ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

обобщение

- **Технология разработки программного обеспечения (ТРПО)** – это совокупность процессов и методов создания программного продукта.
- **Технология разработки программного обеспечения** – это система инженерных принципов для создания экономичного ПО, которое надежно и эффективно работает в реальных компьютерах.
- **Технология разработки программного обеспечения** – это система инженерных принципов для создания экономичного ПО с заданными характеристиками качества.

ИНСТРУМЕНТАРИЙ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Инструментарий технологии программирования – совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов.

ЖИЗНЕННЫЙ ЦИКЛ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Жизненный цикл ПО (ЖЦ ПО) - **это** непрерывный и упорядоченный набор видов деятельности, осуществляемый и управляемый в рамках каждого проекта по разработке и эксплуатации ПО, начинающийся с момента появления идеи (замысла) создания некоторого программного обеспечения и принятия решения о крайне важности его создания и заканчивающийся в момент его полного изъятия из эксплуатации по причинам: а) морального старения; б) потери крайне важности решения соответствующих задач.

Простейшее представление жизненного цикла, включает стадии:

-Анализ
-Проектирование
-Реализация
-Тестирование и отладка
-Внедрение, эксплуатация и сопровождение.
- Завершение эксплуатации

ЗАДАЧИ ЭТАПОВ

- ***Этап анализа*** концентрируется на системных требованиях.

Требования определяются и специфицируются (описываются). Осуществляется выработка и интеграция функциональных моделей и моделей данных для системы. Вместе с тем, фиксируются нефункциональные и другие системные требования.

- **Этап *проектирования*** разделяется на два базовых подэтапа: архитектурное и детализированное проектирование.

В частности, проводится уточнение конструкции программы, пользовательского интерфейса и структур данных. Поднимаются и фиксируются вопросы проектирования, которые влияют на понятность, приспособленность к сопровождению и масштабируемость системы.

- ***Этап реализации*** включает написание программы.

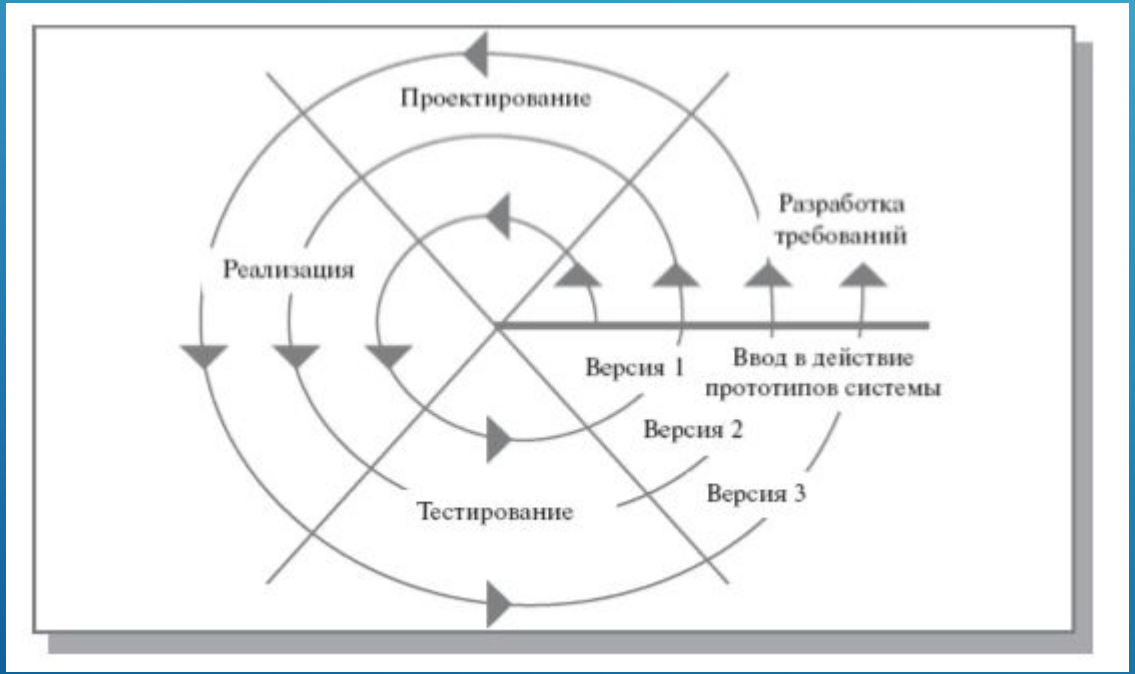
МОДЕЛИ ЖИЗНЕННОГО ЦИКЛА

КАСКАДНАЯ МОДЕЛЬ

- Его основной характеристикой является *разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем*. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

СПИРАЛЬНАЯ МОДЕЛЬ

- модель ЖЦ делающая упор на начальные этапы ЖЦ: анализ и проектирование. На этих этапах реализуемость технических решений проверяется путем создания прототипов. Каждый виток спирали соответствует созданию фрагмента или версии ПО, на нем уточняются цели и характеристики проекта, определяется его качество и планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.



ИНКРЕМЕНТНАЯ (ПОЭТАПНАЯ МОДЕЛЬ С ПРОМЕЖУТОЧНЫМ КОНТРОЛЕМ)

- **Инкрементная модель** (англ. *increment* — увеличение, приращение) подразумевает разработку программного обеспечения с линейной последовательностью стадий, но в несколько инкрементов (версий), т.е. с запланированным улучшением продукта за все время пока Жизненный цикл разработки ПО не подойдет к окончанию

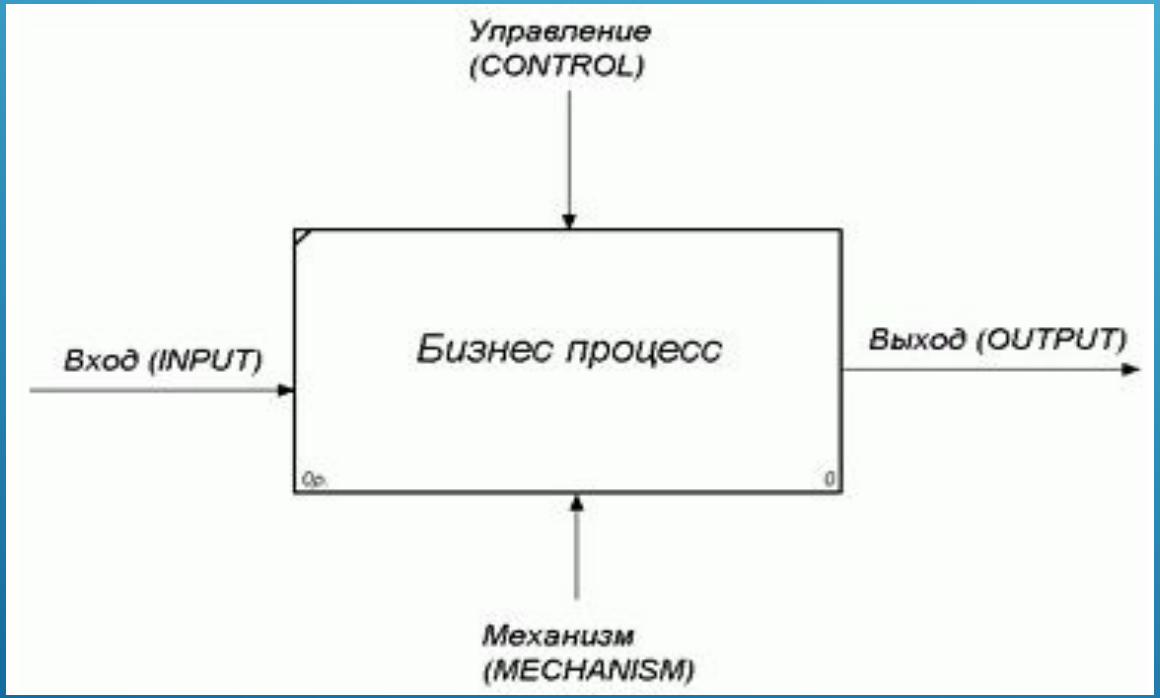
ТРЕБОВАНИЯ

ТИПЫ ТРЕБОВАНИЙ

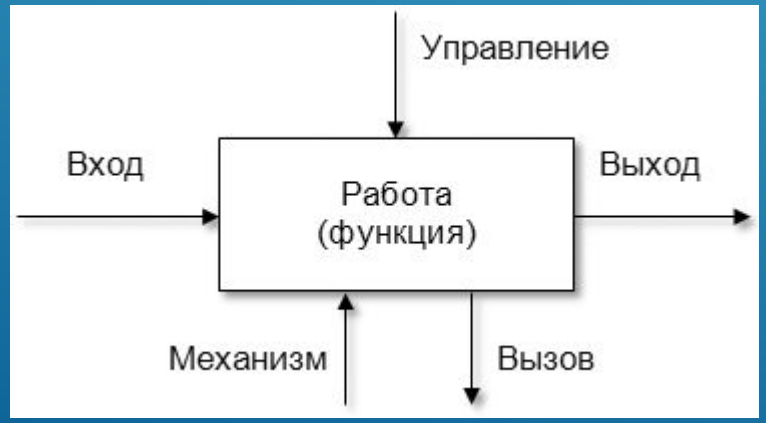
- *Функциональные требования*
- *Нефункциональные требования*
- *Бизнес-требования*
- *Требования пользователей*
- *Системные требования*
- *Бизнес-правила*

IDEF0

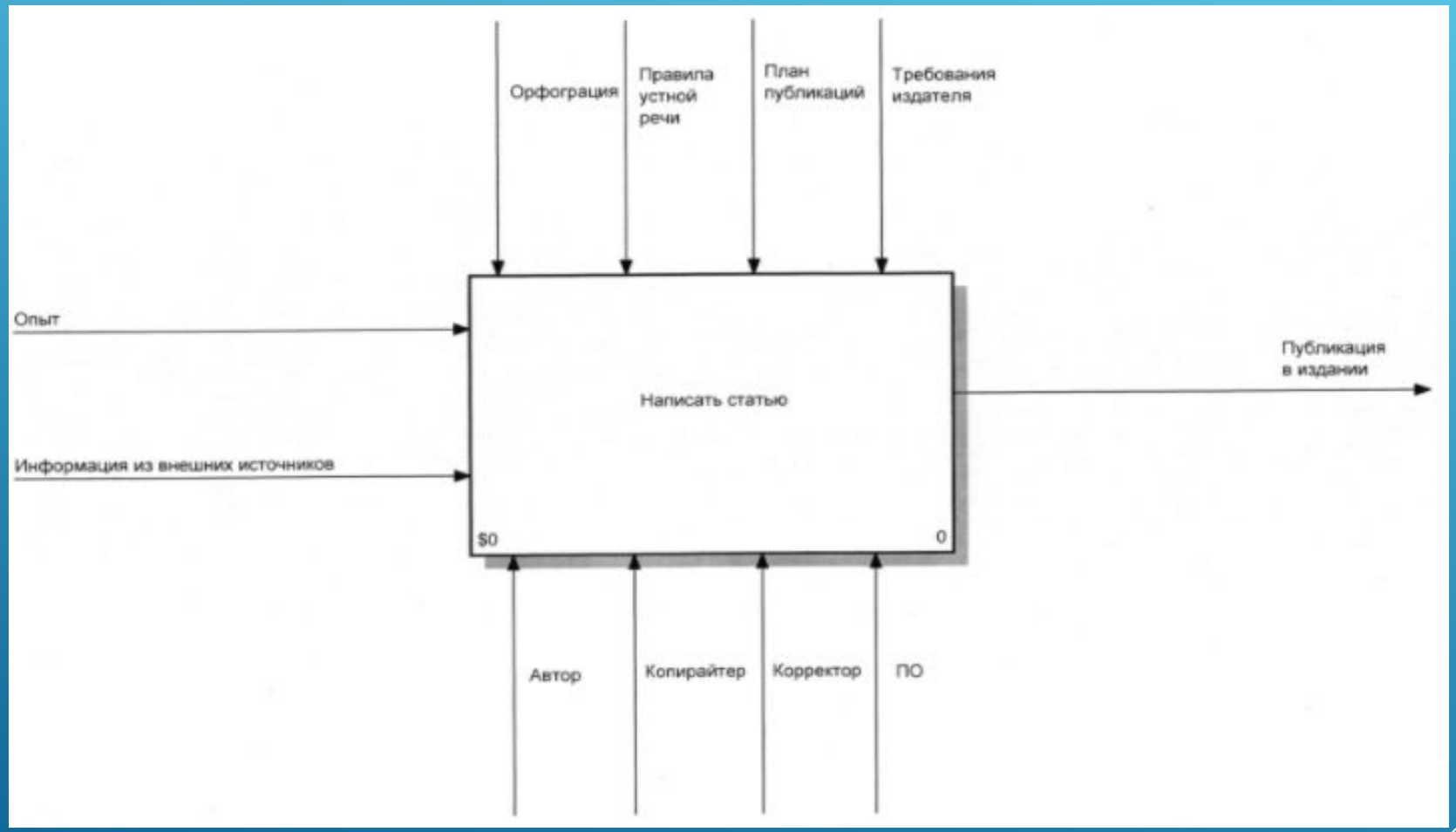
IDEF0 — методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временна́я последовательность (поток работ).

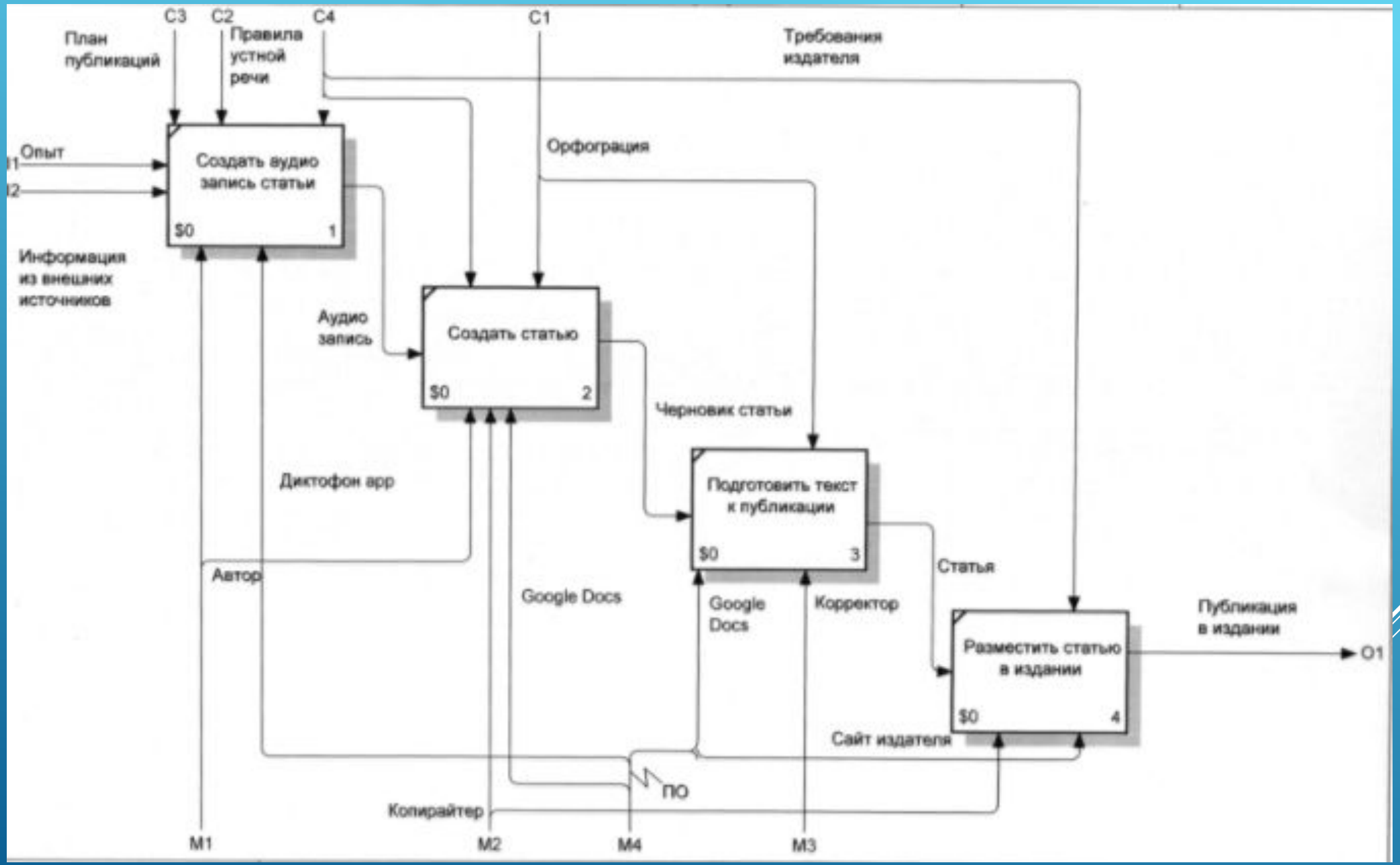


- ***вход*** (англ. input) – материал или информация, которые используются и преобразуются работой для получения результата (выхода). Вход отвечает на вопрос «Что подлежит обработке?». В качестве входа может быть как материальный объект (сырье, деталь, экзаменационный билет), так и не имеющий четких физических контуров (запрос к БД, вопрос преподавателя). Допускается, что работа может не иметь ни одной стрелки входа. Стрелки входа всегда рисуются входящими в левую грань работы;
- ***управление*** (англ. control) – управляющие, регламентирующие и нормативные данные, которыми руководствуется работа. Управление отвечает на вопрос «В соответствии с чем выполняется работа?». Управление влияет на работу, но не преобразуется ей, т.е. выступает в качестве ограничения. В качестве управления могут быть правила, стандарты, нормативы, расценки, устные указания. Стрелки управления рисуются входящими в верхнюю грань работы. Если при построении диаграммы возникает вопрос, как правильно нарисовать стрелку сверху или слева, то рекомендуется ее рисовать как вход (стрелка слева);
- ***выход*** (англ. output) – материал или информация, которые представляют результат выполнения работы. Выход отвечает на вопрос «Что является результатом работы?». В качестве выхода может быть как материальный объект (деталь, автомобиль, платежные документы, ведомость), так и нематериальный (выборка данных из БД, ответ на вопрос, устное указание). Стрелки выхода рисуются исходящими из правой грани работы;
- ***механизм*** (англ. mechanism) – ресурсы, которые выполняют работу. Механизм отвечает на вопрос «Кто выполняет работу или посредством чего?». В качестве механизма могут быть персонал предприятия, студент, станок, оборудование, программа. Стрелки механизма рисуются входящими в нижнюю грань работы;



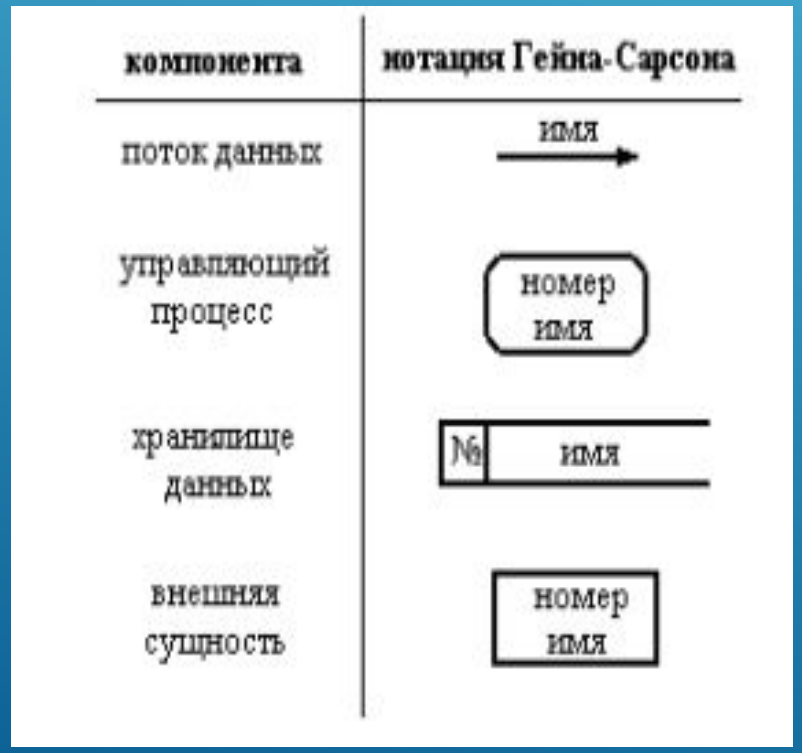
ПРИМЕР

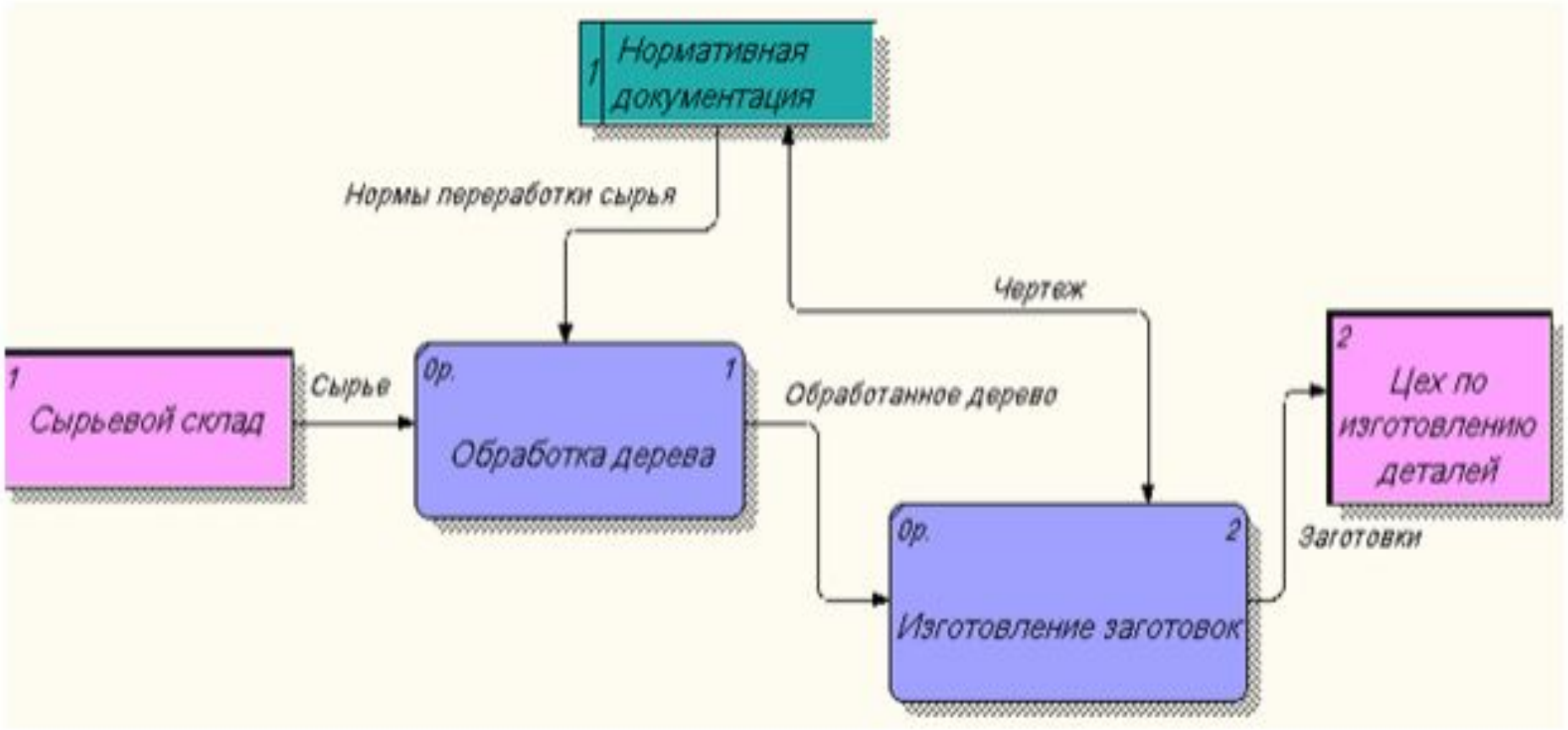

Орфография
Правила устной речи
План публикаций
Требования издателя
Опыт
Информация из внешних источников
Написать статью
A0
0
Публикация в издании
Автор
Копирайтер
Корректор
ПО


C3
C2
C4
C1
План публикаций
Правила устной речи
Требования издателя
Опыт
I1
I2
Создать аудио запись статьи
$0
1
Орфография
Информация из внешних источников
Аудио запись
Создать статью
$0
2
Черновик статьи
Диктофон app
Подготовить текст к публикации
$0
3
Статья
Автор
Google Docs
Google Docs
Корректор
Публикация в издании
O1
Разместить статью в издании
$0
4
Сайт издателя
ПО
Копирайтер
M1
M2
M4
M3

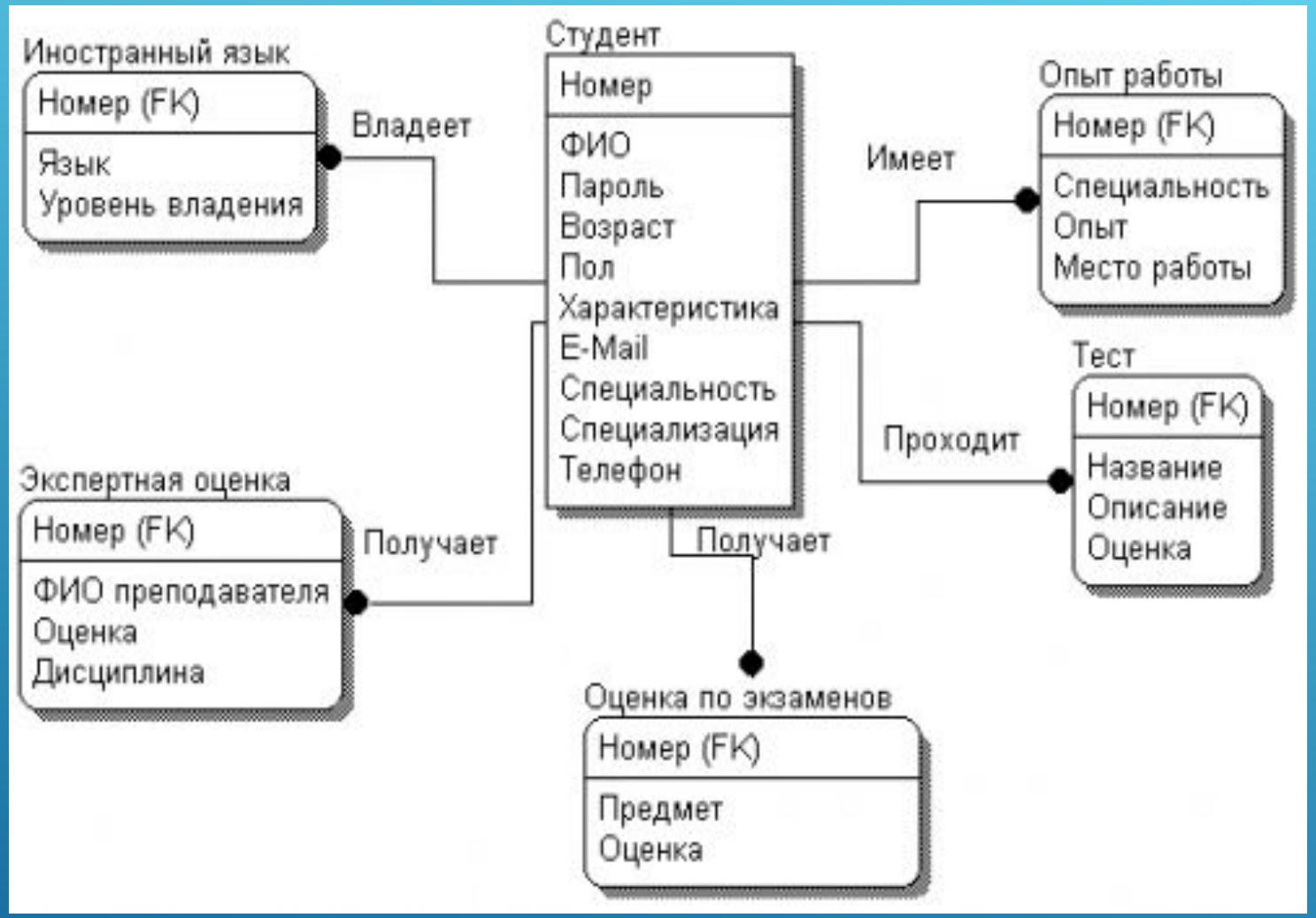
DFD - ДИАГРАММЫ ПОТОКОВ ДАННЫХ

- Диаграммы потоков данных используются для описания документооборота и обработки информации и представляют модельную систему как сеть связанных между собой работ. Диаграммы потоков данных (DFD) показывают внешние источники и приемники данных, потоки данных и хранилища (накопители) данных, к которым осуществляется доступ.




1
Нормативная документация
Нормы переработки сырья
Чертеж
1
Сырьевой склад
Сырье
0р.
1
Обработка дерева
Обработанное дерево
0р.
2
Изготовление заготовок
2
Цех по изготовлению деталей
Заготовки

ERD – ДИАГРАММЫ «СУЩНОСТЬ-СВЯЗЬ»

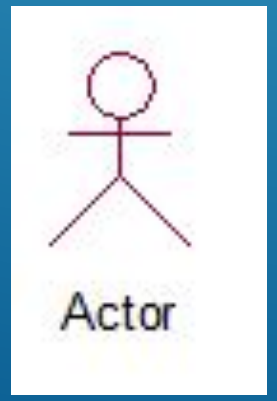

Иностранный язык
Номер (FK)
Язык
Уровень владения
Владеет
Студент
Номер
ФИО
Пароль
Возраст
Пол
Характеристика
E-Mail
Специальность
Специализация
Телефон
Имеет
Опыт работы
Номер (FK)
Специальность
Опыт
Место работы
Тест
Номер (FK)
Название
Описание
Оценка
Проходит
Экспертная оценка
Номер (FK)
ФИО преподавателя
Оценка
Дисциплина
Получает
Получает
Оценка по экзаменов
Номер (FK)
Предмет
Оценка

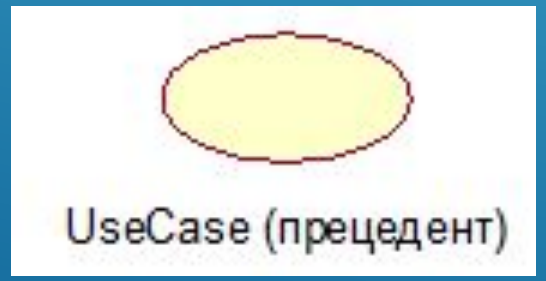
УНИФИЦИРОВАННЫЙ ЯЗЫК МОДЕЛИРОВАНИЯ **UML** (UNIFIED MODELING LANGUAGE)

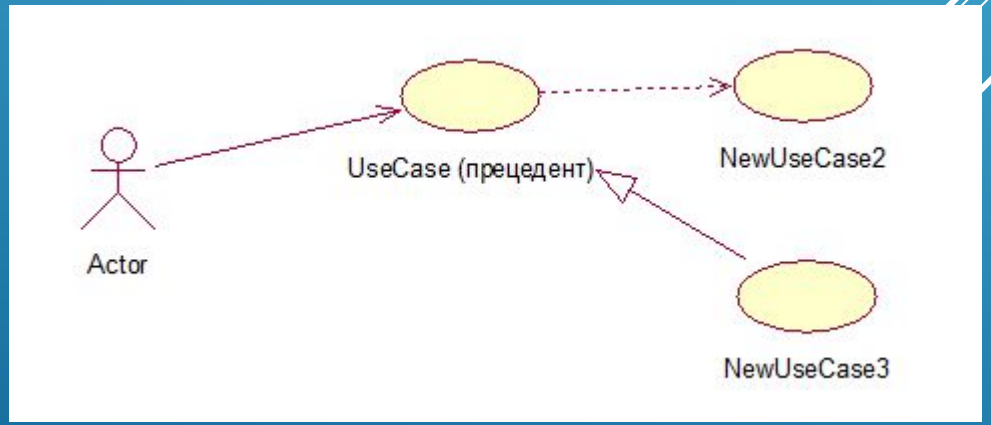
ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

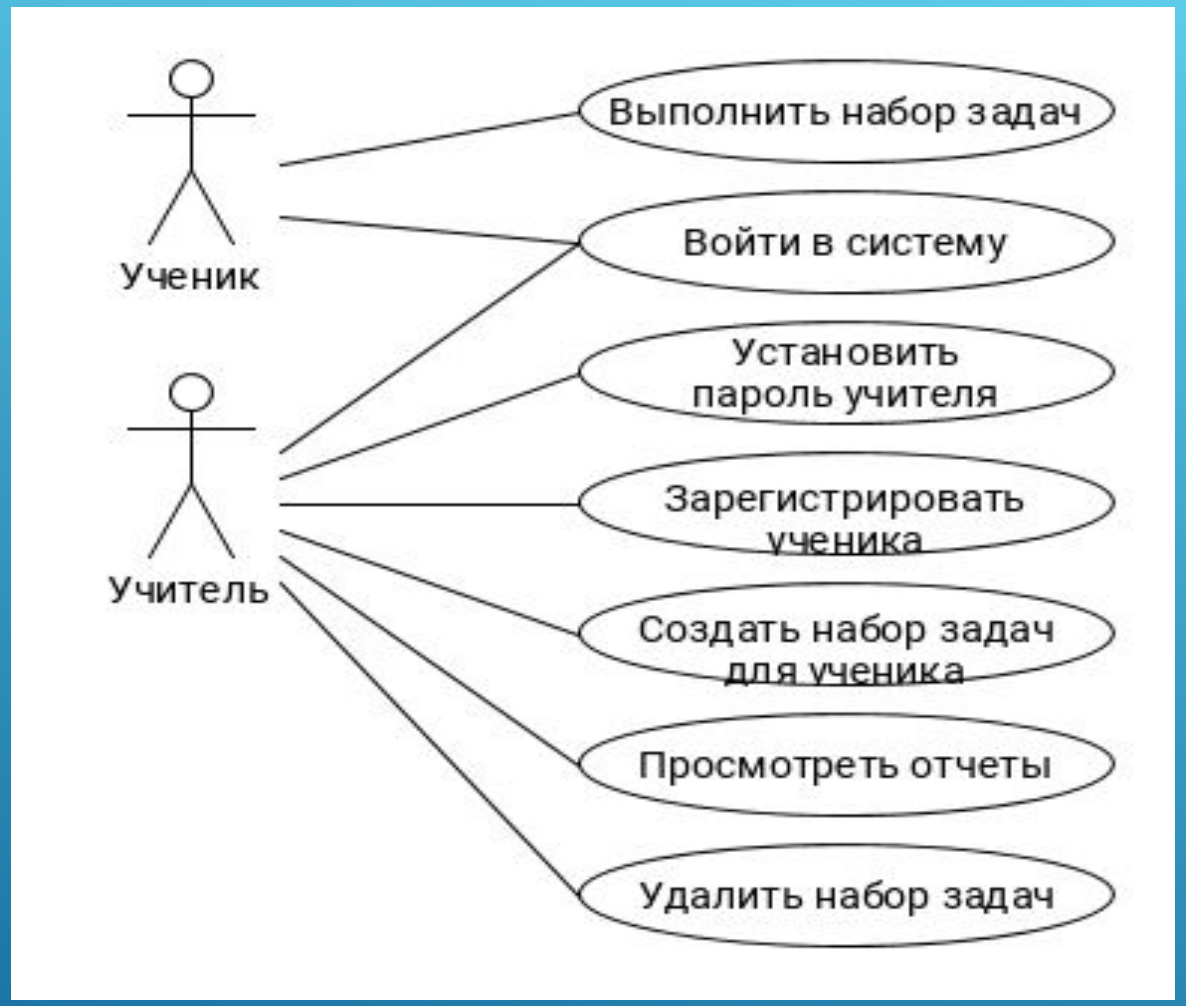
На диаграмме использования изображаются:

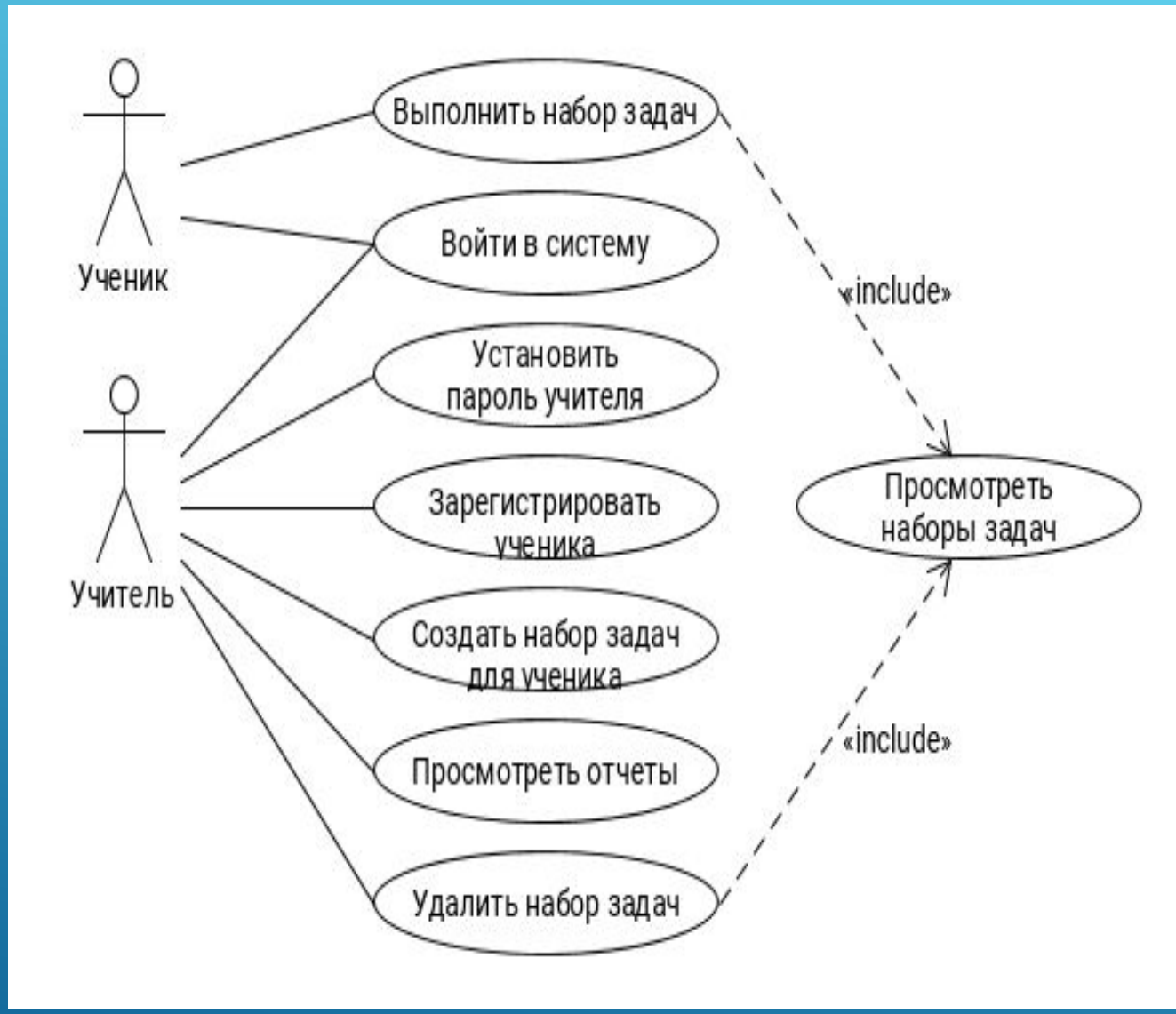
- актеры – группы лиц или систем, взаимодействующих с нашей системой;
- варианты использования (прецеденты) – сервисы, которые наша система предоставляет актерам;
- комментарии;
- отношения между элементами диаграммы.








Ученик
Учитель
Выполнить набор задач
Войти в систему
Установить пароль учителя
Зарегистрировать ученика
Создать набор задач для ученика
Просмотреть отчеты
Удалить набор задач


Ученик
Учитель
Выполнить набор задач
Войти в систему
Установить пароль учителя
Зарегистрировать ученика
Создать набор задач для ученика
Просмотреть отчеты
Удалить набор задач
«include»
«include»
Просмотреть наборы задач

ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ (ACTIVITY DIAGRAM)

- Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Результат может привести к изменению состояния системы или возвращению некоторого значения.
- Диаграмма деятельности отличается от традиционной блок-схемы

более высоким уровнем абстракции;

возможностью представления с помощью диаграмм деятельности управления параллельными потоками наряду с последовательным управлением.

- Основными направлениями использования диаграмм деятельности являются

визуализация особенностей реализации операций классов;

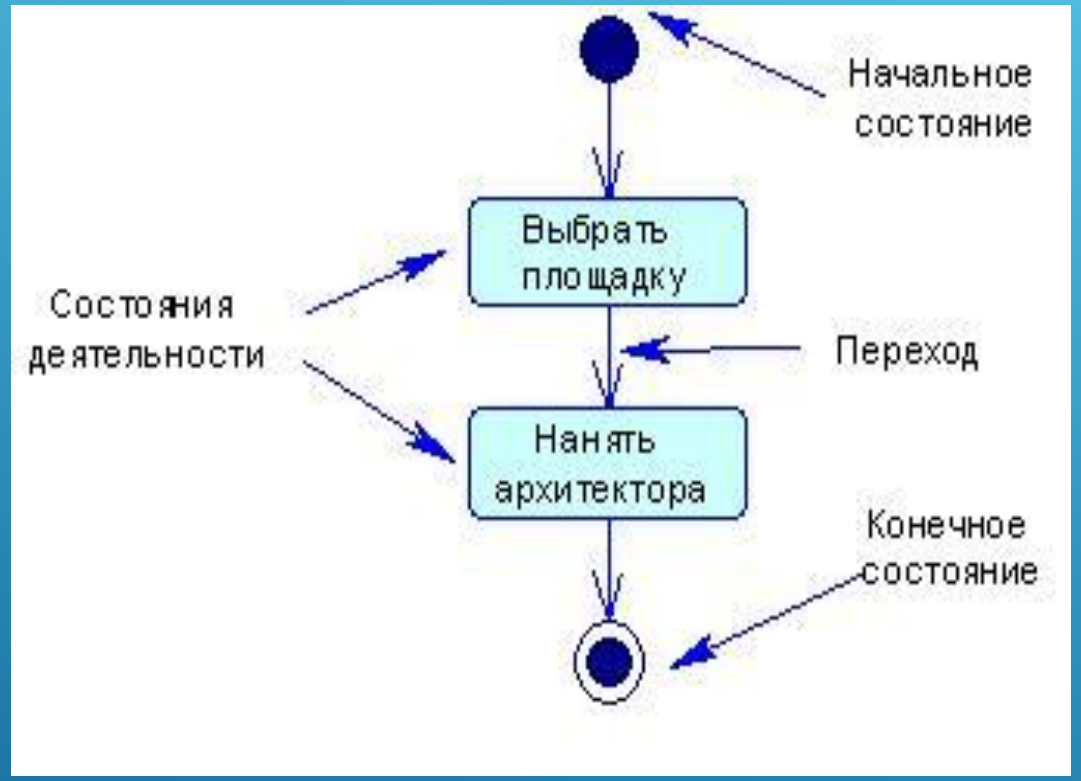
отображение внутрисистемной точки зрения на прецедент.

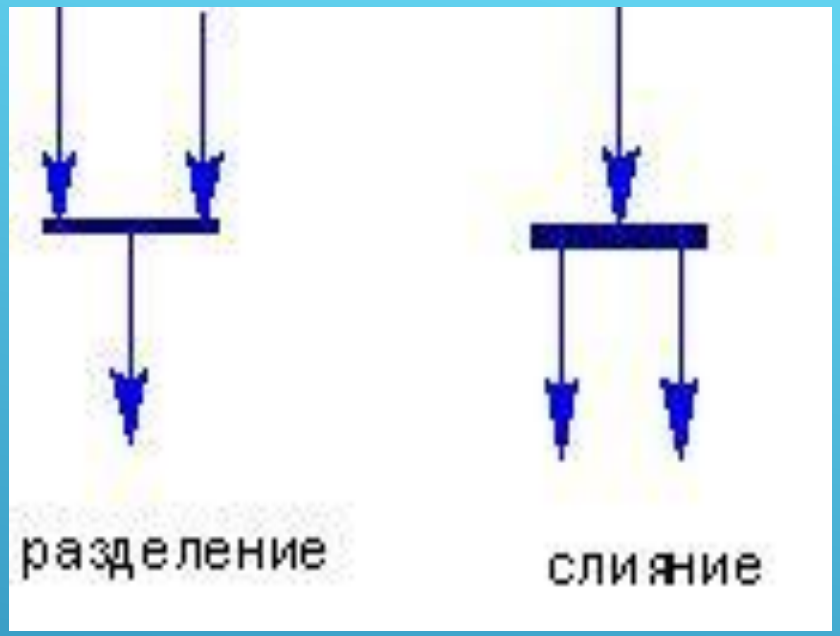
- Разработка диаграммы деятельности преследует цели:

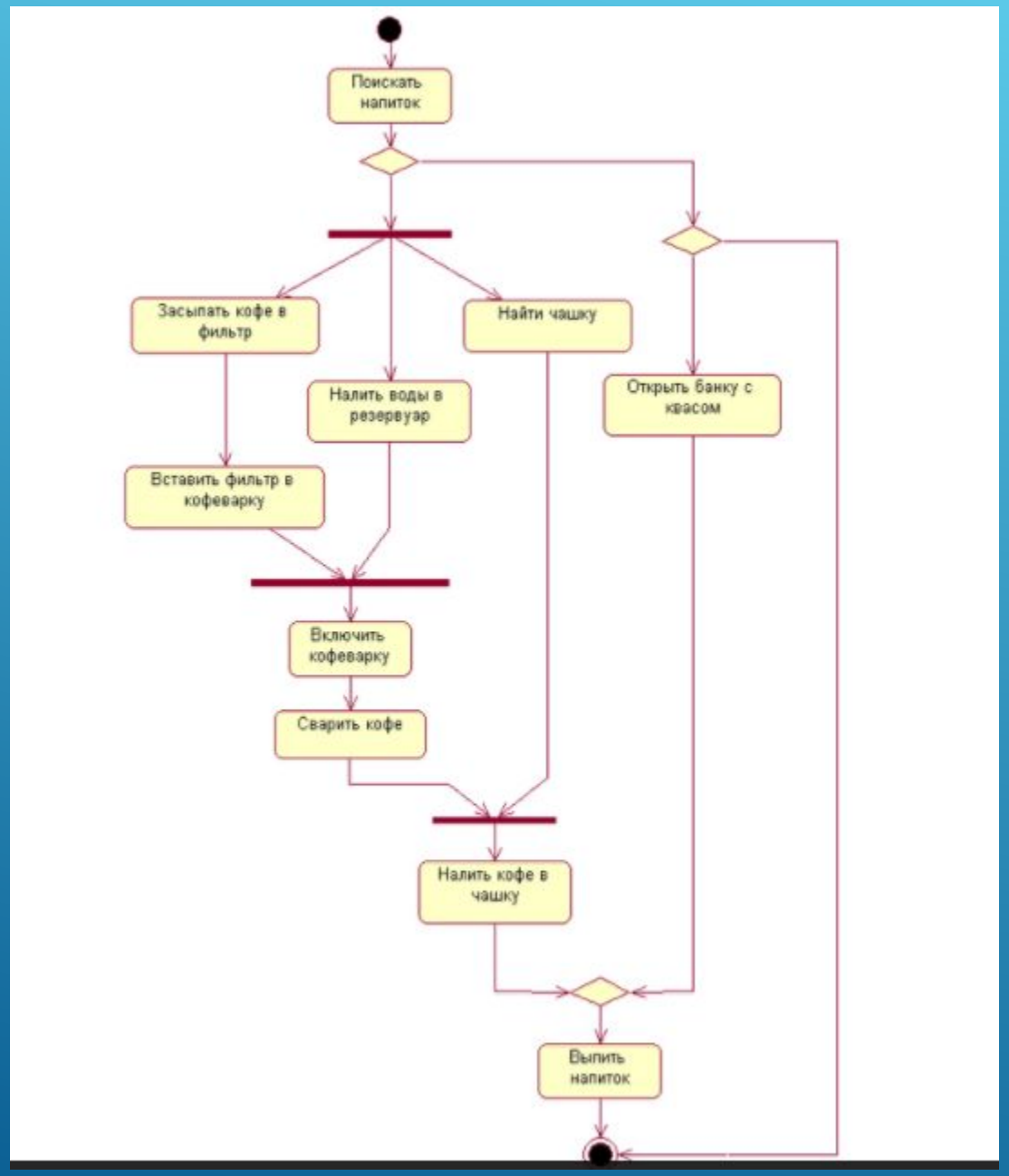
детализировать особенности алгоритмической и логической реализации выполняемых системой операций и прецедентов;

выделить последовательные и параллельные потоки управления;

подготовить детальную документацию для взаимодействия разработчиков системы с ее заказчиками и проектировщиками.


Начальное состояние
Выбрать площадку
Состояния деятельности
Переход
Нанять архитектора
Конечное состояние


разделение
слияние


Поискать напиток
Засыпать кофе в фильтр
Найти чашку
Налить воды в резервуар
Открыть банку с квасом
Вставить фильтр в кофеварку
Включить кофеварку
Сварить кофе
Налить кофе в чашку
Выпить напиток

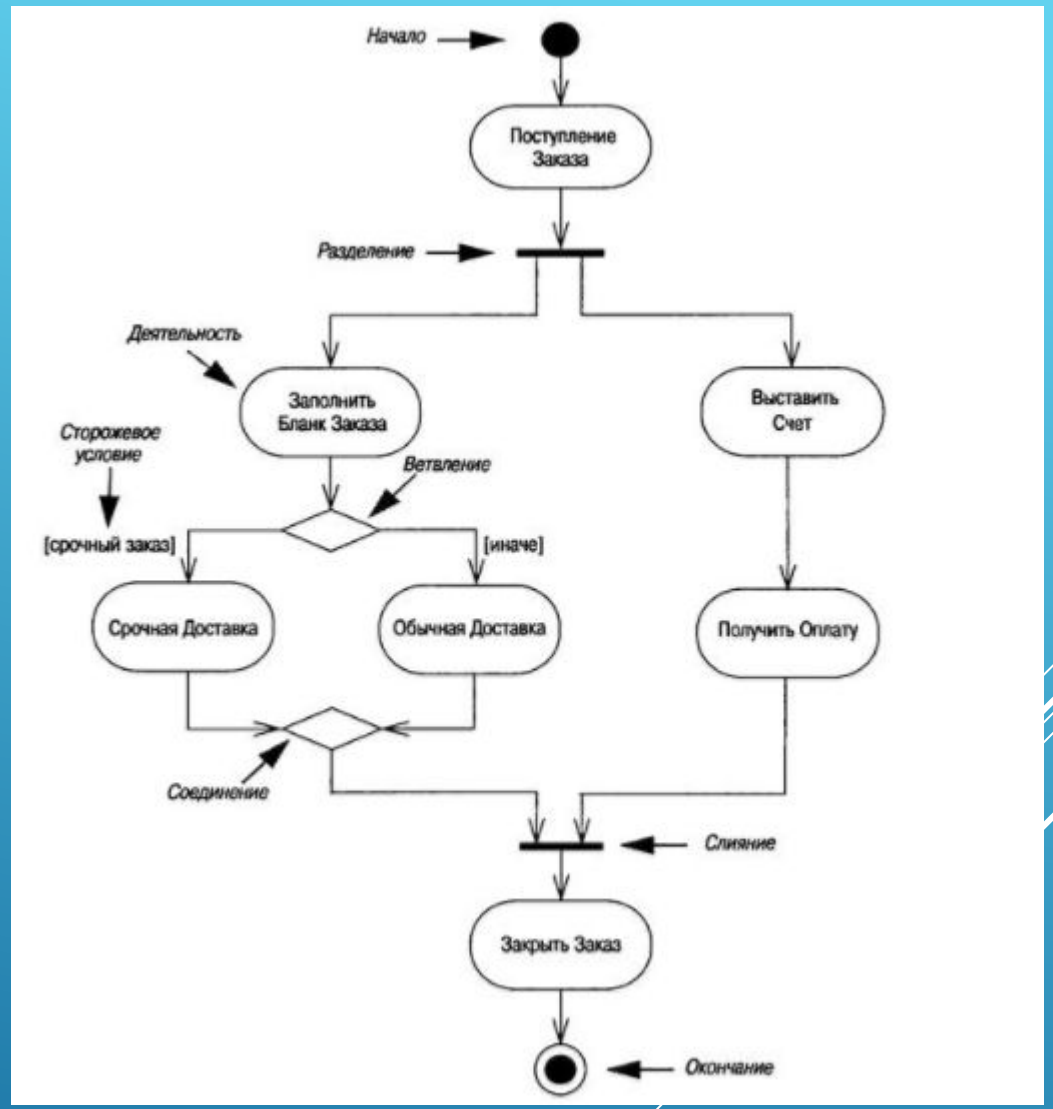

Начало
Поступление Заказа
Разделение
Деятельность
Заполнить Бланк Заказа
Выставить Счет
Сторожевое условие
Ветвление
[срочный заказ]
[иначе]
Срочная Доставка
Обычная Доставка
Получить Оплату
Соединение
Слияние
Закрыть Заказ
Окончание

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ (SEQUENCE DIAGRAMS)

- Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.
- Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций.
- На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.
- Каждое сообщение представляется в виде стрелки между линиями жизни двух объектов. Сообщения появляются в том порядке, как они показаны на странице сверху вниз. Каждое сообщение помечается как минимум именем сообщения. При желании можно добавить также аргументы и некоторую управляющую информацию. Можно показать самоделегирование (self-delegation) – сообщение, которое объект посылает самому себе, при этом стрелка сообщения указывает на ту же самую линию жизни.

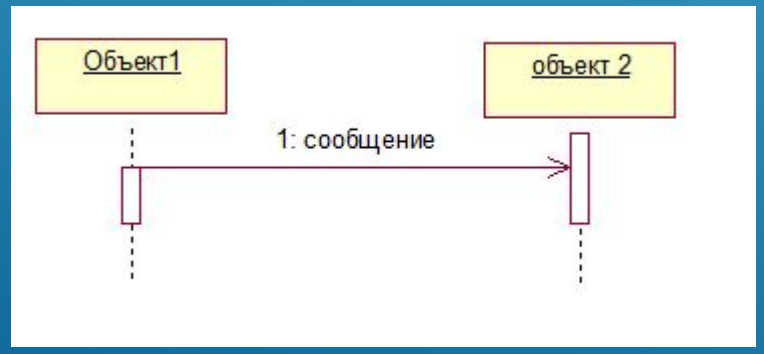


Диаграмма последовательности для прецедента авторизация

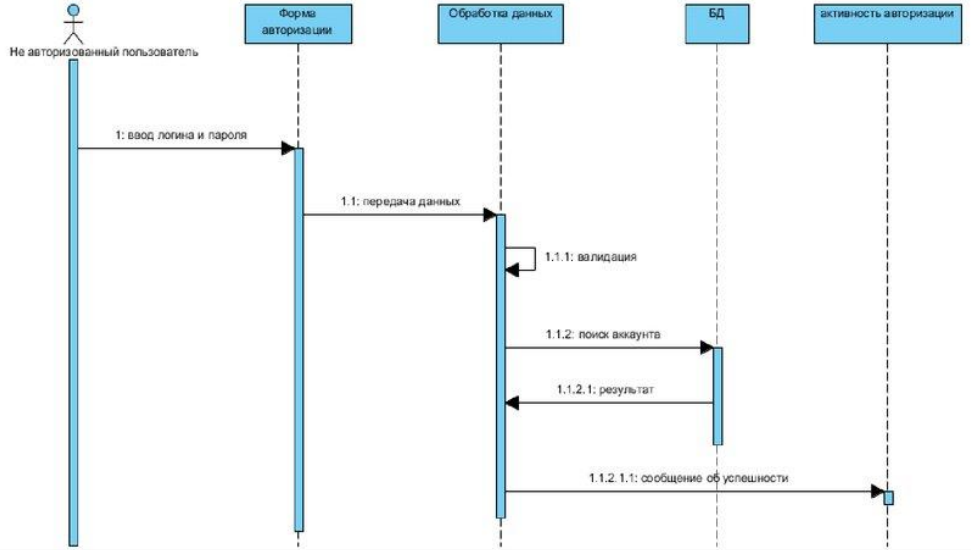


Диаграмма последовательности для прецедента регистрация

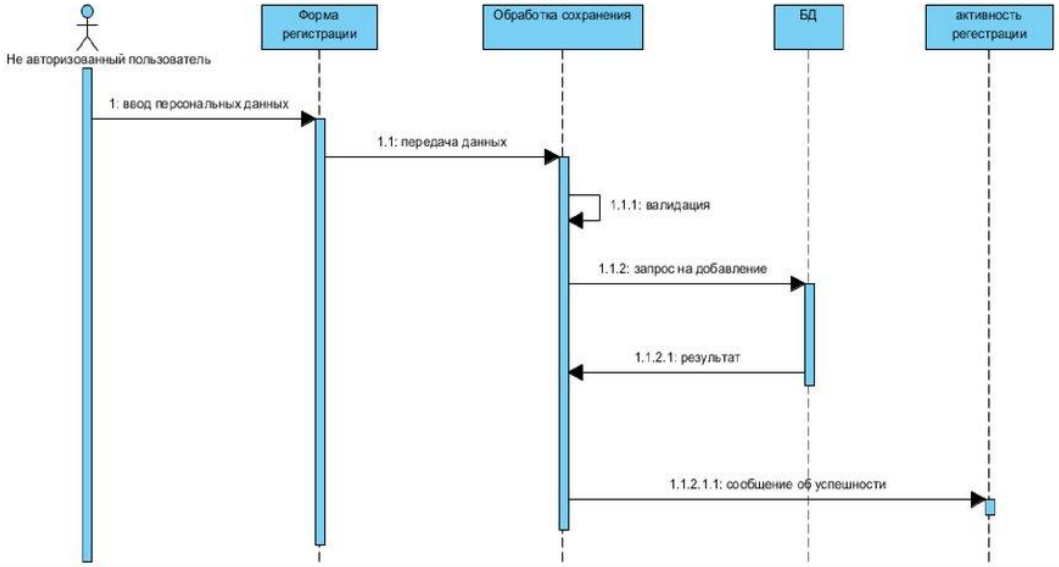


ДИАГРАММА КООПЕРАЦИИ (COLLABORATION DIAGRAM)

- Диаграммы кооперации отображают поток событий через конкретный сценарий варианта использования, упорядочены по времени, а кооперативные диаграммы больше внимания заостряют на связях между объектами.
- На диаграмме кооперации представлена вся та информация, которая есть и на диаграмме последовательности, но кооперативная диаграмма по-другому описывает поток событий. Из нее легче понять связи между объектами, однако, труднее уяснить последовательность событий.
- На кооперативной диаграмме так же, как и на диаграмме последовательности, стрелки обозначают сообщения, обмен которыми осуществляется в рамках данного варианта использования. Их временная последовательность указывается путем нумерации сообщений.

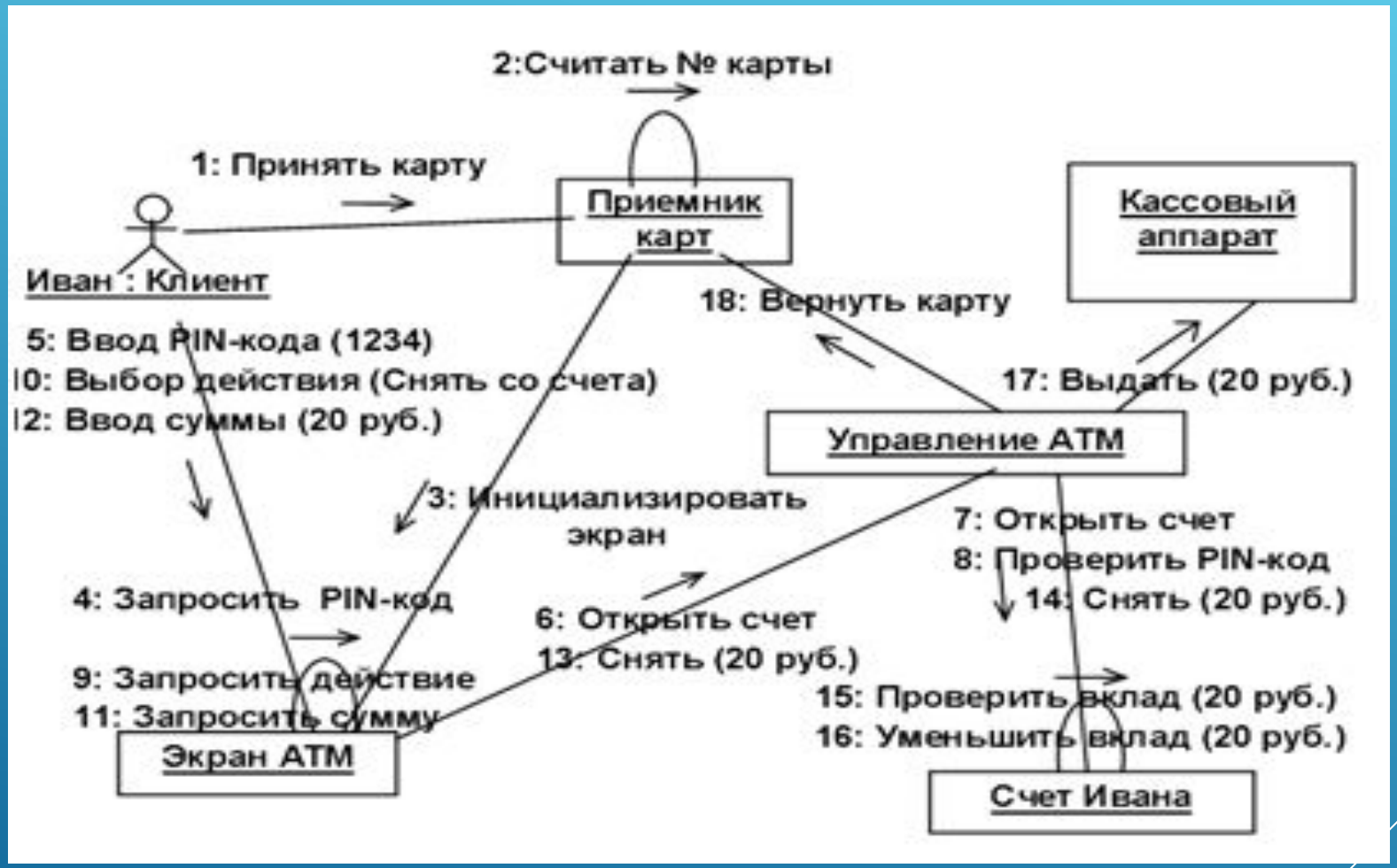
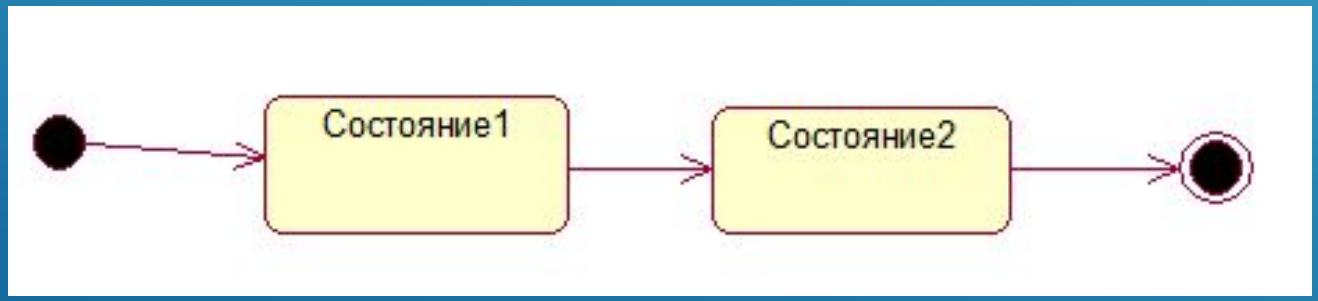

2:Считать № карты
1: Принять карту
Приемник карт
Кассовый аппарат
Иван : Клиент
18: Вернуть карту
5: Ввод PIN-кода (1234)
10: Выбор действия (Снять со счета)
12: Ввод суммы (20 руб.)
17: Выдать (20 руб.)
Управление ATM
3: Инициализировать экран
7: Открыть счет
8: Проверить PIN-код
14: Снять (20 руб.)
4: Запросить PIN-код
6: Открыть счет
13: Снять (20 руб.)
9: Запросить действие
11: Запросить сумму
15: Проверить вклад (20 руб.)
16: Уменьшить вклад (20 руб.)
Экран ATM
Счет Ивана

ДИАГРАММА СОСТОЯНИЙ (STATECHART DIAGRAM)

- Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Очевидно, что диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и диаграммы деятельности).
- Диаграмма состояний полезна при моделировании жизненного цикла объекта (как и ее частная разновидность - диаграмма деятельности).
- От других диаграмм диаграмма состояний отличается тем, что *описывает процесс изменения состояний только одного экземпляра определенного класса* - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

- **Скругленные прямоугольники** представляют **состояния**, через которые проходит объект в течение своего жизненного цикла.
- **Стрелками** показываются **переходы** между состояниями, которые вызваны выполнением методов описываемого диаграммой объекта.
- Существует также два вида псевдосостояний: **начальное**, в котором находится объект сразу после его создания (обозначается **сплошным кружком**), и **конечное**, которое объект не может покинуть, если перешел в него (обозначается **кружком, обведенным окружностью**).



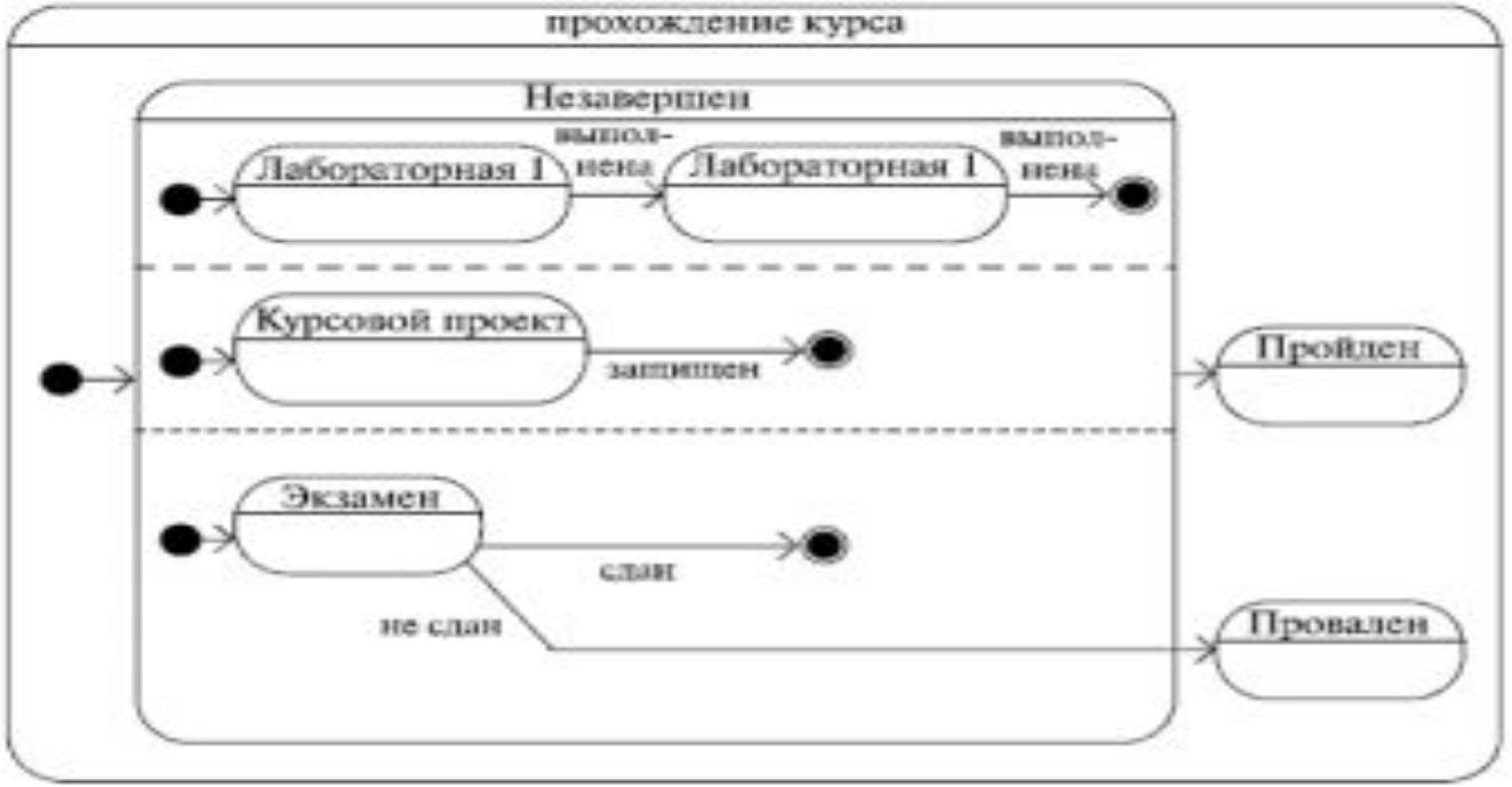
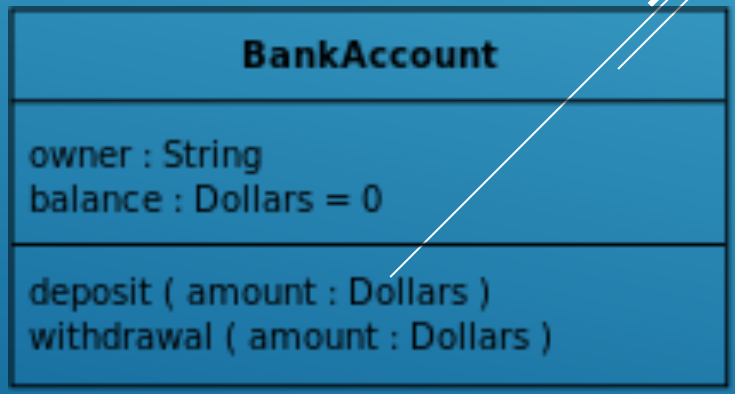

прохождение курса
Незавершен
выполнена
Лабораторная 1
выполнена
Лабораторная 1
Курсовой проект
защищен
Экзамен
сдан
не сдан
Пройден
Провален

Рис. 2.19:

ДИАГРАММА КЛАССОВ

- Класс - это группа сущностей (объектов), обладающих сходными свойствами, а именно, данными и поведением. Отдельный представитель некоторого класса называется объектом класса или просто объектом.
- Под поведением объекта в UML понимаются любые правила взаимодействия объекта с внешним миром и с данными самого объекта.
- На диаграммах класс изображается в виде прямоугольника со сплошной границей, разделенного горизонтальными линиями на 3 секции:
- Верхняя секция (секция имени) содержит имя класса и другие общие свойства (в частности, стереотип).
- В средней секции содержится список атрибутов
- В нижней - список операций класса, отражающих его поведение (действия, выполняемые классом).
- Любая из секций атрибутов и операций может не изображаться (а также обе сразу). Для отсутствующей секции не нужно рисовать разделительную линию и как-либо указывать на наличие или отсутствие элементов в ней.



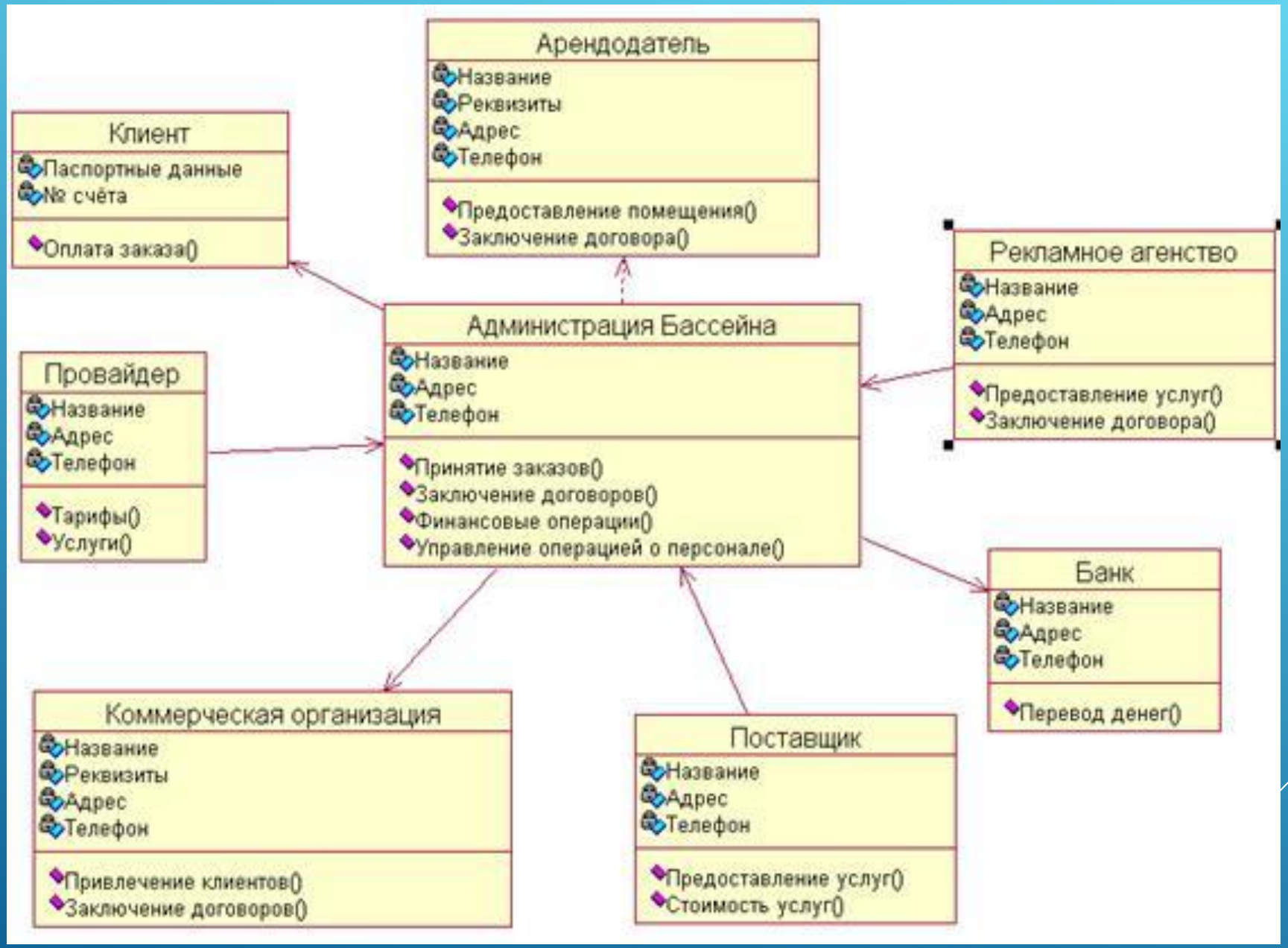
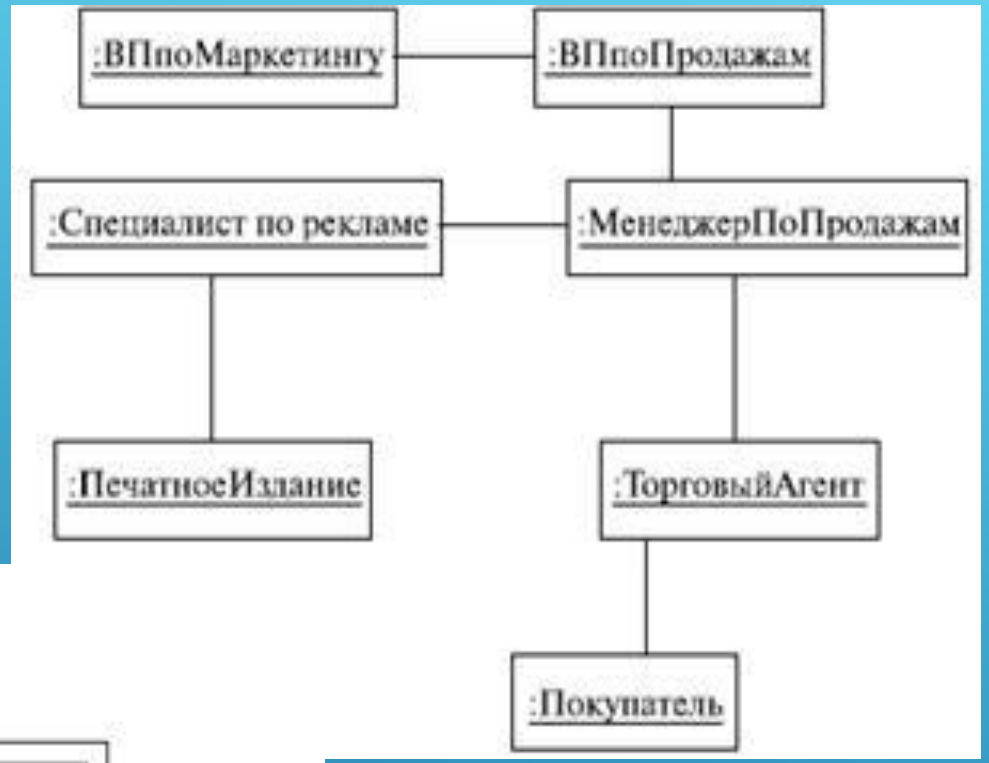
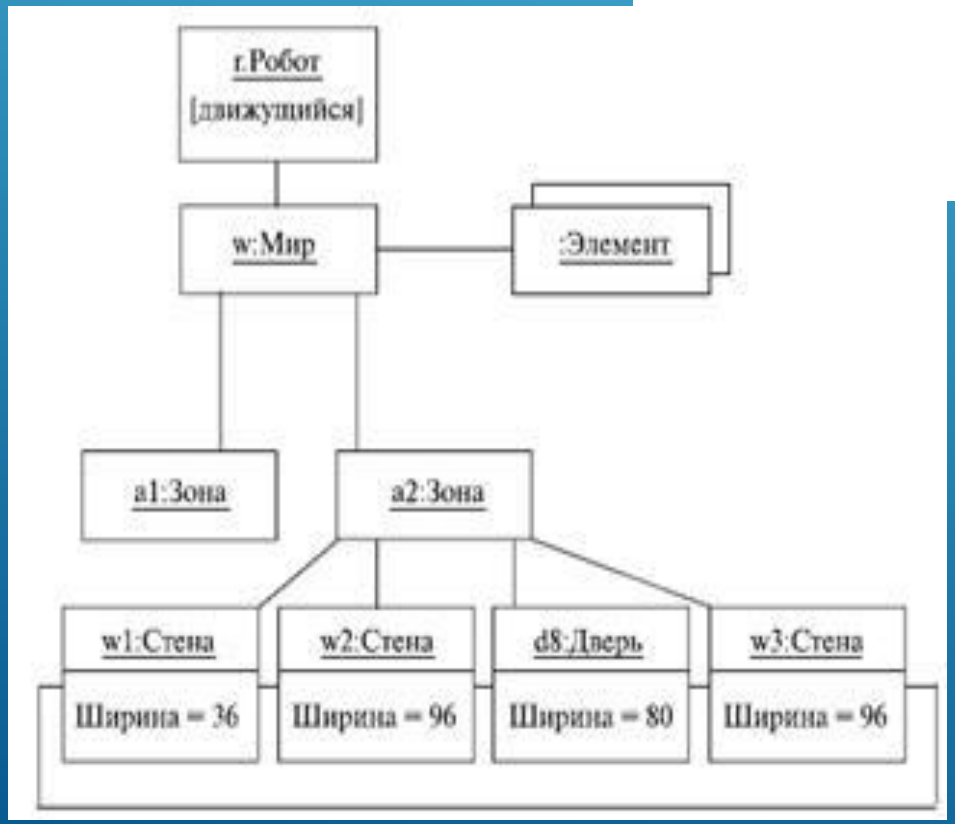
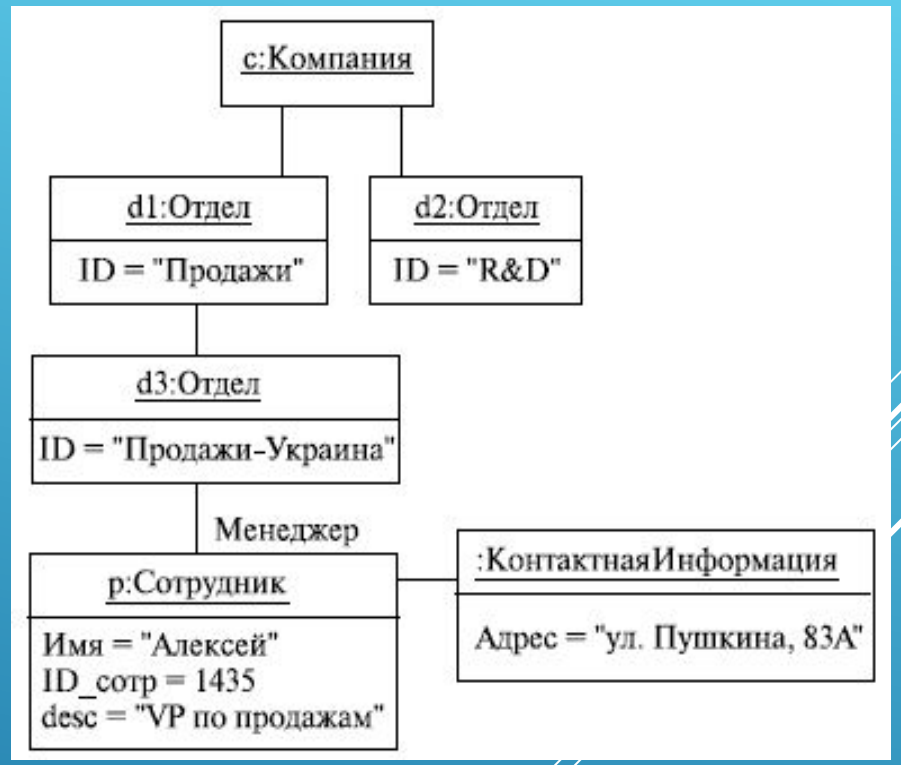

Арендодатель
Название
Реквизиты
Адрес
Телефон
Предоставление помещения()
Заключение договора()
Клиент
Паспортные данные
№ счёта
Оплата заказа()
Рекламное агенство
Название
Адрес
Телефон
Предоставление услуг()
Заключение договора()
Администрация Бассейна
Название
Адрес
Телефон
Принятие заказов()
Заключение договоров()
Финансовые операции()
Управление операцией о персонале()
Провайдер
Название
Адрес
Телефон
Тарифы()
Услуги()
Банк
Название
Адрес
Телефон
Перевод денег()
Коммерческая организация
Название
Реквизиты
Адрес
Телефон
Привлечение клиентов()
Заключение договоров()
Поставщик
Название
Адрес
Телефон
Предоставление услуг()
Стоимость услуг()

ДИАГРАММА ОБЪЕКТОВ (OBJECT DIAGRAM)

- объект, *как и класс, обозначается прямоугольником, но его имя подчеркивается*. Под словом имя здесь мы понимаем название объекта и наименование его класса, разделенные двоеточием.
- Показывает множество объектов - экземпляров классов (изображенных на диаграмме классов) и отношений между ними в некоторый момент времени.
- *диаграмма объектов - это своего рода снимок состояния системы в определенный момент времени*, показывающий множество объектов, их состояния и отношения между ними в данный момент.
- *диаграммы объектов* представляют *статический вид системы с точки зрения проектирования и процессов*, являясь основой для сценариев, описываемых диаграммами взаимодействия.
- *диаграмма объектов используется для пояснения и детализации диаграмм взаимодействия*, например, диаграмм последовательностей. Впрочем, авторам курса очень редко доводилось применять этот тип диаграмм.


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

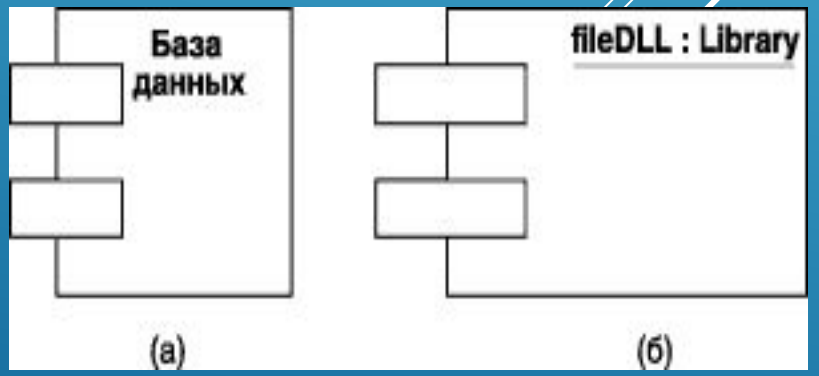

r:Робот
[движущийся]
w:Мир
:Элемент
a1:Зона
a2:Зона
w1:Стена
Ширина = 36
w2:Стена
Ширина = 96
d8:Дверь
Ширина = 80
w3:Стена
Ширина = 96


c:Компания
d1:Отдел
ID = "Продажи"
d2:Отдел
ID = "R&D"
d3:Отдел
ID = "Продажи-Украина"
Менеджер
p:Сотрудник
Имя = "Алексей"
ID_сотр = 1435
desc = "VP по продажам"
:КонтактнаяИнформация
Адрес = "ул. Пушкина, 83А"

В языке *UML* для физического представления моделей систем используются так называемые диаграммы реализации, которые включают в себя две отдельные *канонические диаграммы*: диаграмму компонентов и *диаграмму развертывания*.

ДИАГРАММА КОМПОНЕНТОВ

- *Диаграмма компонентов* описывает особенности физического представления системы.
- *Диаграмма компонентов* позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными *компонентами*, в роли которых может выступать исходный, бинарный и *исполняемый код*.
- Во многих средах разработки *модуль* или *компонент* соответствует файлу.
- Пунктирные стрелки, соединяющие *модули*, показывают отношения взаимозависимости, аналогичные тем, которые имеют *место* при компиляции исходных текстов программ.
- Основными графическими элементами диаграммы *компонентов* являются *компоненты*, *интерфейсы* и зависимости между ними.
- ***Компонент*** (component) — физически существующая часть системы, которая обеспечивает реализацию классов и отношений, а также функционального поведения моделируемой программной системы.



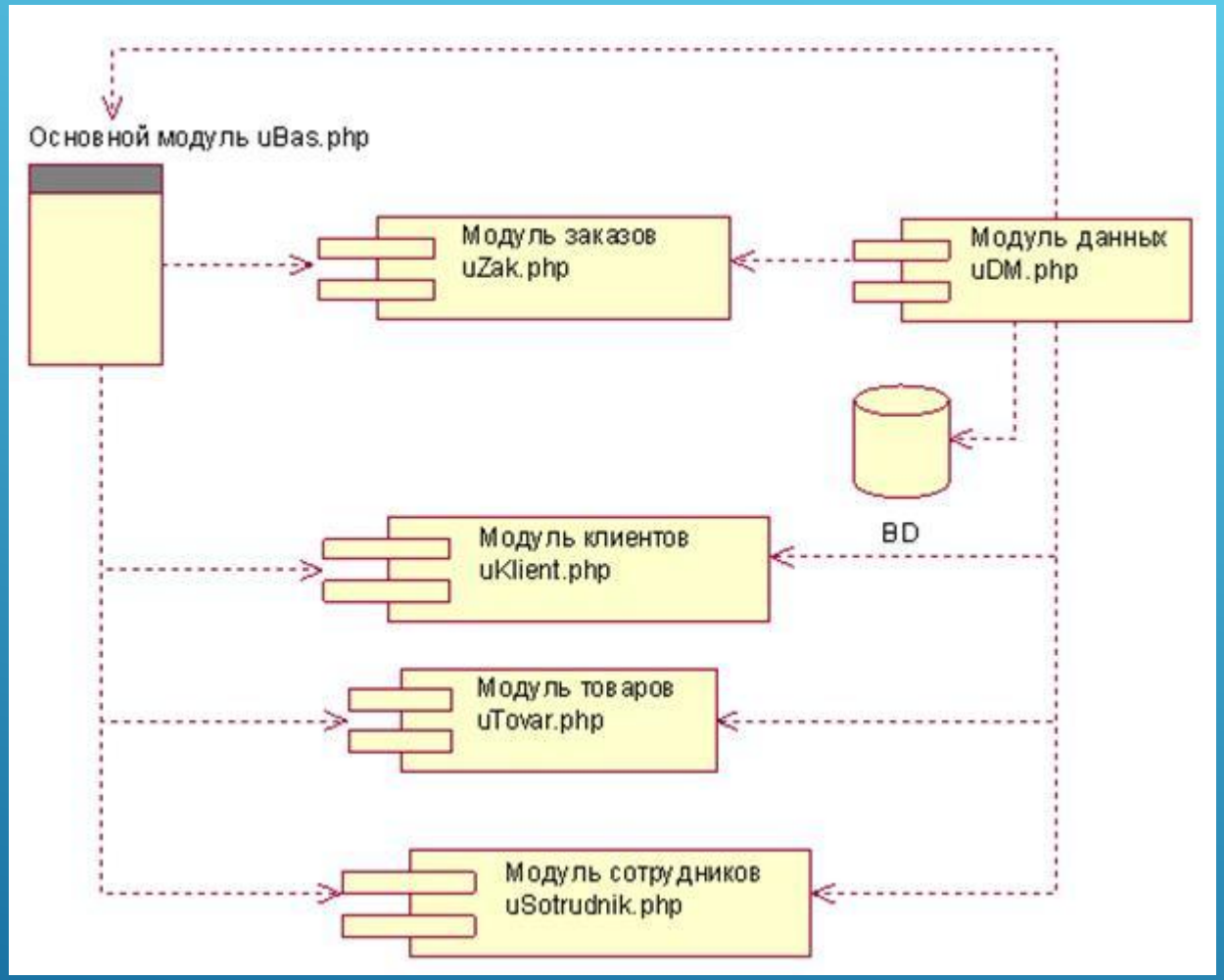
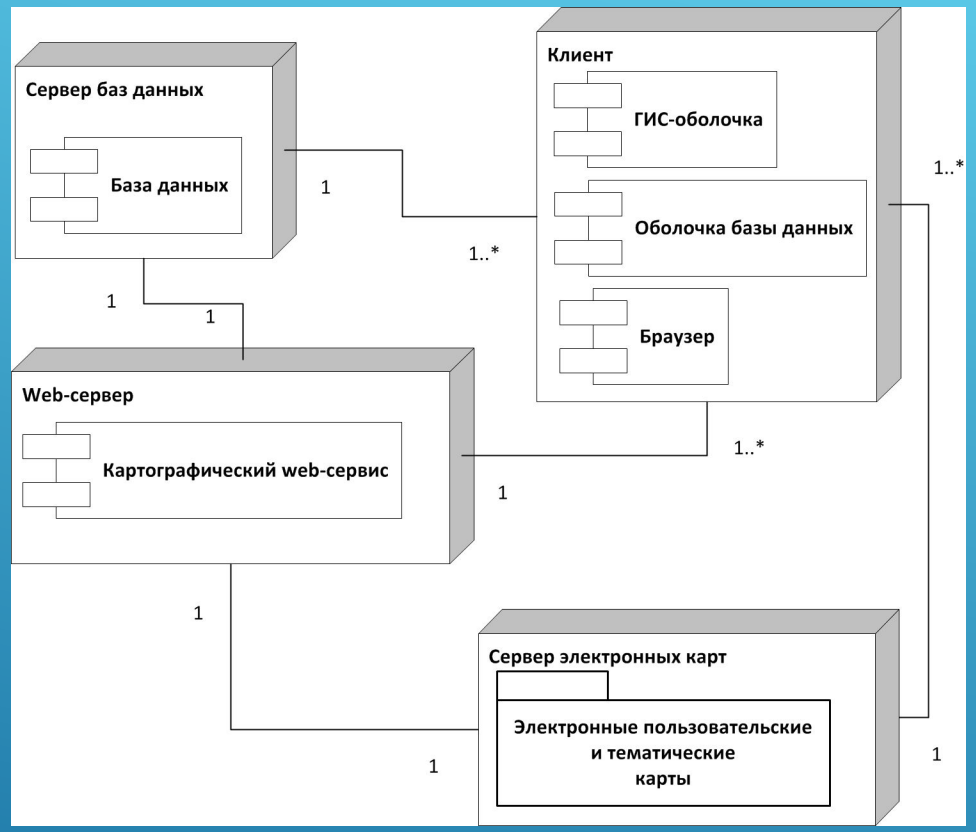
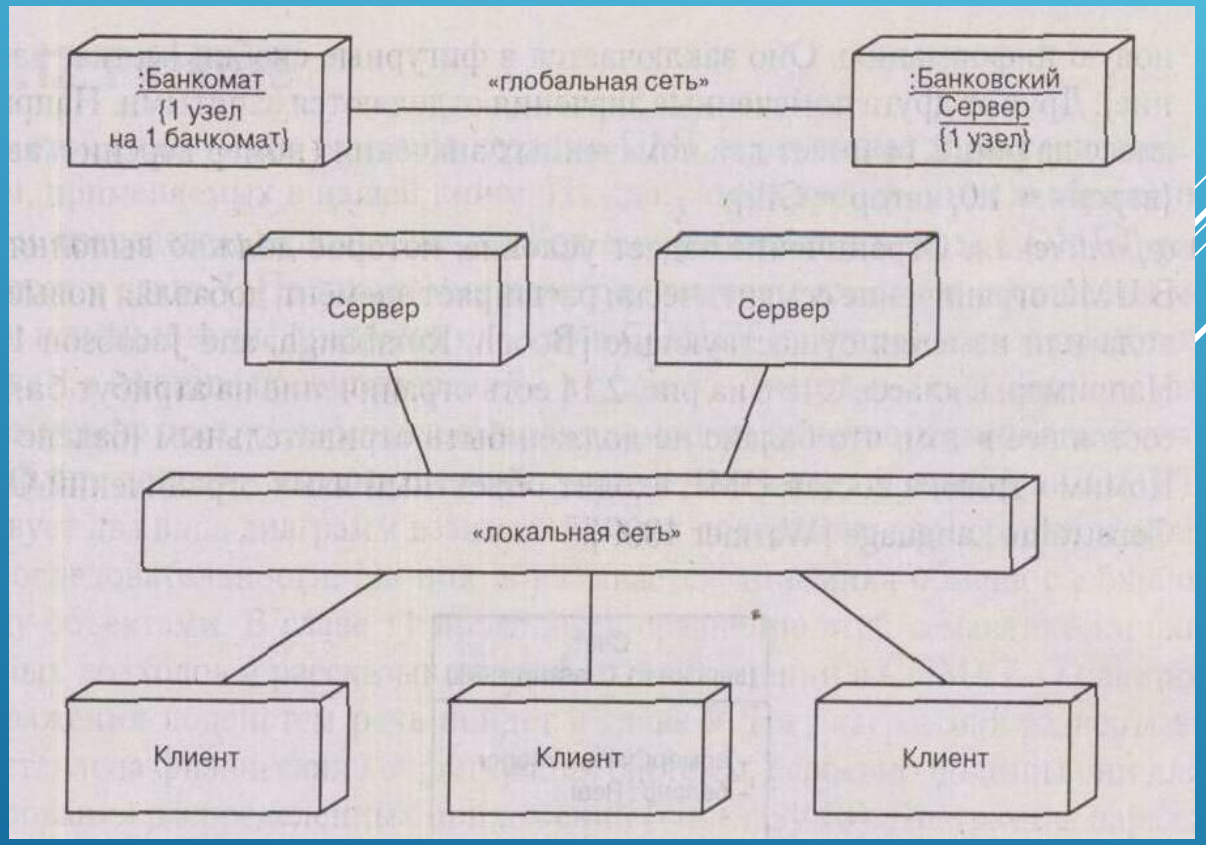

Основной модуль uBas.php
Модуль заказов
uZak.php
Модуль данных
uDM.php
BD
Модуль клиентов
uKlient.php
Модуль товаров
uTovar.php
Модуль сотрудников
uSotrudnik.php

ДИАГРАММА РАЗВЕРТЫВАНИЯ (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. Диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками.


Сервер баз данных
База данных
Клиент
ГИС-оболочка
Оболочка базы данных
Браузер
Web-сервер
Картографический web-сервис
Сервер электронных карт
Электронные пользовательские и тематические карты
1
1..*


:Банкомат
{1 узел на 1 банкомат}
«глобальная сеть»
:Банковский Сервер
{1 узел}
Сервер
Сервер
«локальная сеть»
Клиент
Клиент
Клиент