотрит видео
смотрит_
презентации
читает_тексты
Студент
Выполняет_задания
Проходит_тестирование
Получает_консультацию
через
ICQ
Получает_оценку


Hard-wired Meta-metamodel
meta-metamodel
MetaModel ("RecordTypes",
MetaClass ("Record",
[MetaAttr ("name", String),
MetaAttr ("fields", List < "Field">)]
MetaClass ("Field", ...)
metamodel
Record ("StockQuote",
[Field ("company", String)
Field ("price", FixedPoint)])
model
StockQuote ("Sunbeam Harvesters", 98.77)
StockQuote ("Ace Taxi Cab Ltd", 12.32)
. . .
information

- UML - еще один формальный язык, который необходимо освоить каждому, кто собирается заниматься программной инженерией.
- Само собой разумеется, что знание UML не гарантирует построения разумных и понятных моделей, хотя и является для этого необходимым.
- UML предоставляет огромную свободу при рисовании диаграмм и выборе инструмента рисования. Производители инструментов также воспользовались этой свободой, чтобы по своему разумению

Виды диаграмм

- DOMAIN CATEGORIES

Business Modeling	**14**	***C4i***	**8**	***Communications***	**6**	***Component Architecture***	**6**
Enterprise Modeling	2	Finance	9	Government	4	Healthcare	12
High Performance Computing	2	*Industrial Systems*	5	Information Security	1	Interoperability	1
Spase	4	*Lifesciences*	10	Manufacturing	5	Robotics	5
		Transport	2				

Виды диаграмм

- диаграмма прецедентов;
- диаграмма классов;
- диаграмма объектов;
- диаграмма последовательностей;
- диаграмма взаимодействия;
- диаграмма состояний;
- диаграмма активности;
- диаграмма развертывания.

Диаграмма прецедентов (use case diagram)

- ***Эктор (actor)*** *- это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Эктором может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.*
- Графически эктор изображается либо *человечком,* либо *символом класса с соответствующим стереотипом*

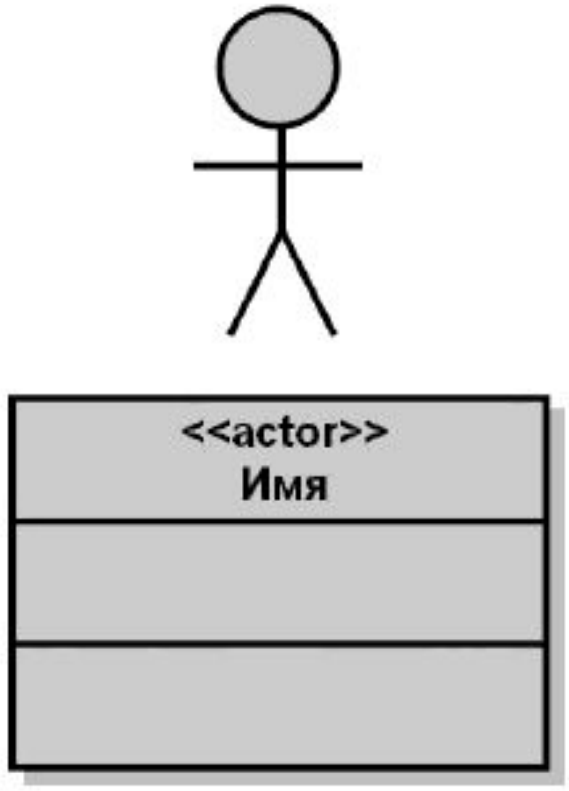


Диаграмма прецедентов (use case diagram)

- ***Прецедент (use-case)*** *- описание отдельного аспекта поведения системы с точки зрения пользователя (Буч).*
- Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название (рис.4.2). *Прецеденты и экторы соединяются с помощью линий*. Часто на одном из концов линии изображают *стрелку*, причем *направлена она к тому, у кого запрашивают сервис*, другими словами, чьими услугами пользуются.
- Прецеденты могут включать другие прецеденты, расширяться ими,

Диаграмма прецедентов (use case diagram)

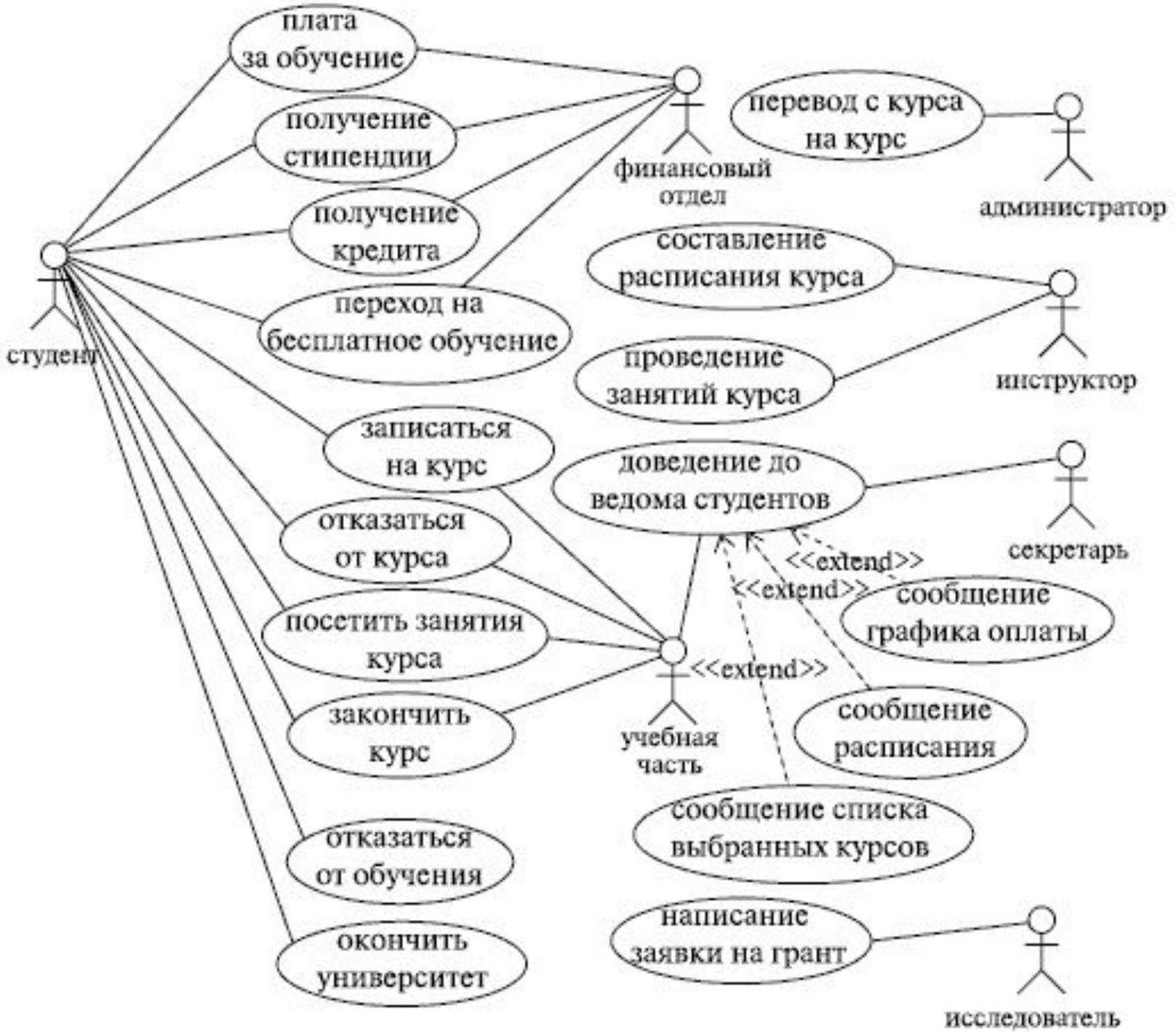

плата
за обучение
получение
стипендии
получение
кредита
переход на
бесплатное обучение
финансовый
отдел
перевод с курса
на курс
администратор
студент
составление
расписания курса
проведение
занятий курса
инструктор
записаться
на курс
доведение до
ведома студентов
секретарь
отказаться
от курса
<<extend>>
<<extend>>
сообщение
графика оплаты
посетить занятия
курса
<<extend>>
закончить
курс
учебная
часть
сообщение
расписания
сообщение списка
выбранных курсов
отказаться
от обучения
написание
заявки на грант
окончить
университет
исследователь

Диаграмма прецедентов (use case diagram)

- **Цели создания диаграмм прецедентов**:
- 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 системы.

Диаграмма классов (class diagram)

- ***Класс (class)*** *- категория вещей, которые имеют общие атрибуты и операции.*

Диаграмма классов - это *набор статических,* декларативных *элементов модели*.

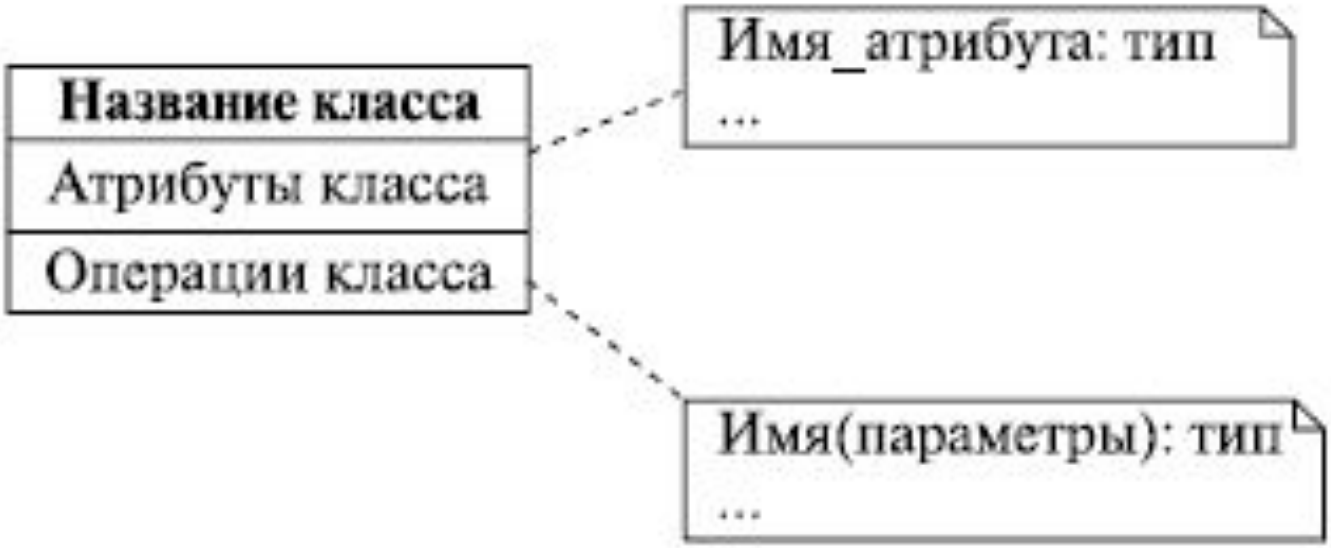
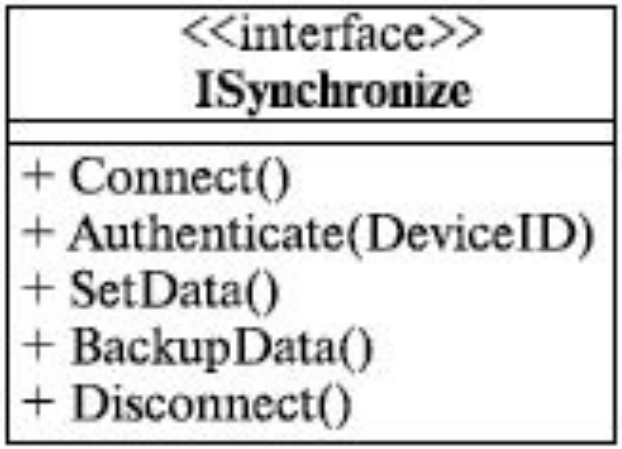


Диаграмма классов (class diagram)

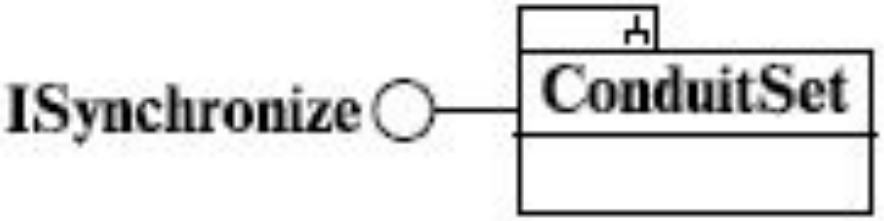
- Сокрытие от пользователя внутреннего устройства объектов называется *инкапсуляцией*.

Символ	Значение
+	public - открытый доступ
-	private - только из операций того же класса
#	protected - только из операций этого же класса и классов, создаваемых на его основе

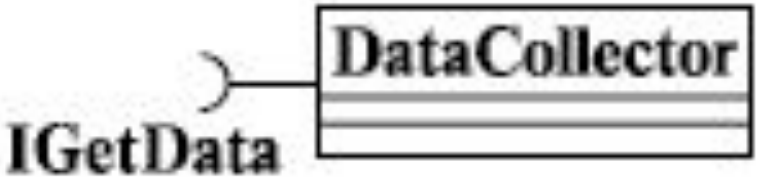
Изображение интерфейса



Класс со стереотипом <<interface>>



Изображение интерфейса



Изображение интерфейса

Символы предоставляемого и требуемого интерфейсов

Диаграмма классов (class diagram)

- **Обобщение** - это отношение между более общей сущностью, называемой *суперклассом*, и ее конкретным воплощением, называемым *подклассом*.
- Обобщение (или, как часто говорят, наследование) на диаграммах обозначается очень просто - незакрашенной треугольной стрелкой, направленной на суперкласс

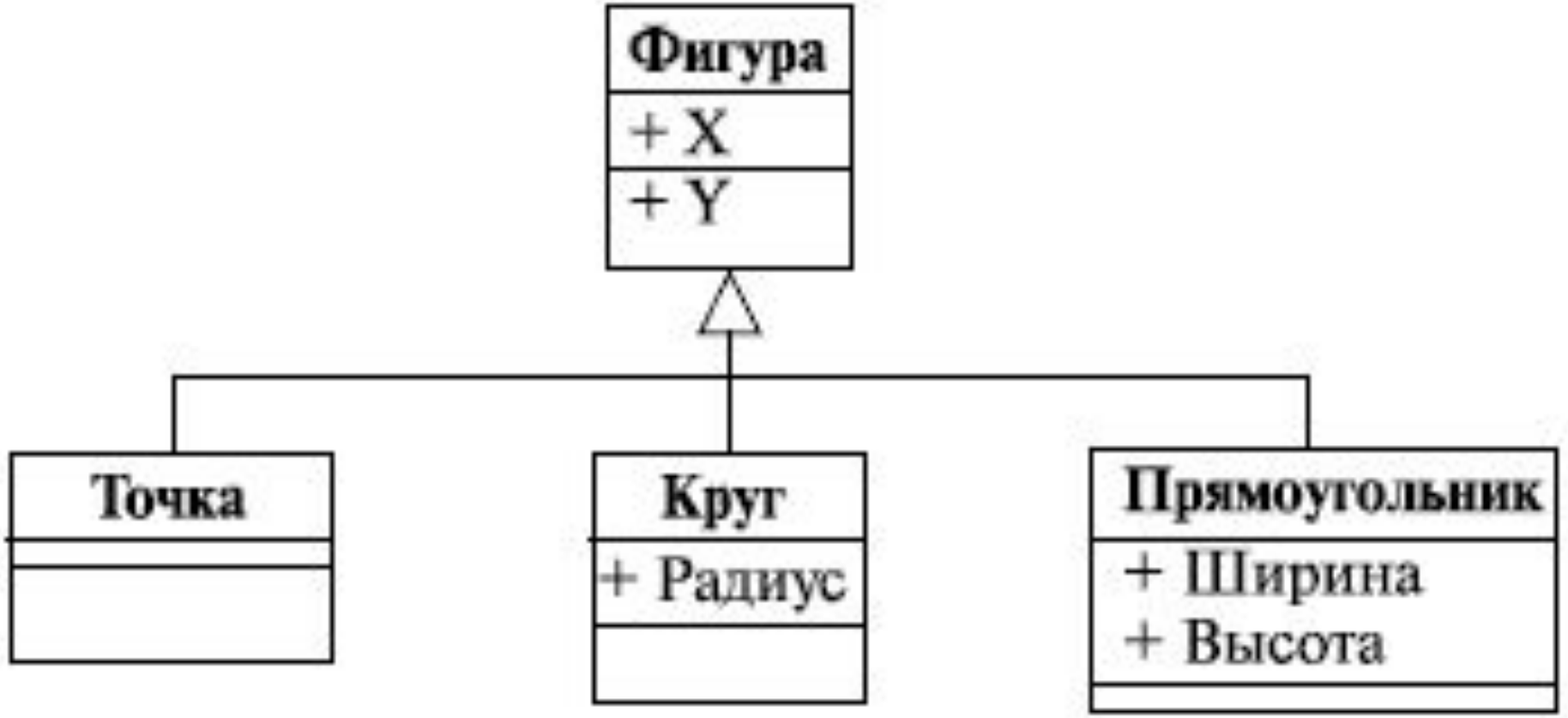
Обобщение

1. Найдите атрибуты, операции и обязанности, общие для двух или более классов из данной совокупности. Это позволит избежать ненужного дублирования структуры и функциональности объектов.

2. Вынесите эти элементы в некоторый общий суперкласс, а если такого не существует, то создайте новый класс.

3. Отметьте в модели, что подклассы наследуются от суперкласса, установив между ними отношение обобщения.

Пример использования обобщения

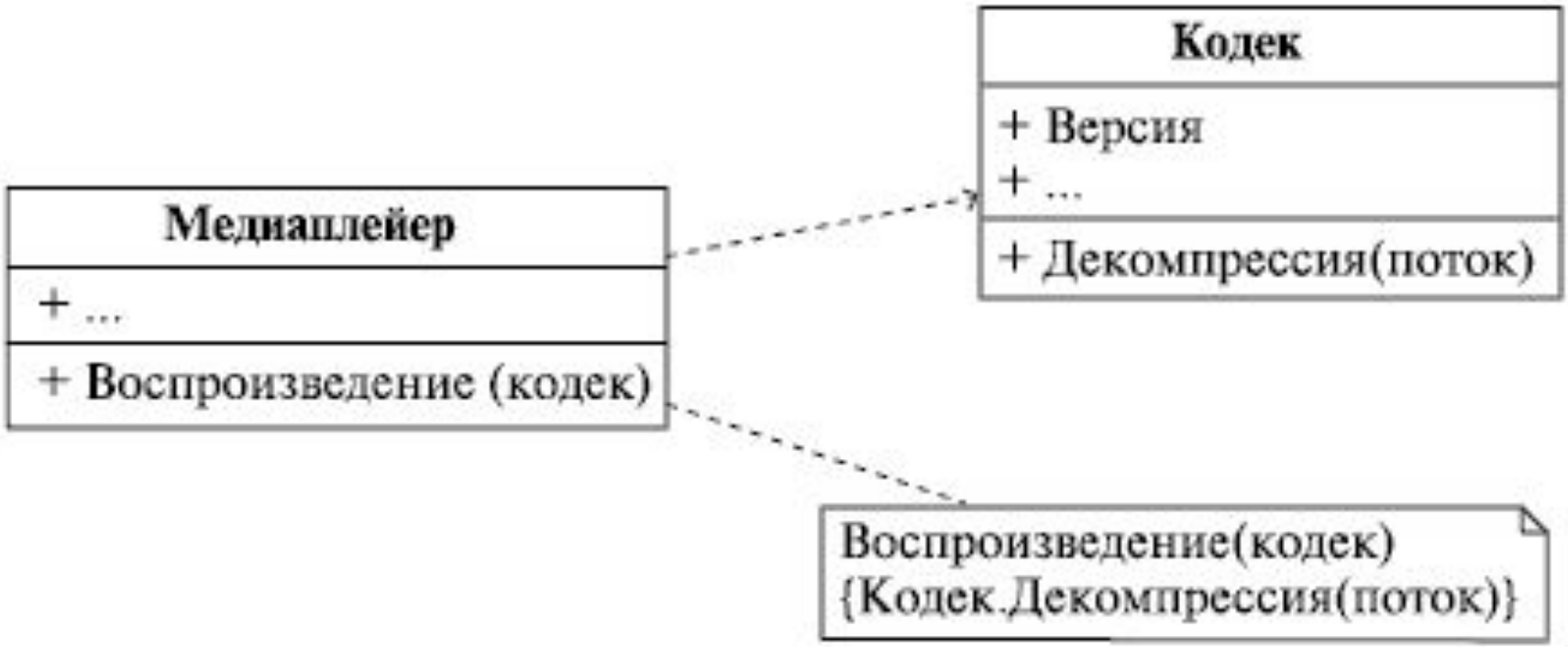


Полиморфизм

- **Полиморфи́зм** (в языках программирования) — возможность объектов с одинаковой спецификацией иметь различную реализацию.
- Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество реализаций».
- Общие свойства объектов объединяются в систему, которую могут называть по-разному — интерфейс, класс. Общность имеет внешнее и внутреннее выражение:
- внешняя общность проявляется как одинаковый набор методов с одинаковыми именами и сигнатурами (именем методов и типами аргументов и их количеством);
- внутренняя общность — одинаковая функциональность методов. Её можно описать интуитивно или выразить в виде строгих законов, правил, которым должны подчиняться методы. Возможность приписывать разную функциональность одному методу (функции, операции) называется *перегрузкой метода* (*перегрузкой функций*, *перегрузкой операций*).

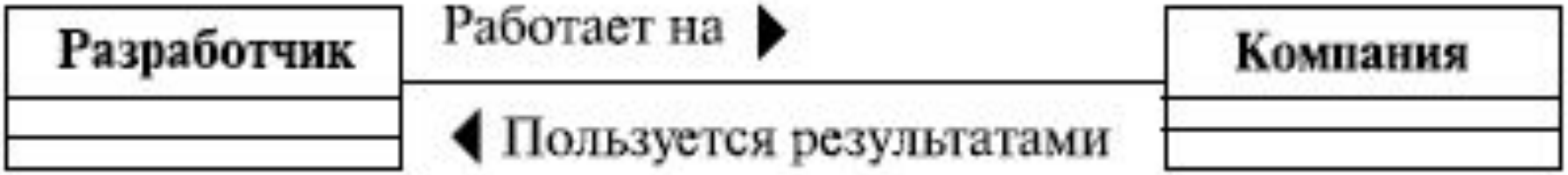
Отношения между классами

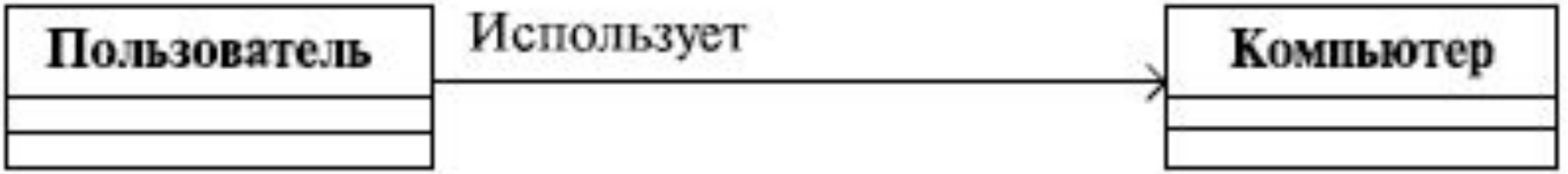
- **Зависимость** возникает тогда, когда реализация класса одного объекта зависит от спецификации операций класса другого объекта



Отношения между классами

- **Ассоциация** - Это просто связь между объектами, по которой можно между ними перемещаться. Ассоциация может иметь имя, показывающее природу отношений между объектами, при этом в имени может указываться *направление* чтения связи при помощи треугольного маркера.



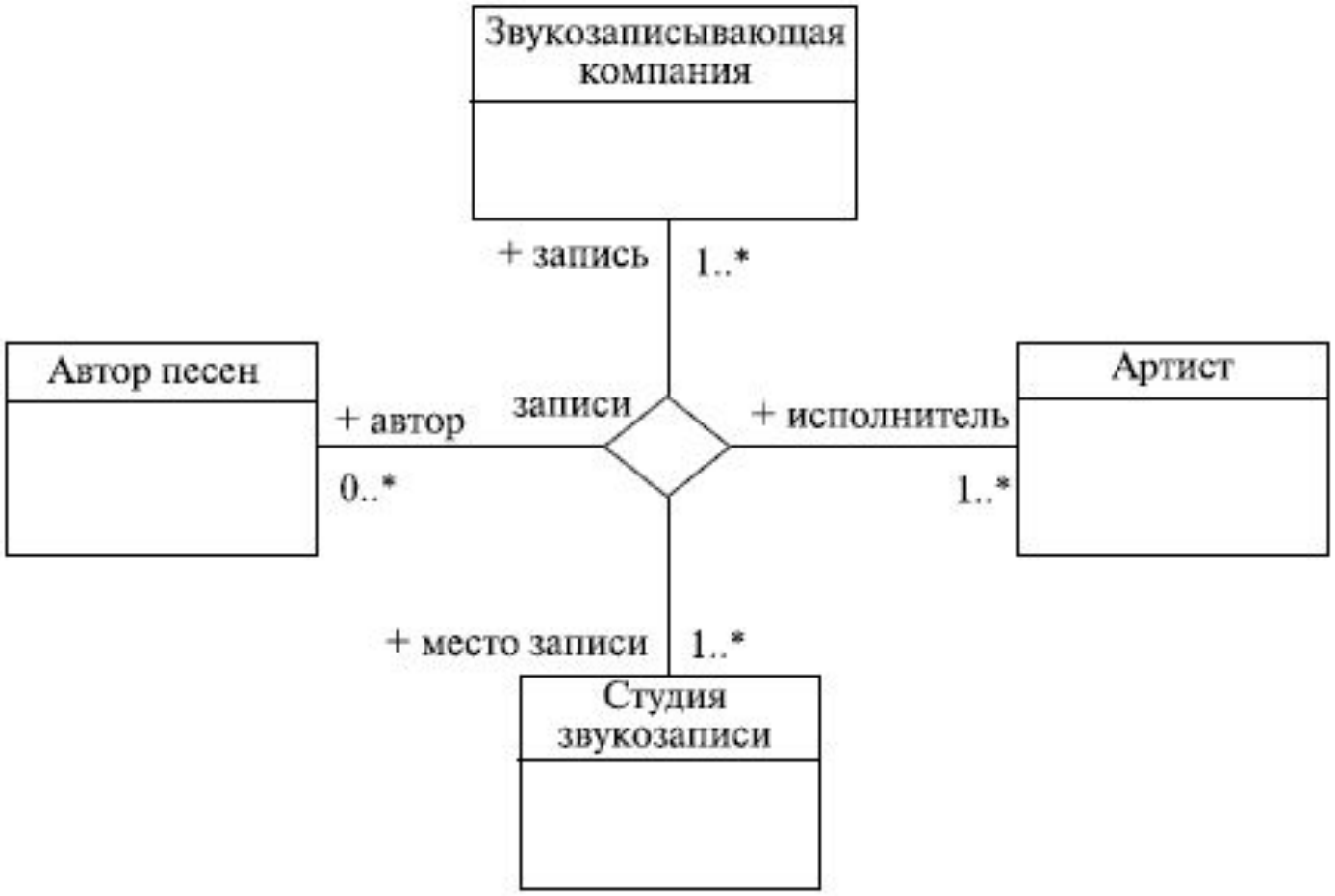


Отношения между классами

- Кроме направления ассоциации, мы можем указать на диаграмме *роли*, которые каждый класс играет в данном отношении, и *кратность*, то есть количество объектов, связанных отношением

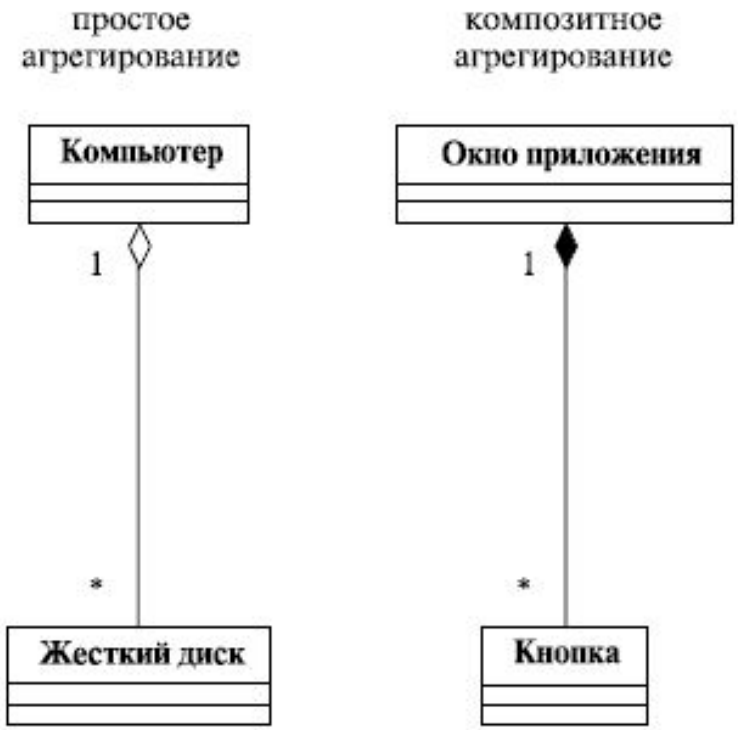
Отношения между классами

- Ассоциация может объединять три и более класса. В этом случае она называется **n-арной** и изображается ромбом на пересечении линий, как показано на этой диаграмме



Отношения между классами

Ассоциация - более сложное отношение между классами, например, связь типа "часть-целое". Такой вид ассоциации называется **ассоциацией с агрегированием**. В этом случае один класс имеет более высокий статус (целое) и состоит из низших по статусу классов (частей). При этом выделяют простое и композитное агрегирование и говорят о собственно **агрегации** и **композиции**. Простая агрегация предполагает, что части, отделенные от целого, могут продолжать свое существование независимо от него



Отношения между классами

В отношении между двумя классами сама ассоциация тоже может иметь свойства и, следовательно, тоже может быть представлена в виде класса.

Отношения между классами

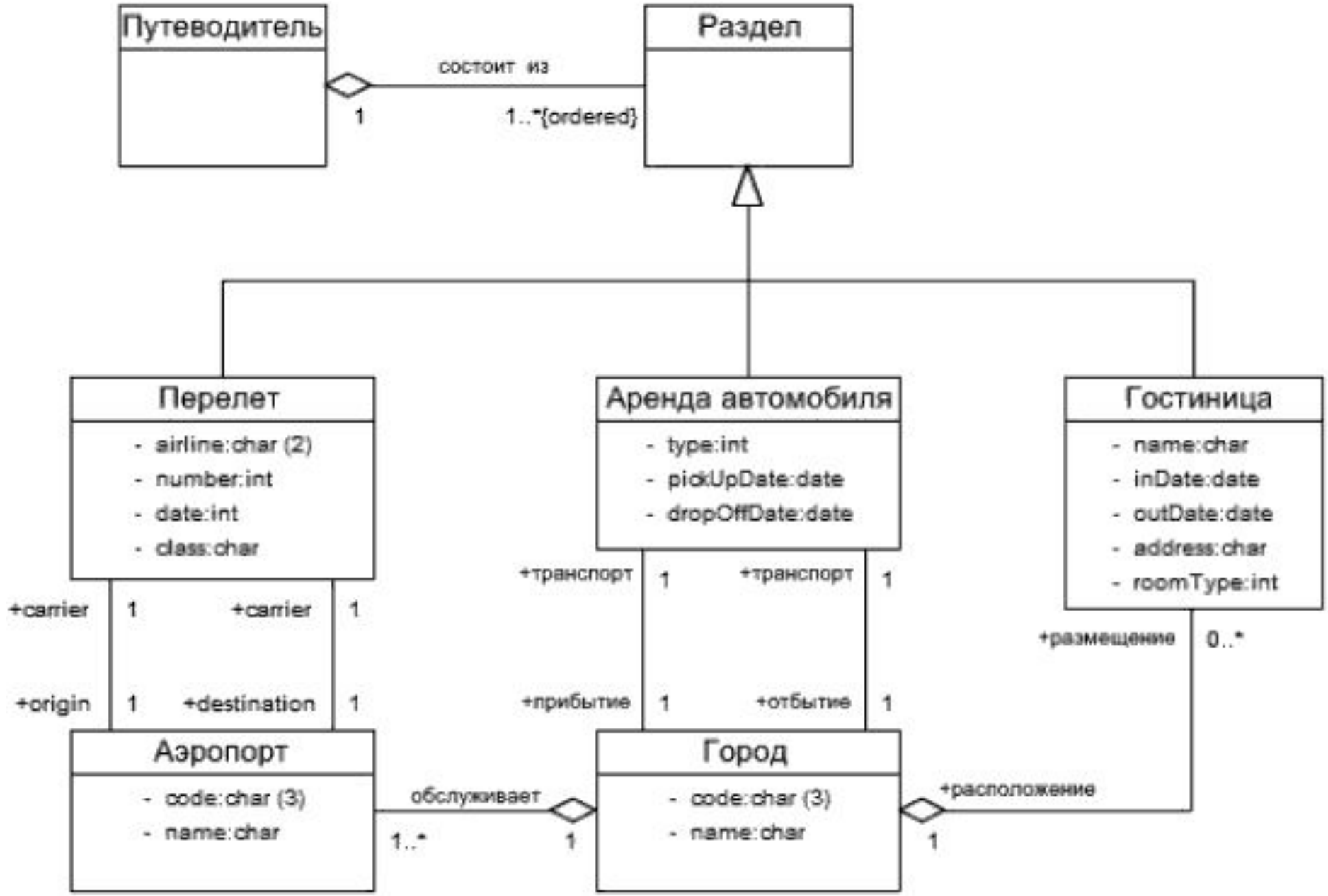


Диаграмма классов (class diagram)

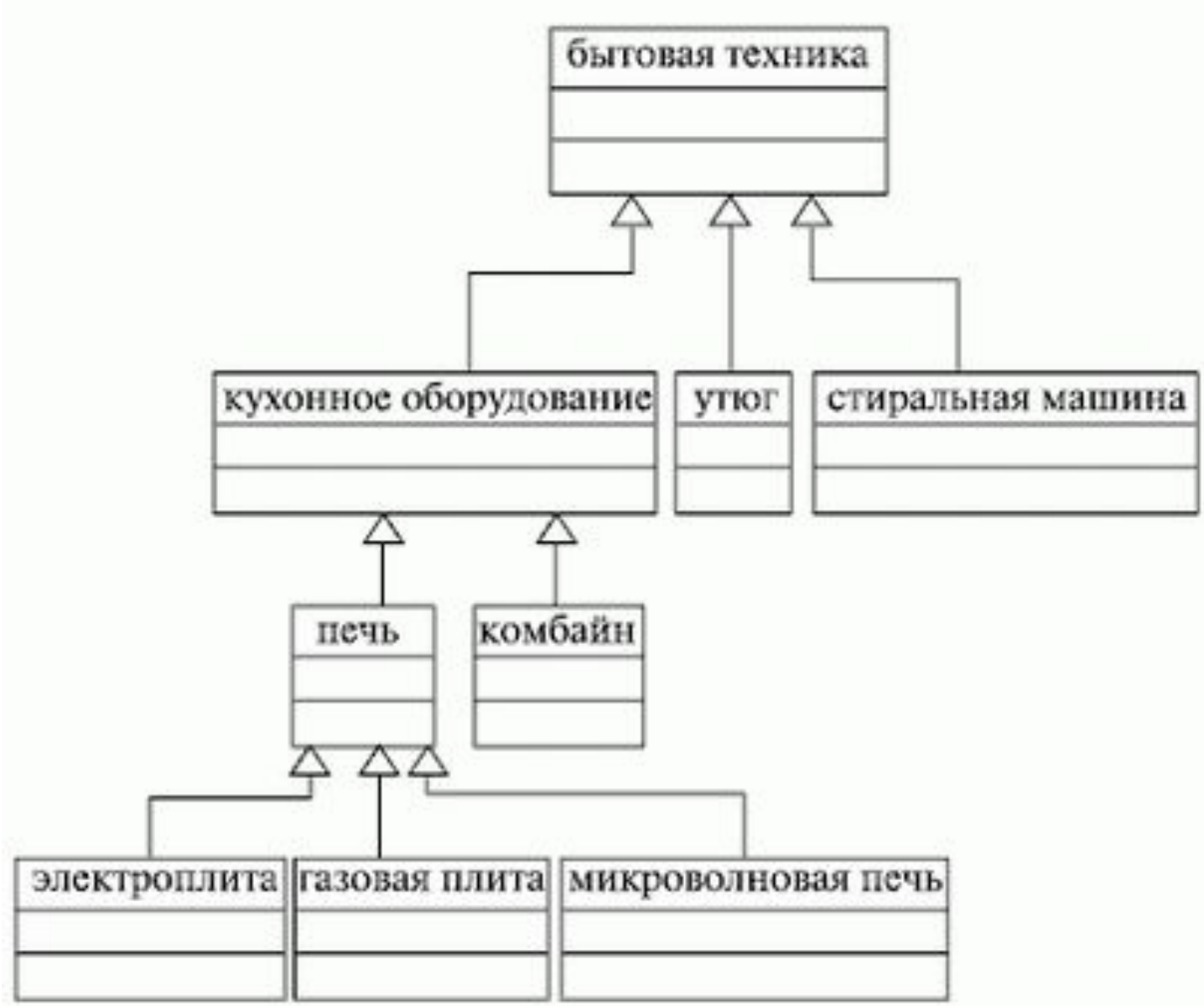


Диаграмма классов (class diagram)

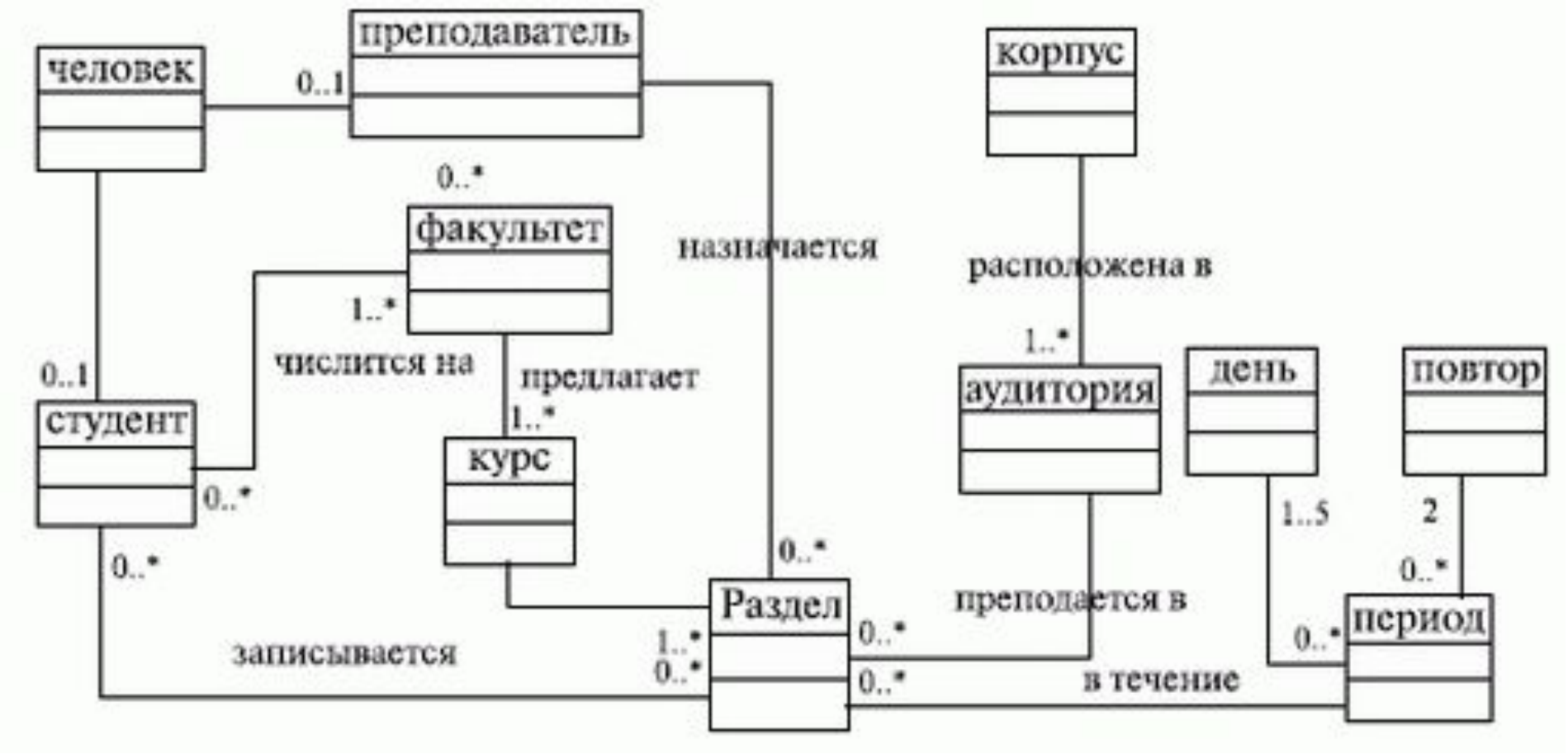


Диаграмма классов (class diagram)

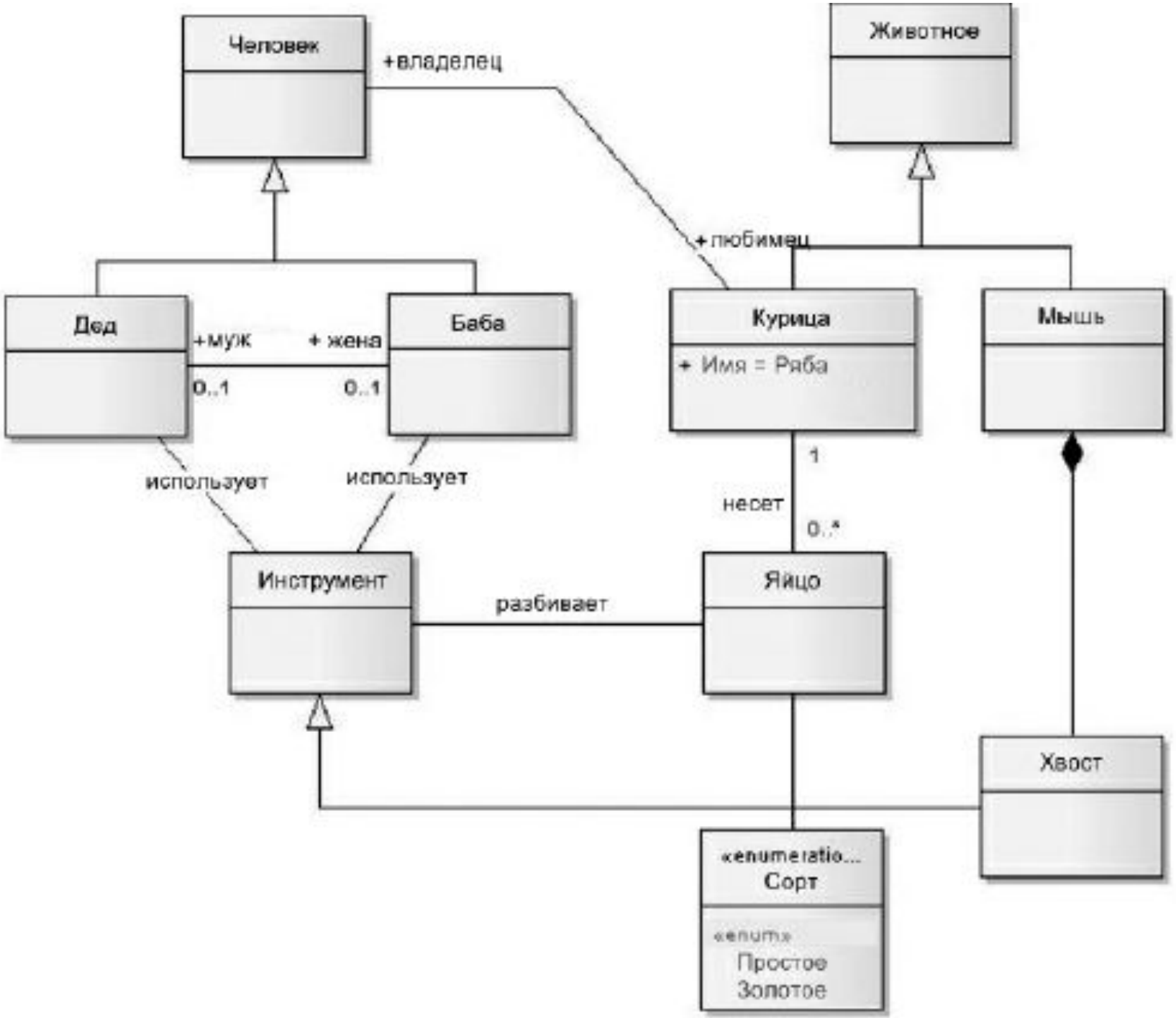
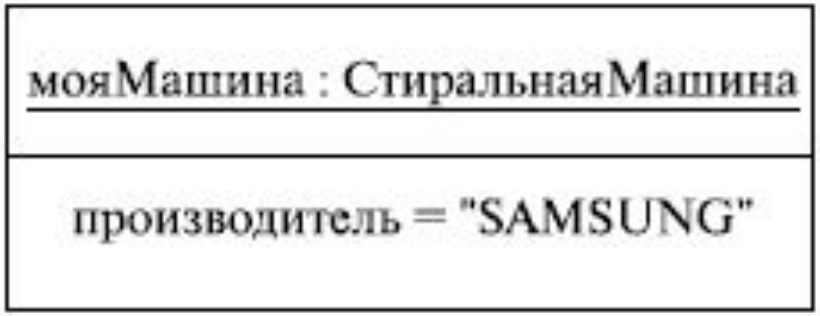

Человек
+владелец
Животное
+любимец
Дед
+муж
+ жена
Баба
0..1
0..1
Курица
+ Имя = Ряба
Мышь
1
несет
0..*
используует
используует
Инструмент
разбивает
Яйцо
Хвост
«enumeratio...
Сорт
«enum»
Простое
Золотое

Диаграмма объектов (object diagram)

- ***Объект (object)*** *- экземпляр класса.*
- **Объект (object)** -
- *конкретная материализация абстракции;*
- *сущность с хорошо определенными границами, в которой инкапсулированы состояние и поведение;*
- *экземпляр класса (вернее, классификатора - эктор, класс или интерфейс). Объект уникально идентифицируется значениями атрибутов, определяющими его состояние в данный момент времени.*

Объект, *как и класс, обозначается прямоугольником, но его имя подчеркивается.*

Диаграмма объектов (object diagram)



Для чего нужны диаграммы объектов? Они показывают множество объектов - экземпляров классов (изображенных на диаграмме классов) и отношений между ними в некоторый момент времени.

То есть *диаграмма объектов - это своего рода снимок состояния системы в определенный момент времени*, показывающий множество объектов, их состояния и отношения между ними в данный момент.

Диаграмма объектов (object diagram)

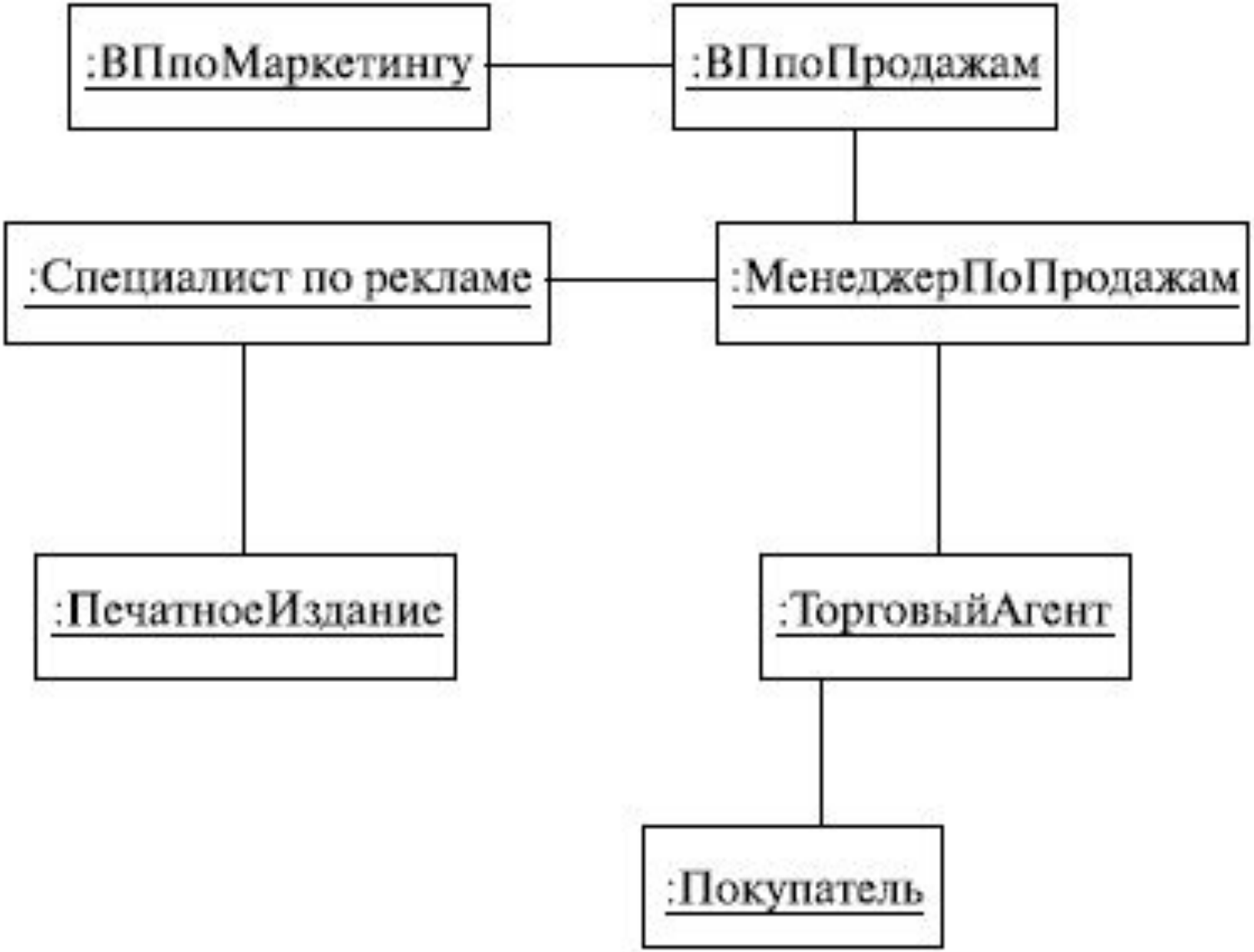


Диаграмма объектов (object diagram)

Диаграмма последовательностей (sequence diagram)

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

Диаграмма последовательностей (sequence diagram)

Диаграмма последовательностей (sequence diagram)

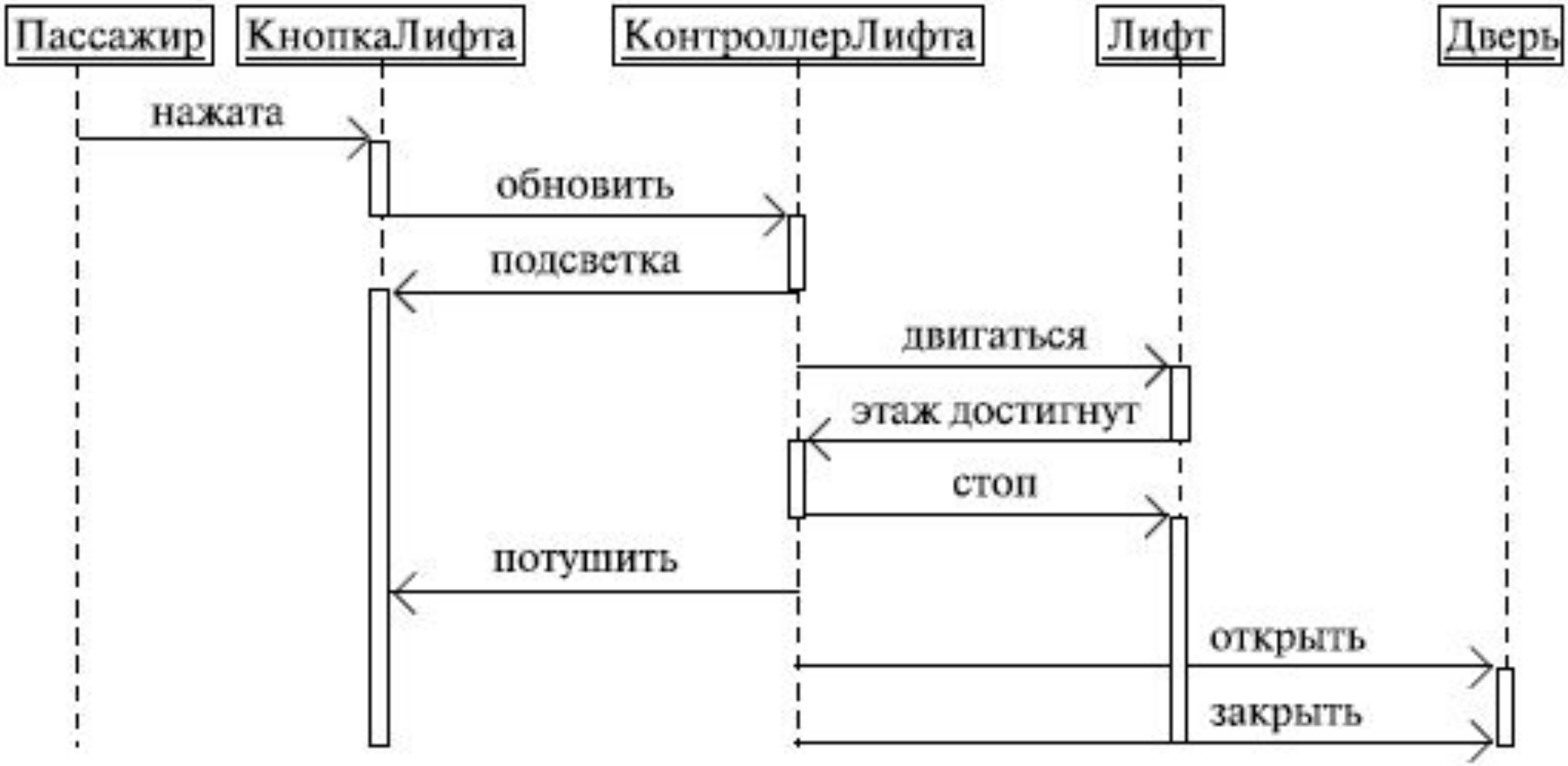


Диаграмма последовательностей (sequence diagram)

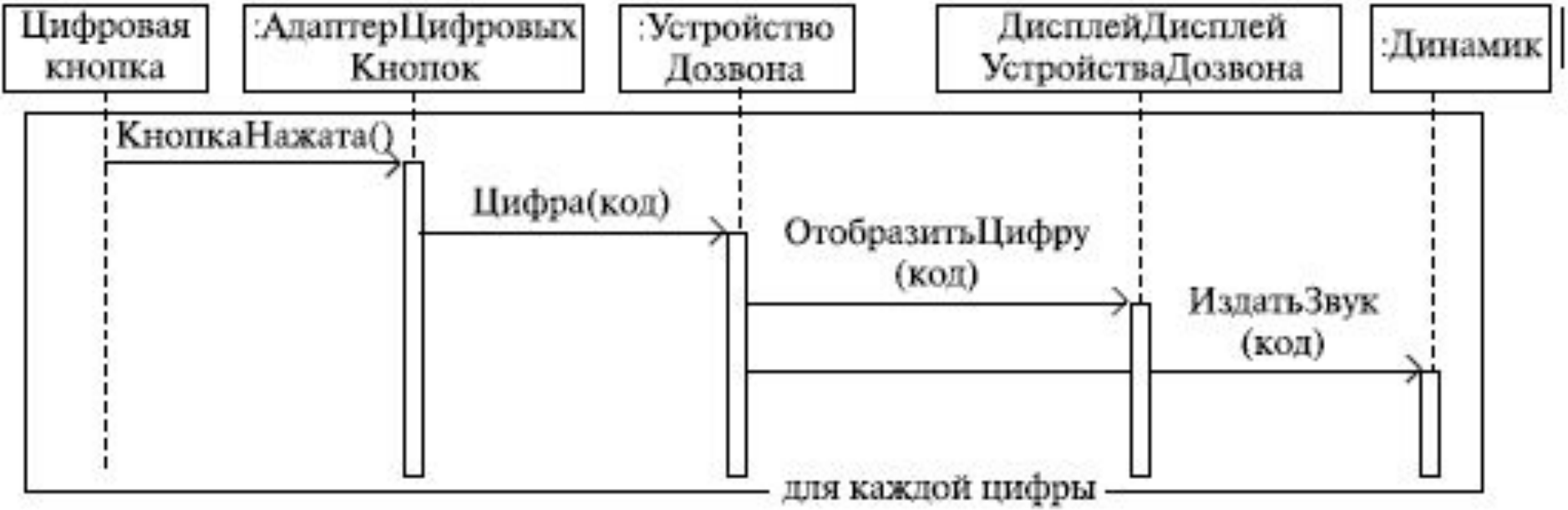


Диаграмма взаимодействия (кооперации, collaboration diagram)

Диаграмма взаимодействия (кооперации, collaboration diagram)

Диаграмма состояний (statechart diagram)

- ". **Диаграммы состояний** *применяются для того, чтобы объяснить, каким образом работают сложные объекты.*
- ***Состояние (state)*** *- ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.*

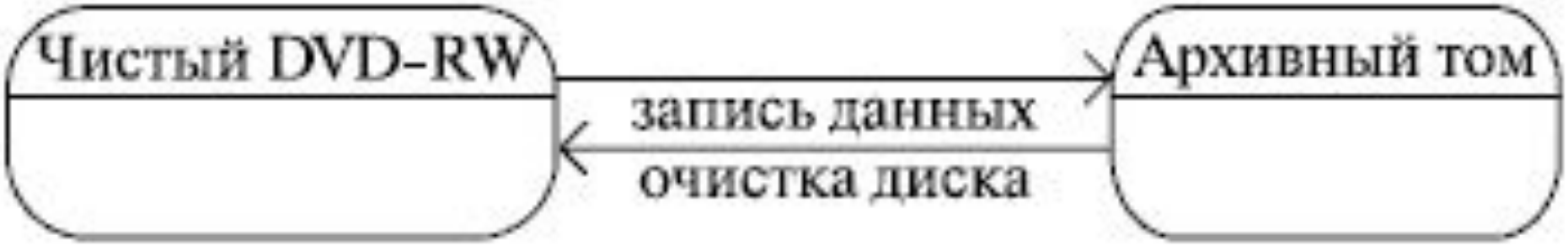


Диаграмма состояний (statechart diagram)

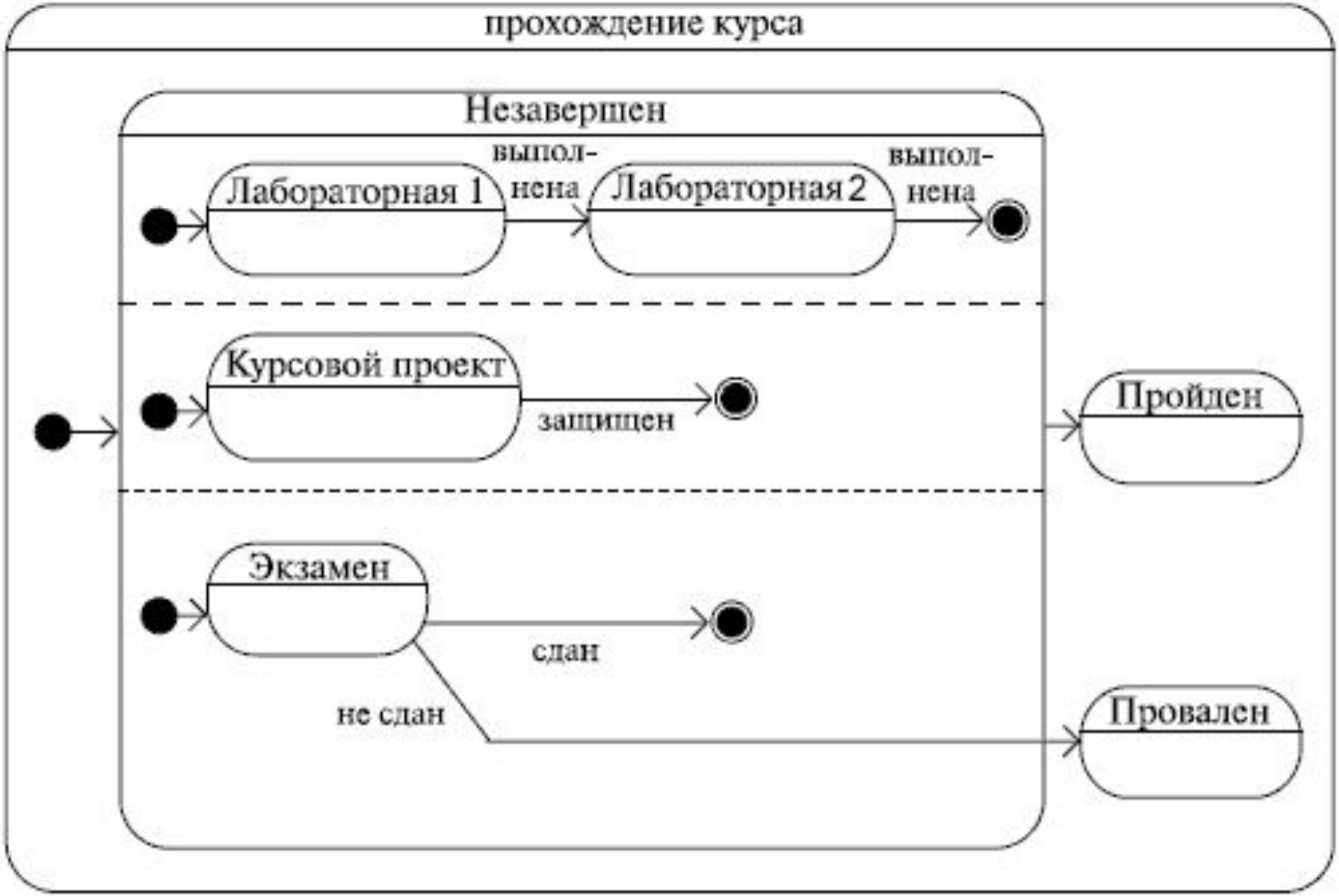


Диаграмма состояний (statechart diagram)

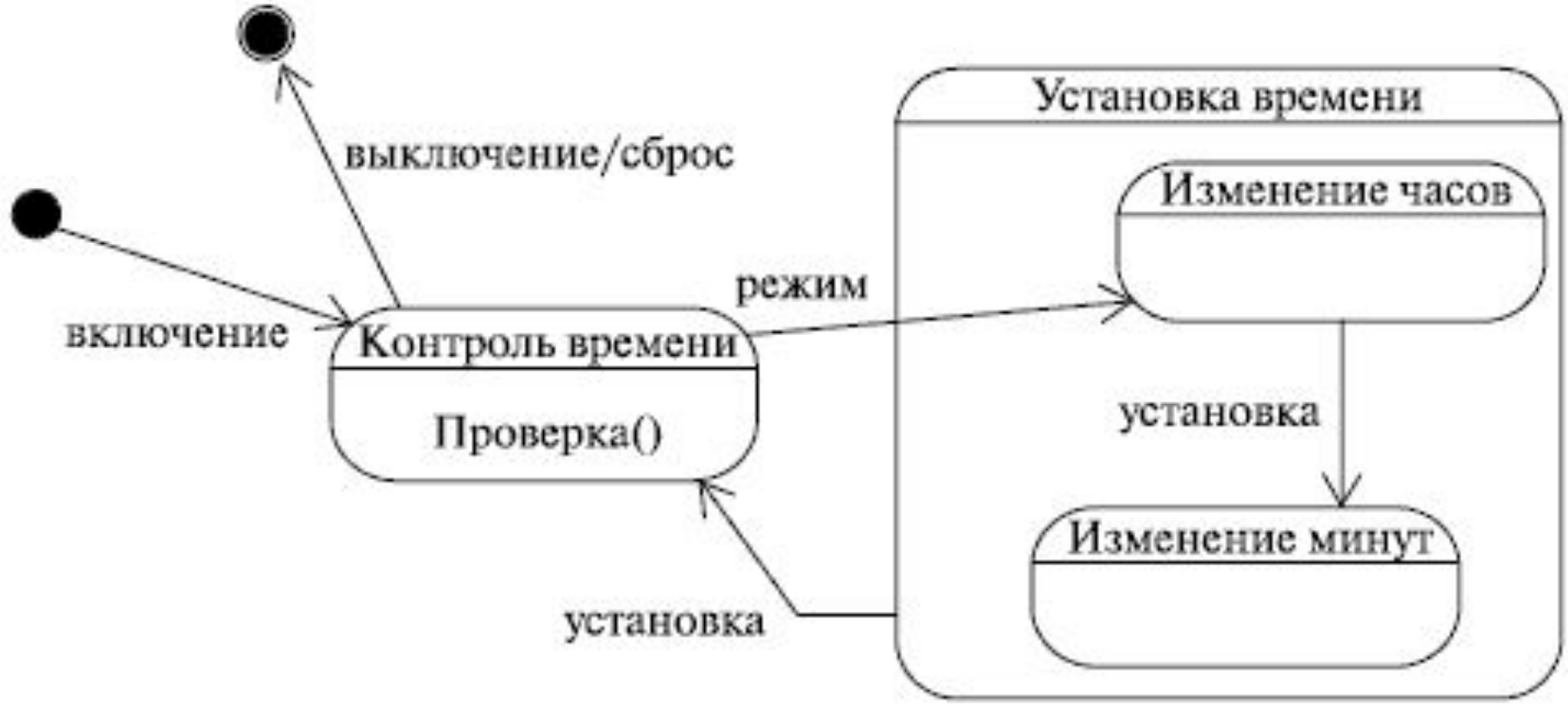


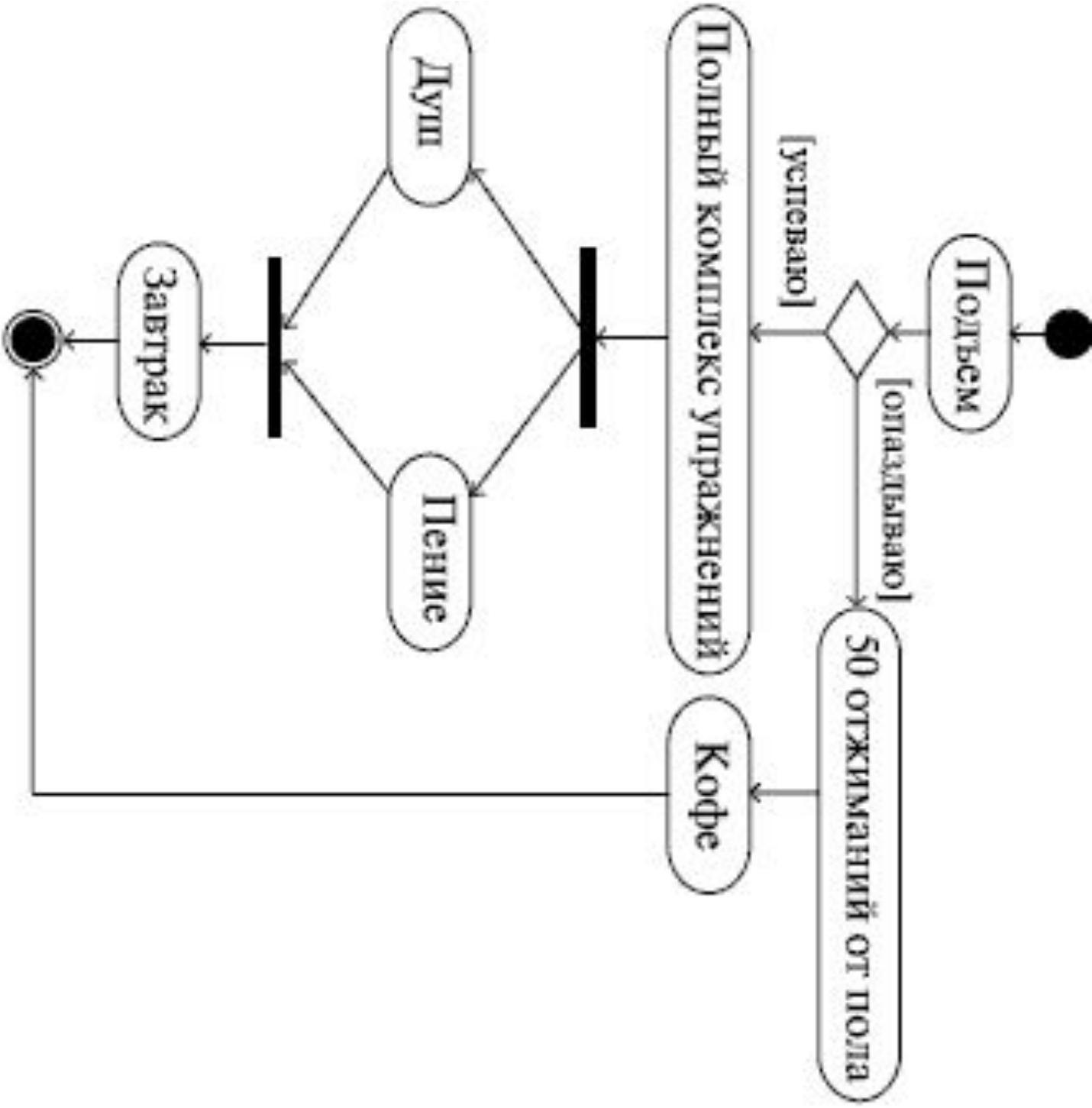
Диаграмма активности (деятельности, activity diagram)

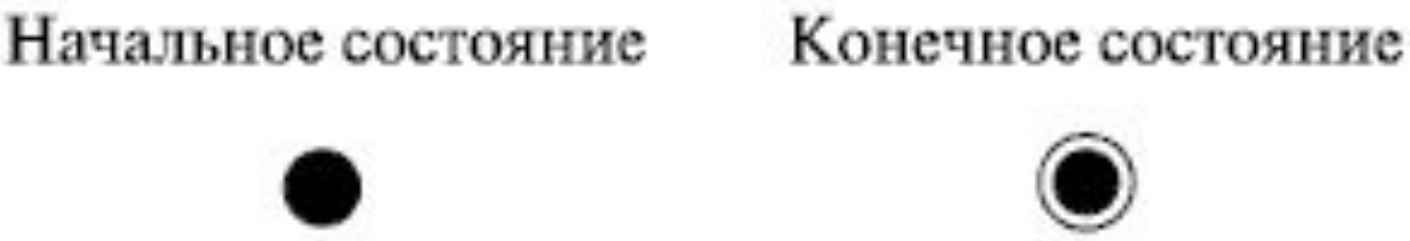
- Диаграммы деятельности удобно применять для визуализации алгоритмов, по которым работают операции классов.
- **Алгоритм** - последовательность определенных действий или элементарных операций, выполнение которых приводит к получению желаемого результата.

Диаграмма активностей

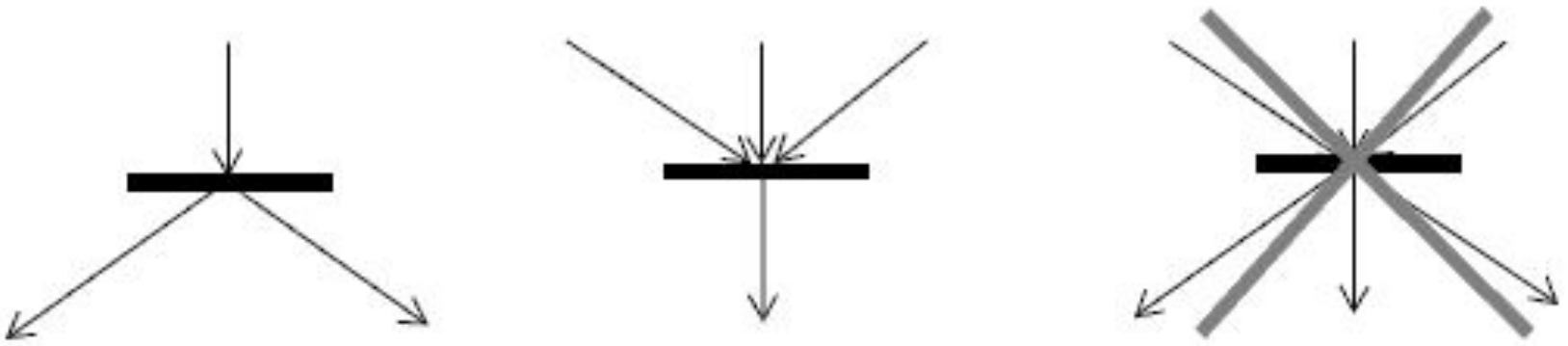
- Диаграммы активностей (Activity Diagrams) являются представлением алгоритмов неких действий (активностей), выполняющихся в системе.
- Существует пять представлений системы:
- Вид системы с точки зрения *прецедентов*.
- Вид с точки зрения *проектирования*.
- Вид с точки зрения *процессов*.
- Вид с точки зрения *развертывания*.
- Вид с точки зрения *реализации*.

Пример диаграммы деятельности



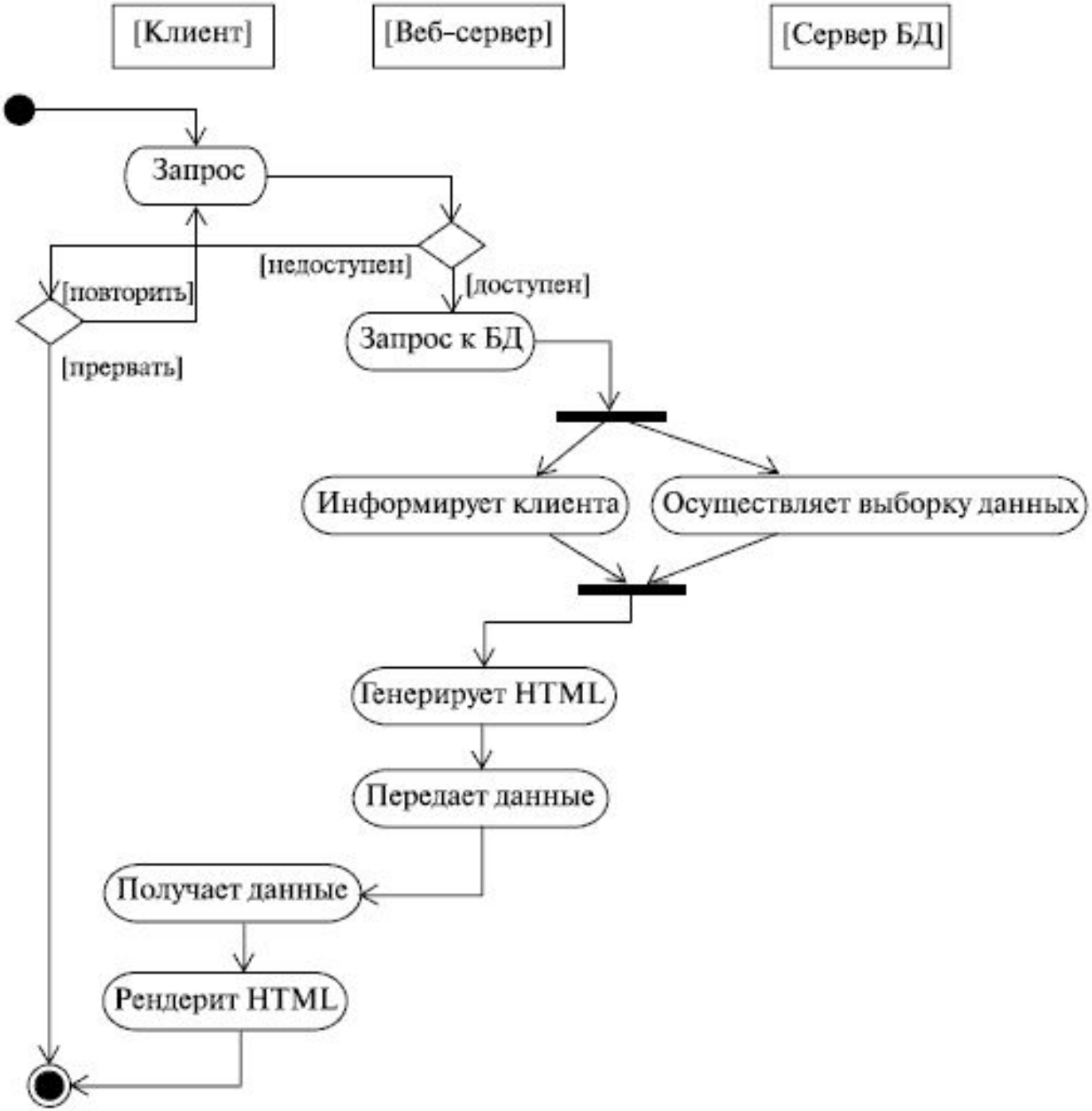


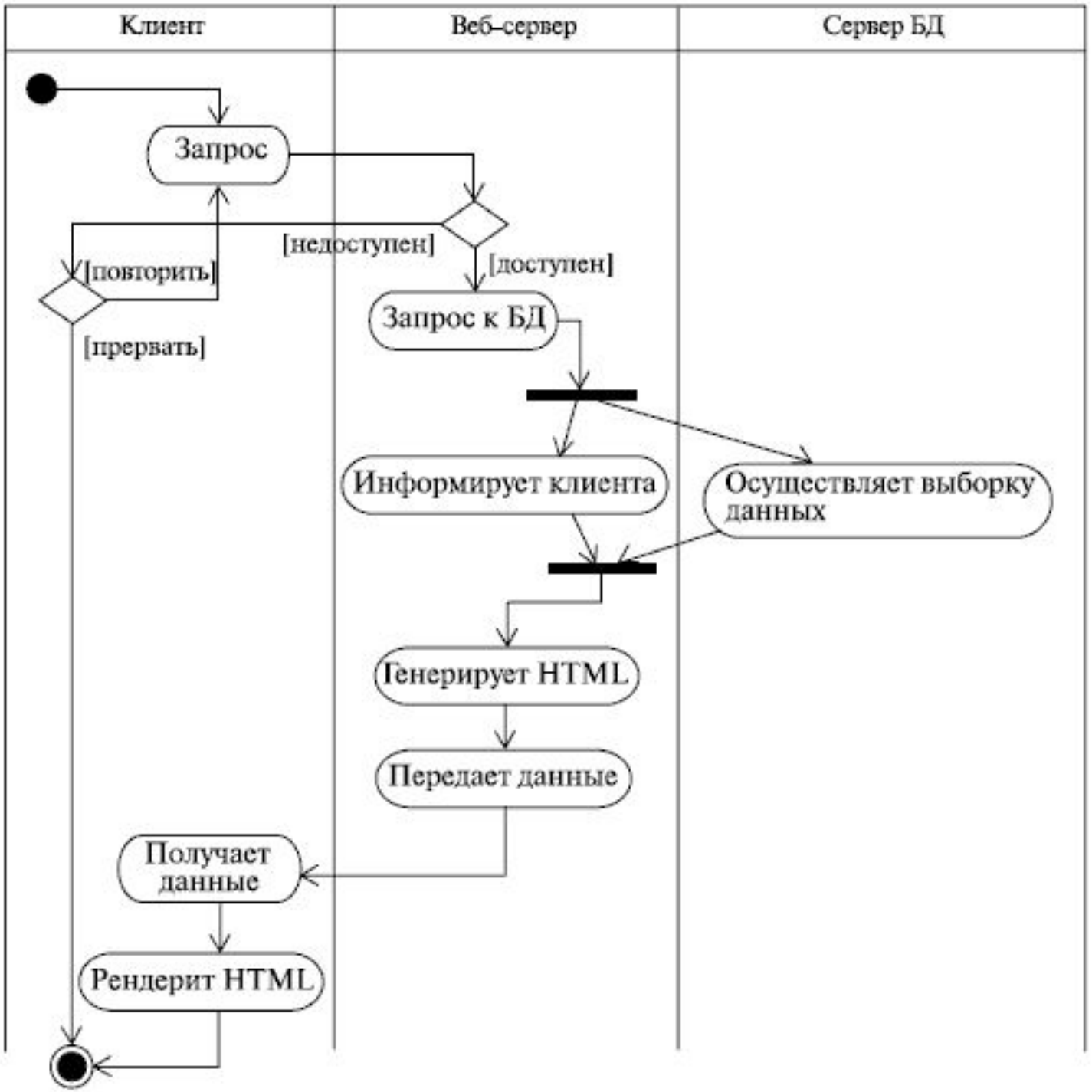
Обозначение начального и конечного состояний

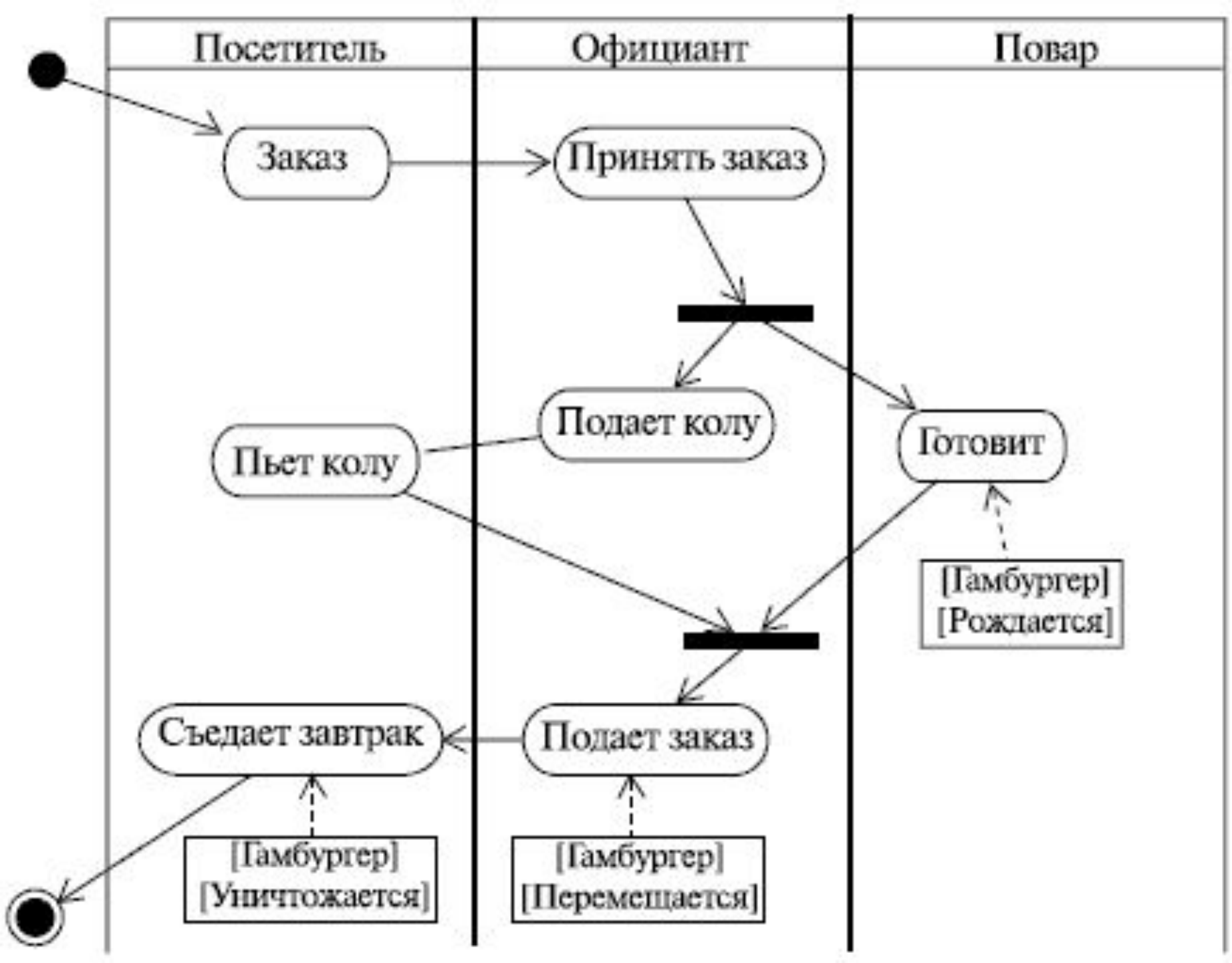


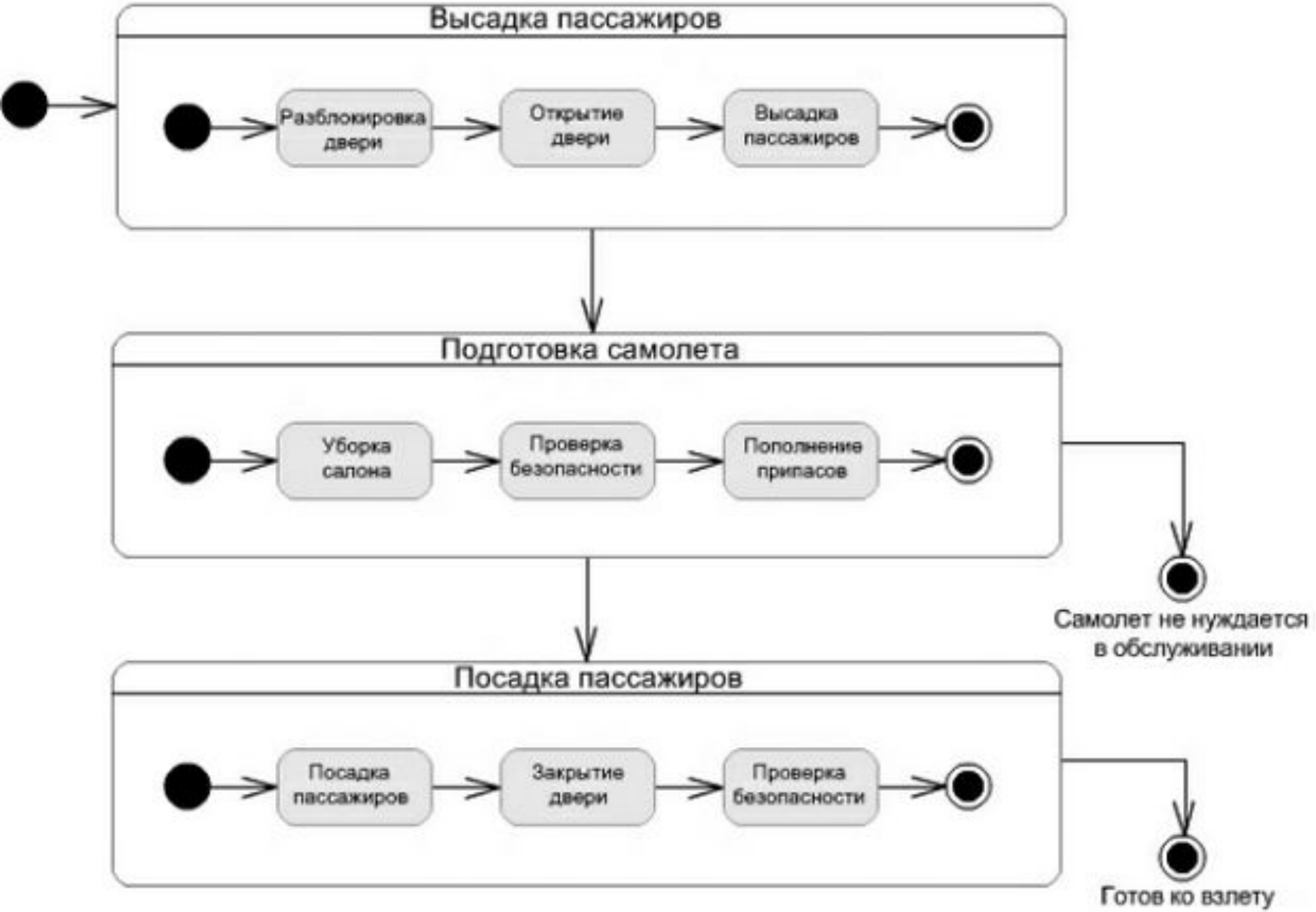
Обозначение начального и конечного состояний

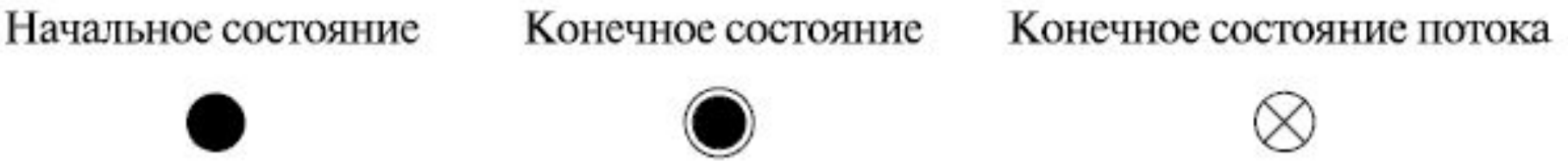
Пример диаграммы активностей




Клиент
Веб–сервер
Сервер БД
Запрос
[недоступен]
[доступен]
[повторить]
[прервать]
Запрос к БД
Информирует клиента
Осуществляет выборку данных
Генерирует HTML
Передает данные
Получает данные
Рендерит HTML


Посетитель
Официант
Повар
Заказ
Принять заказ
Подает колу
Пьет колу
Готовит
[Гамбургер]
[Рождается]
Съедает завтрак
Подает заказ
[Гамбургер]
[Уничтожается]
[Гамбургер]
[Перемещается]


Высадка пассажиров
Разблокировка двери
Открытие двери
Высадка пассажиров
Подготовка самолета
Уборка салона
Проверка безопасности
Пополнение припасов
Самолет не нуждается в обслуживании
Посадка пассажиров
Посадка пассажиров
Закрытие двери
Проверка безопасности
Готов ко взлету



Обозначение состояний потока

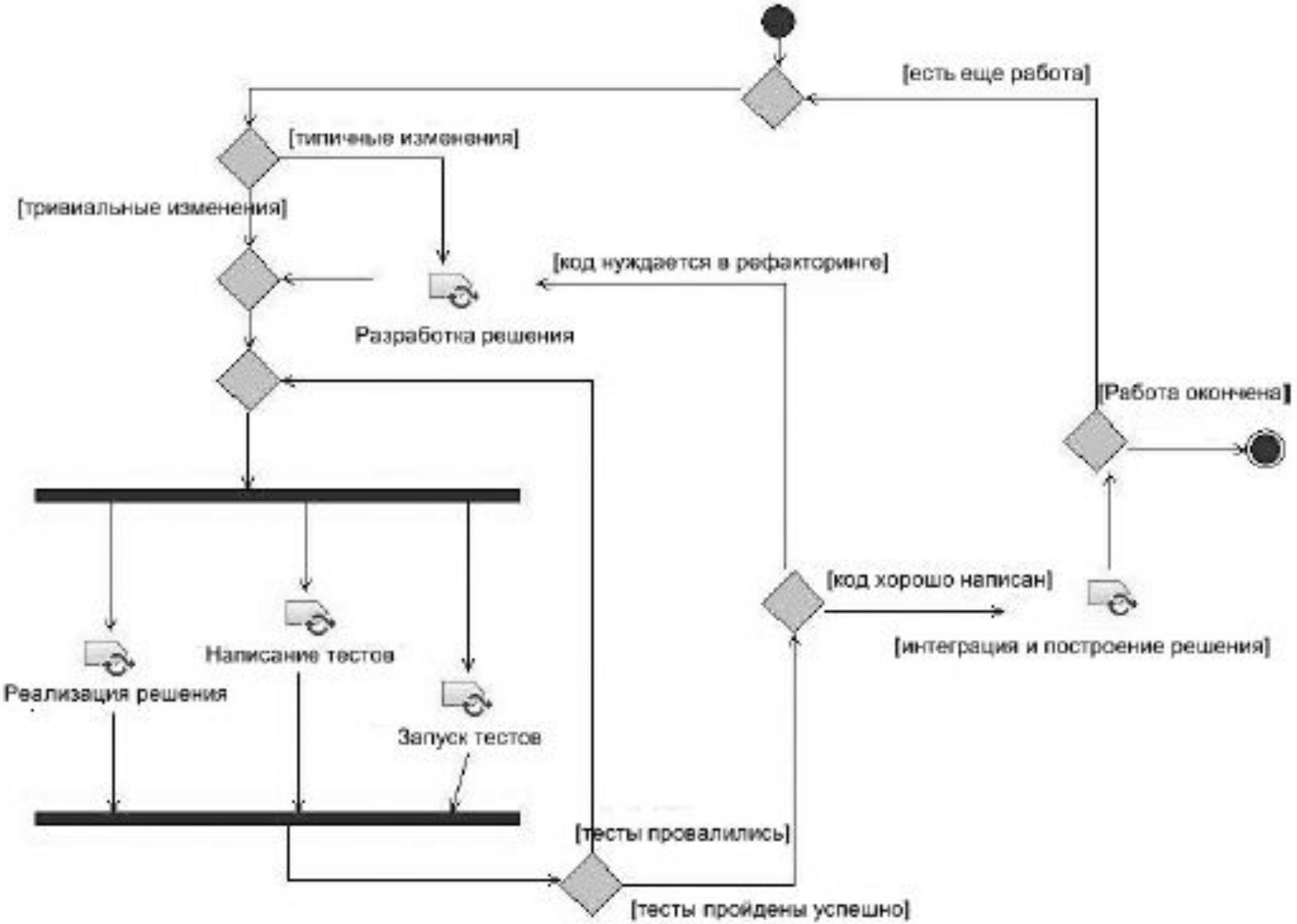
Примеры использования таких диаграмм

- На практике диаграммы деятельности применяются в основном двумя способами:
- **Для моделирования процессов**

В этом случае внимание фокусируется на деятельности с точки зрения экторов, которые работают с системой.

- **Для моделирования операций**

В этом случае диаграммы деятельности играют роль "продвинутых" блок-схем и применяются для подробного моделирования вычислений.


[есть еще работа]
[типичные изменения]
[тривиальные изменения]
[код нуждается в рефакторинге]
Разработка решения
[Работа окончена]
[код хорошо написан]
[интеграция и построение решения]
Написание тестов
Реализация решения
Запуск тестов
[тесты провалились]
[тесты пройдены успешно]

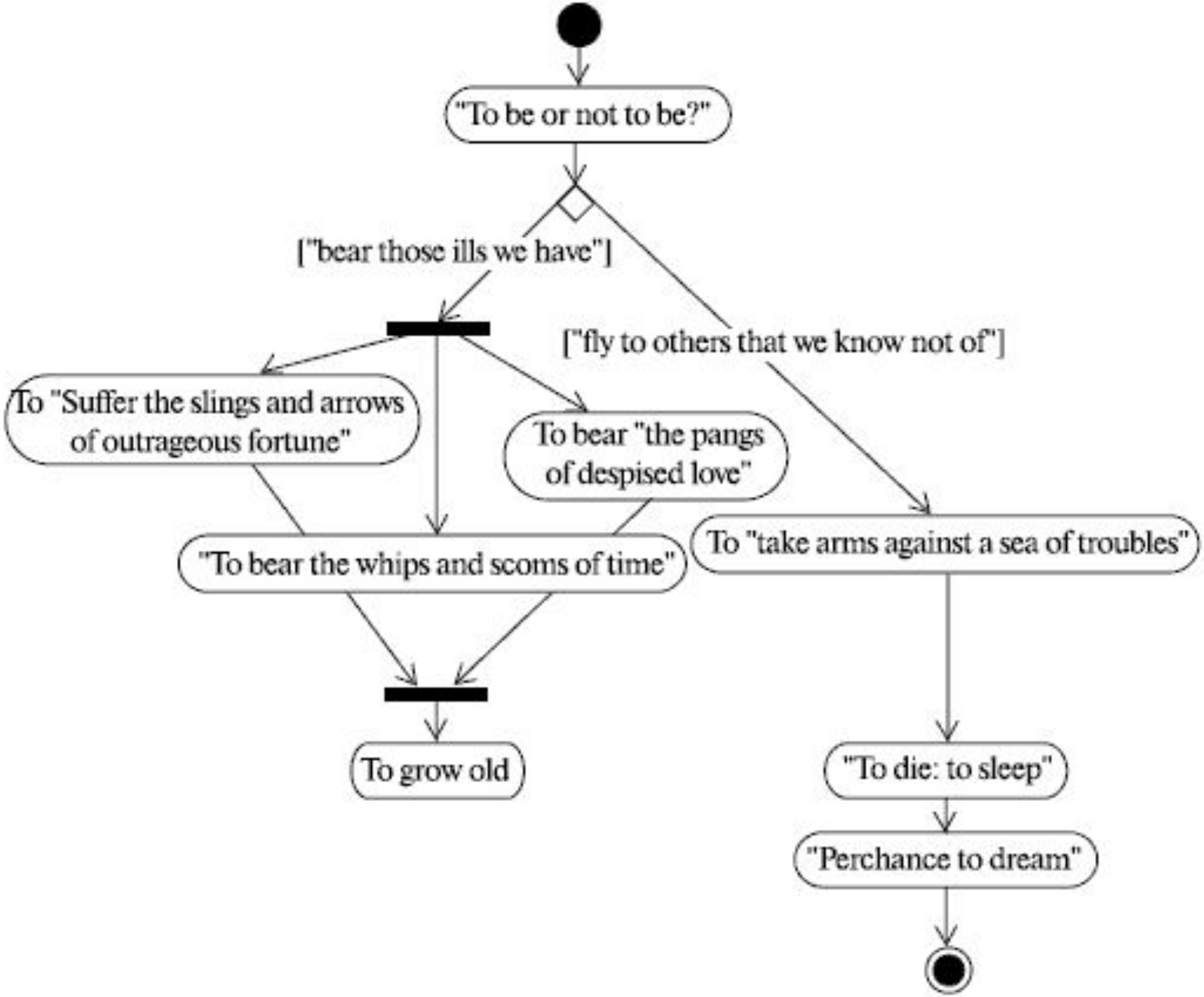
Примеры использования таких диаграмм

- **Рефакторинг** (*refactoring*) — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. **Цель рефакторинга** — сделать код программы легче для понимания; без этого рефакторинг нельзя считать успешным.


i: = 0; n: = 10
Выполнение операций
i: = i + 1
[i < n]
[i >= n]

Последовательность построения диаграммы

1. **Составление перечня деятельностей в системе**
2. **Принятие решения о необходимости построения диаграммы деятельностей**
3. **Определение зависимостей между деятельностями**
4. **Выделение параллельных потоков деятельностей**
5. **Определение условий переходов**
6. **Уточните сложные деятельности**


"To be or not to be?"
["bear those ills we have"]
["fly to others that we know not of"]
To "Suffer the slings and arrows of outrageous fortune"
To bear "the pangs of despised love"
"To bear the whips and scoms of time"
To "take arms against a sea of troubles"
To grow old
"To die: to sleep"
"Perchance to dream"

Выводы

- Диаграммой деятельности можно дополнить любой элемент модели, имеющий динамическое поведение.
- Диаграммы деятельности являются частным случаем диаграммы состояний.
- В отличие от блок-схем, диаграммы деятельности могут отображать одновременно выполняемые действия.
- На диаграммах активности можно использовать плавательные дорожки, распределяющие деятельности в соответствии с ролями (объектами), их выполняющими.
- Траектория объекта позволяет показать объекты, относящиеся к деятельности, и моменты переходов этих объектов из одного состояния в другое.
- Сложные деятельности можно дополнительно детализировать, разбив на действия и изобразив "диаграмму в диаграмме".
- Диаграммы деятельностей можно использовать для проектирования процессов (например, бизнес-процессов) или операций (вычислений). Во втором случае UML выступает в роли визуального языка программирования.

Диаграмма активности (деятельности, activity diagram)

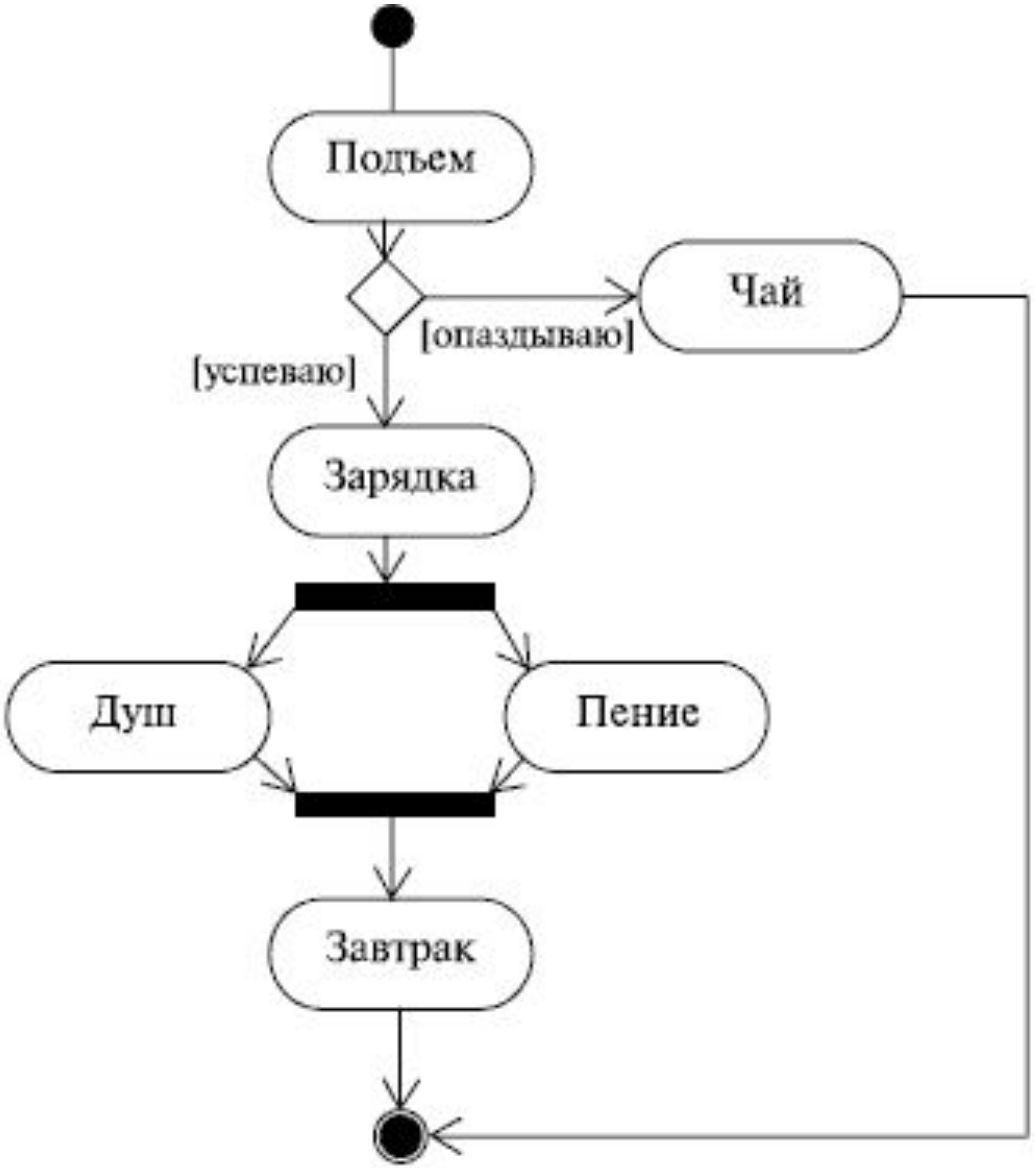


Диаграмма активности (деятельности, activity diagram)

Диаграмма активности (деятельности, activity diagram)

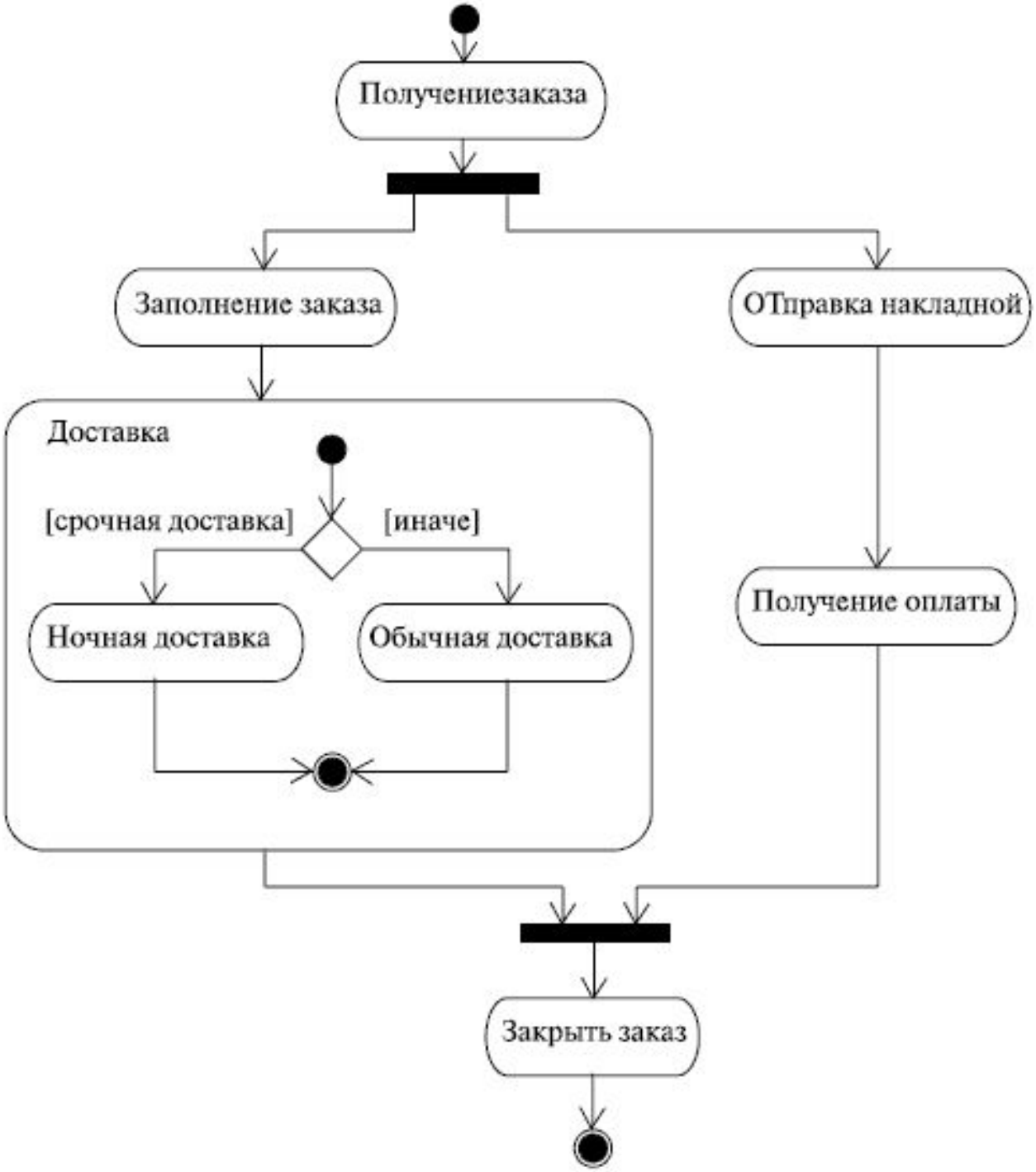
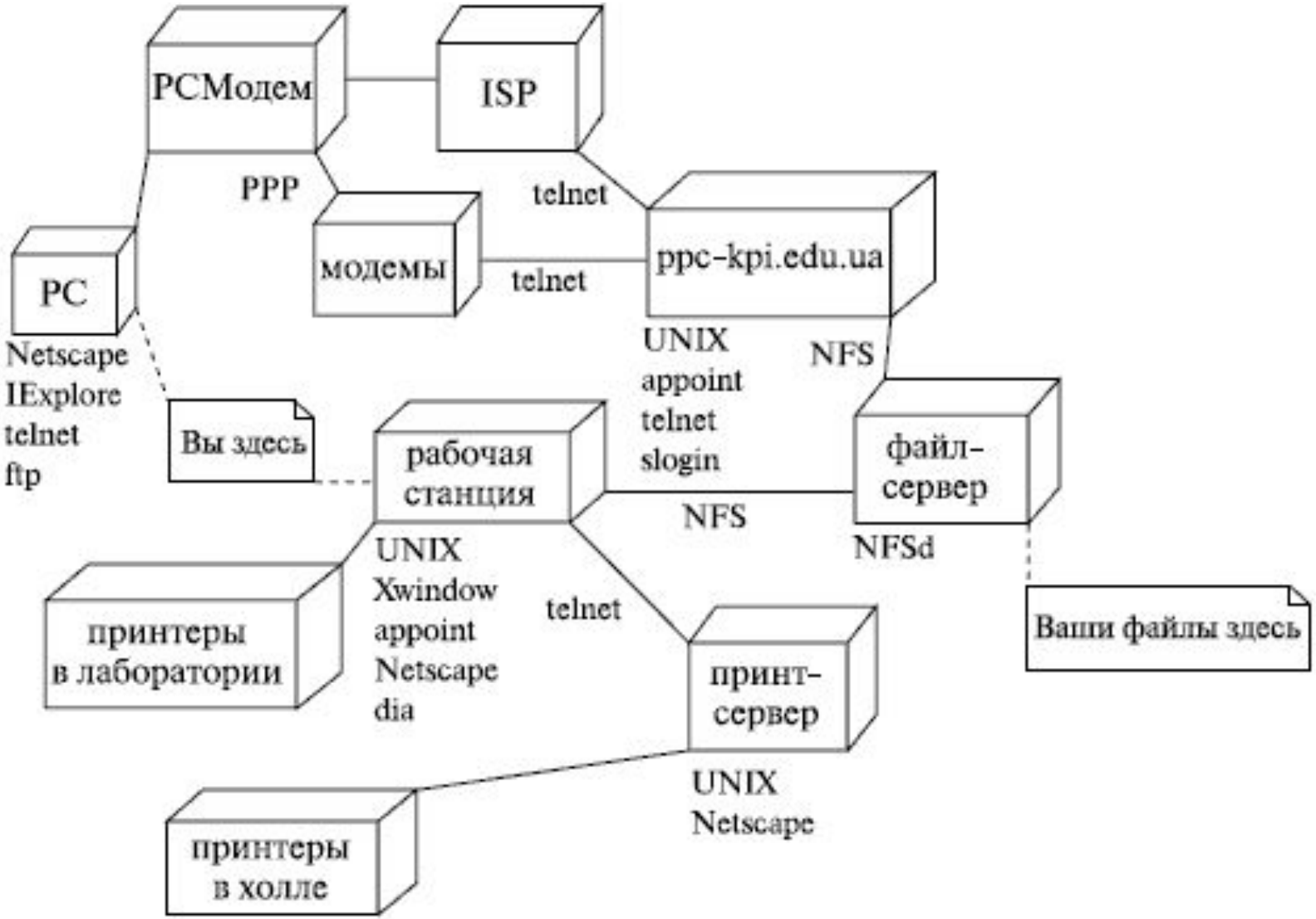


Диаграмма развертывания (deployment diagram)

- . **Диаграммы развертывания представляют** *графическое представление инфраструктуры, на которой будет развернуто приложение*.
- *Такие диаграммы есть смысл строить только для аппаратно-программных систем*, тогда как UML позволяет строить модели любых систем, не обязательно компьютерных.
- Применение:
- Во-первых, графическое представление ИТ-инфраструктуры может помочь *более рационально распределить компоненты системы по узлам сети*, от чего, как известно, зависит в том числе и производительность системы.
- Во-вторых, такая диаграмма может помочь *решить множество вспомогательных задач*, связанных, например, с обеспечением безопасности.

Диаграмма развертывания (deployment diagram)

Диаграмма развертывания (deployment diagram)



Последовательность построения

- Стройте *модели предметной области задачи в виде диаграммы классов*! Это хороший способ понять, как визуализировать множества взаимосвязанных абстракций. Таким же образом стройте модели статической части задач.
- Моделируйте *динамическую часть задачи с помощью простых диаграмм последовательностей и кооперации*. Хорошо начать с модели взаимодействия пользователя с системой - так вы сможете легко выделить наиболее важные прецеденты.
- Ппоследовательность построения диаграмм:
- диаграмма прецедентов,
- диаграмма классов,
- диаграмма объектов,
- диаграмма последовательностей,
- диаграмма кооперации,
- диаграмма состояний,
- диаграмма активности,
- диаграмма развертывания.