Лекция №6

Модели анализа, проектирования, реализации

РТУ МИРЭА, кафедра ППИ

СОДЕРЖАНИЕ ЛЕКЦИИ:

- Модель реализации (диаграммы стадии реализации)

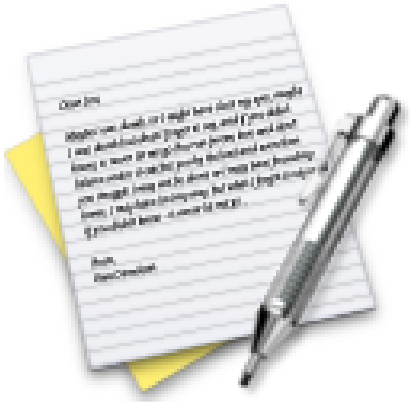
Вопрос 1

Вопрос 2

- Диаграммы компонентов. Правила и рекомендации по построению диаграмм компонентов.

- Диаграммы размещения. Правила и рекомендации по построению диаграмм размещения

Вопрос 3



Ахмедова Х.Г. *email: h.ahmedova@mail.ru*

Различные аспекты диаграмм на разных стадиях ЖЦ

- Диаграммы стадии анализа отражают концептуальные аспекты построения модели систем.
- Диаграммы стадии проектирования отражают логические аспекты построения модели систем.
- Диаграммы стадии реализации отражают **физические (материальные) аспекты** построения модели систем.

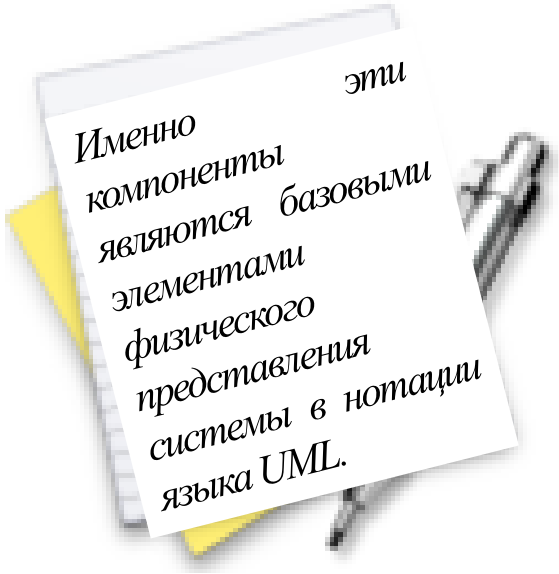
Для создания конкретной физической системы необходимо реализовать все элементы логического представления в конкретные материальные сущности.

Физическая система

- ***Физическая система*** (*physical system*) — реально существующий прототип модели системы.
- Для реализации системы необходимо разработать исходный текст программы на языке программирования: важно разбить исходный код на отдельные модули.

Программный код системы должен быть реализован в форме:

- исполняемых модулей;
- библиотек классов и процедур;
- стандартных графических интерфейсов;
- файлов баз данных.

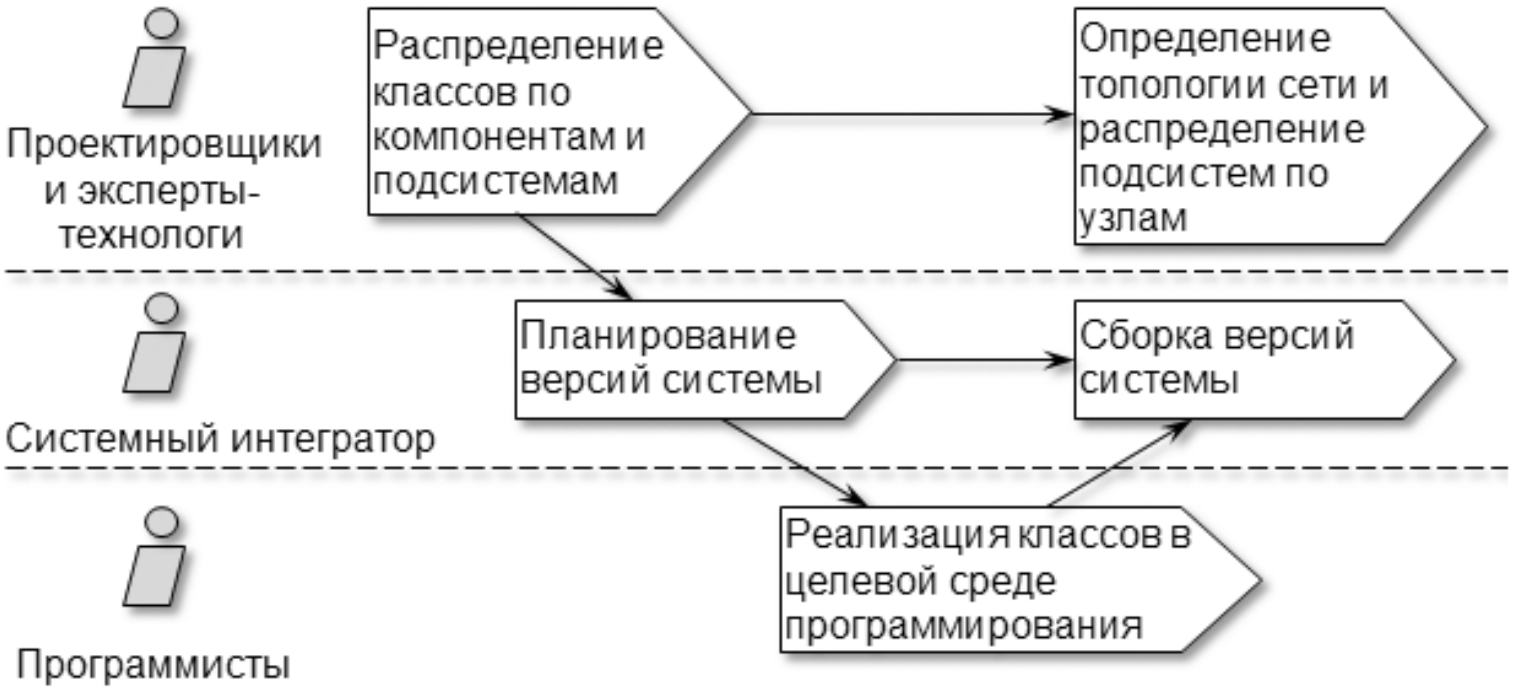


Цели разработки модели реализации

Основная цель - получение работоспособной версии системы

- определение окончательного состава, структуры и кода классов;
- распределение классов по компонентам и подсистемам;
- определение топологии распределенной системы и распределение подсистем по узлам сети;
- планирование итераций (версий) сборки системы;
- сборка версий системы

Алгоритм построения модели реализации



Диаграммы (основные артефакты) модели реализации

Диаграмма компонентов

- *описывает особенности физического представления системы*

Диаграмма развертывания

- *позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами*

Исходные тексты программ

План реализации версий системы

Частично или полностью работоспособные версии системы

Диаграмма компонентов

- **Диаграмма компонентов -** статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между ними.
- **Диаграмма компонентов** позволяет определить состав программных компонентов, в роли которых может выступать *исходный, бинарный и исполняемый код, а также установить зависимости между ними.*
- В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.
- **Диаграмма компонентов** обеспечивает согласованный переход от логического представления к конкретной реализации проекта в форме программного кода.
- Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения.

Цель разработки диаграммы компонентов

Цель разработки

- *спецификация общей структуры исходного кода системы;*
- *спецификация исполнимого варианта системы.*

Разработчики диаграммы

- системные аналитики
- архитекторы
- программисты

Основные элементы диаграммы компонентов

Компонент

- физическая часть системы (файлы с исходным кодом классов, библиотеки, исполняемые модули и т.п.)

Интерфейс

- внешне видимый, именованный набор операций, который класс, компонент или подсистема может предоставить другому классу, компоненту или подсистеме, для выполнения им своих функций

Отношение

- ассоциация (отображается между компонентами и их интерфейсами)
- зависимость (означает зависимость реализации одних компонентов от реализации других)

Элемент диаграммы «Компонент»

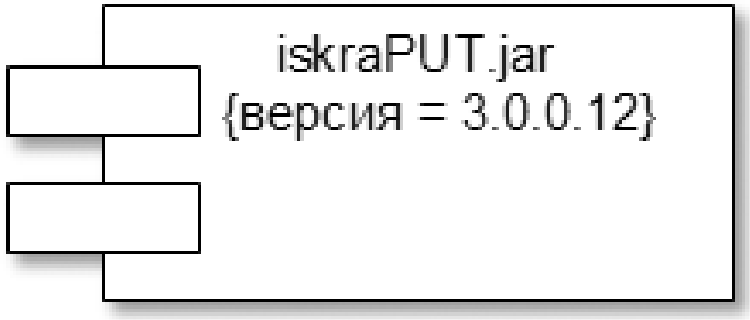
- **Компонент (component)** — физически существующая часть системы, которая обеспечивает реализацию классов и отношений, а также функционального поведения моделируемой программной системы.
- Компонент предназначен для представления физической организации ассоциированных с ним элементов модели.

- Компонентом может быть:
  - исполняемый код отдельного модуля;
  - командные файлы;
  - файлы, содержащие интерпретируемые скрипты.

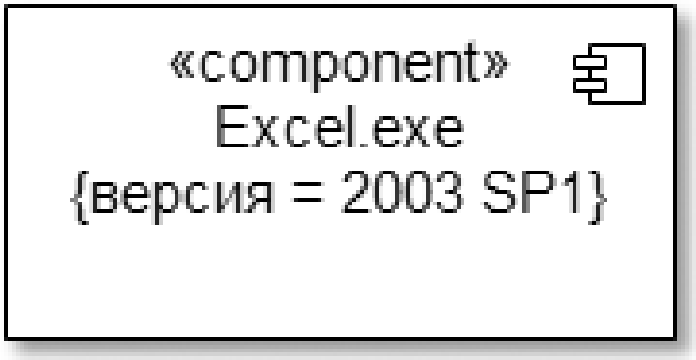
Представление компонента на диаграмме

Собственное графическое представление

- используется специальный символ – прямоугольник со вставленными слева двумя более мелкими прямоугольниками
- внутри объемлющего прямоугольника записывается имя компонента и, возможно, дополнительная информация. Этот символ является базовым обозначением компонента в языке UML.



Текстовый стереотип и помеченные значения

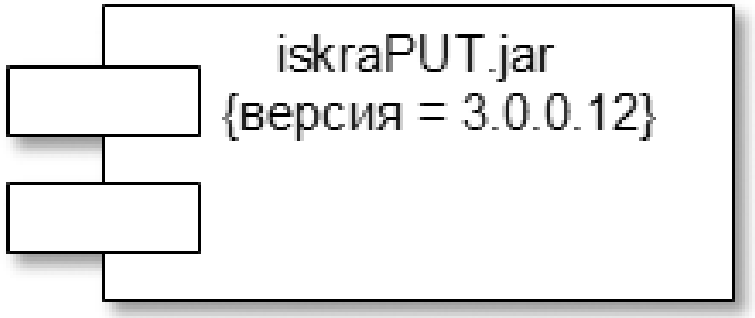


Происхождение изображения компонента

Графическое изображение *компонента* ведет свое происхождение от обозначения *модуля* программы, применявшегося некоторое время для отображения особенностей *инкапсуляции данных* и процедур.

Модуль (module) — часть программной системы, требующая памяти для своего хранения и процессора для исполнения.

- верхний маленький *прямоугольник* концептуально ассоциировался с данными, которые реализует этот *компонент* (иногда он изображается в форме овала);
- нижний маленький *прямоугольник* ассоциировался с операциями или методами, реализуемыми *компонентом*.



В простых случаях имена данных и методов записывались явно в маленьких прямоугольниках, однако в языке UML они не указываются.

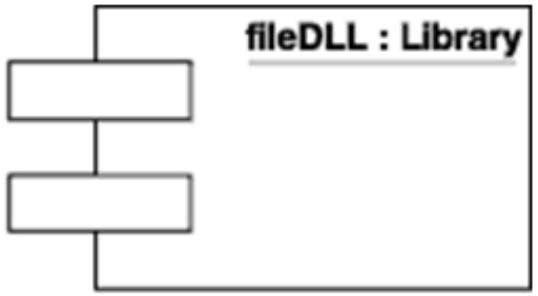
Имя компонента

- **Имя компонента** подчиняется общим правилам именования элементов модели в языке UML и может состоять из любого числа букв, цифр и знаков препинания.

Представление компонентов:

- на уровне типа: записывается только имя типа с заглавной буквы в форме **<Имя типа>**;
- на уровне экземпляра: записывается собственное имя *компонента* и имя типа в форме ***<имя компонента* : Имя типа>, п**ри этом вся строка имени подчеркивается.

(а)



(б)

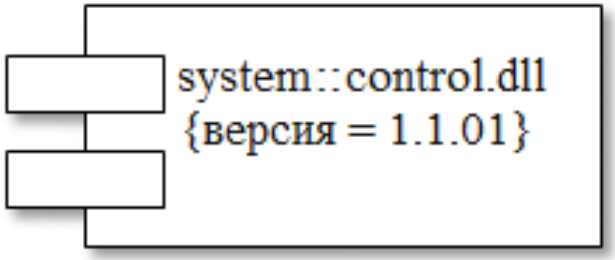
Собственные имена компонентов

- исполняемых файлов
- динамических библиотек
- Web-страниц
- текстовых файлов или файлов справки
- файлов баз данных
- файлов с исходными текстами программ
- файлов скриптов и другие

Обозначение компонента

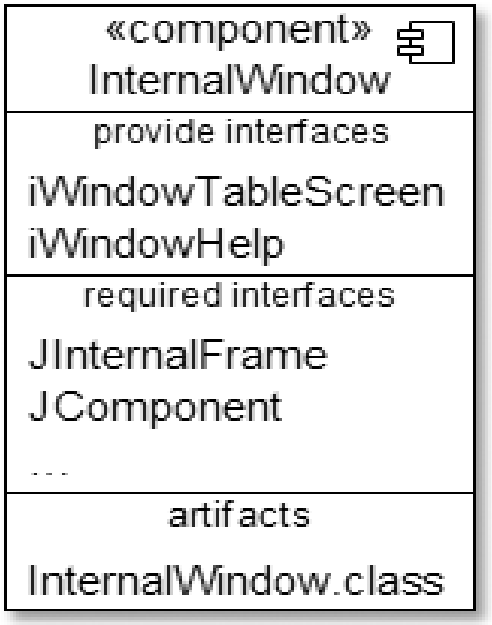
Простое

- к простому имени *компонента* может быть добавлена *информация* об имени объемлющего пакета и о конкретной версии реализации данного *компонента* (номер версии)



Расширенное

- символ *компонента* разделен на секции, чтобы явно указать имена реализованных в нем классов или *интерфейсов*



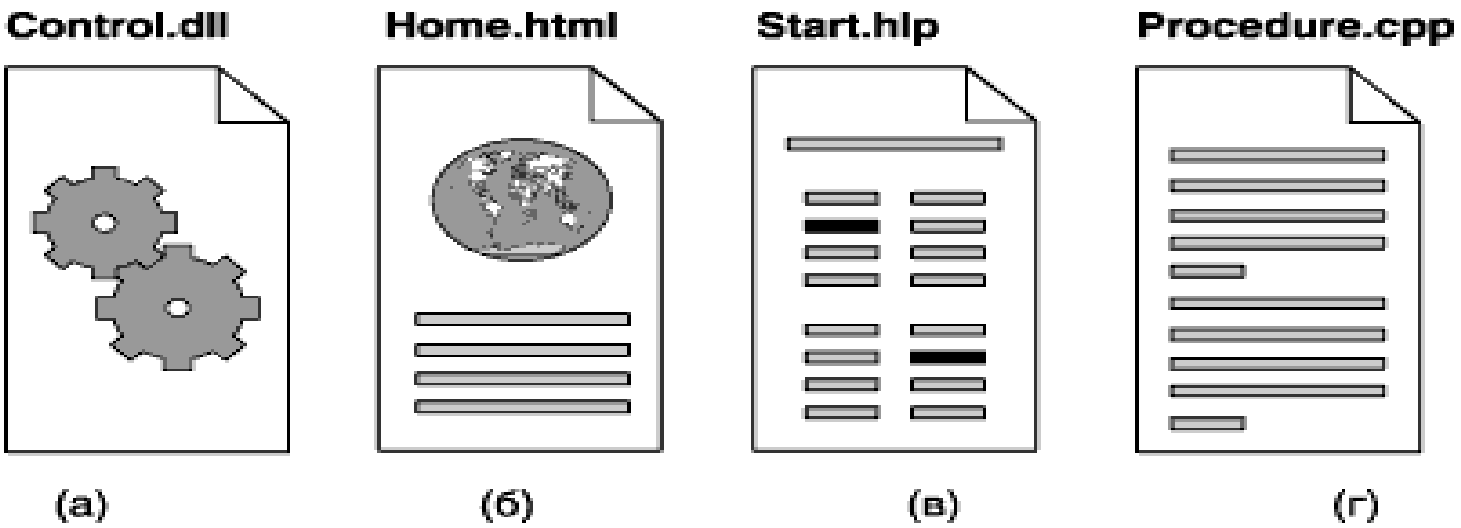
Графическое изображение компонентов: стереотипы

стереотипы для компонентов развертывания

- обеспечивают непосредственное выполнение системой своих функций: динамически подключаемые библиотеки, Web-страницы на языке разметки гипертекста; файлы справки.

стереотипы для компонентов в форме рабочих продуктов

- как правило – это файлы с исходными текстами программ



Текстовый стереотип компонентов

<<file>> (файл)

- – определяет наиболее общую разновидность *компонента*, который представляется в виде произвольного физического файла.

<<executable>> (исполнимый)

- – определяет разновидность компонента-файла, который является исполнимым файлом и может выполняться на компьютерной платформе.

<<document>> (документ)

- – определяет разновидность компонента-файла, который представляется в форме документа произвольного содержания, не являющегося исполнимым файлом или файлом с исходным текстом программы.

<<library>> (библиотека)

- – определяет разновидность компонента-файла, который представляется в форме динамической или статической библиотеки.

<<source>> (источник)

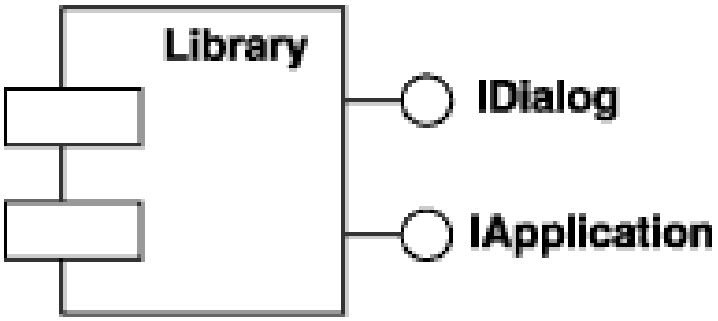
- – определяет разновидность компонента-файла, представляющего собой файл с исходным текстом программы, который после компиляции может быть преобразован в исполнимый файл.

<<table>> (таблица)

- – определяет разновидность *компонента*, который представляется в форме таблицы базы данных.

Элемент «Интерфейс»: графический стереотип

- Интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок

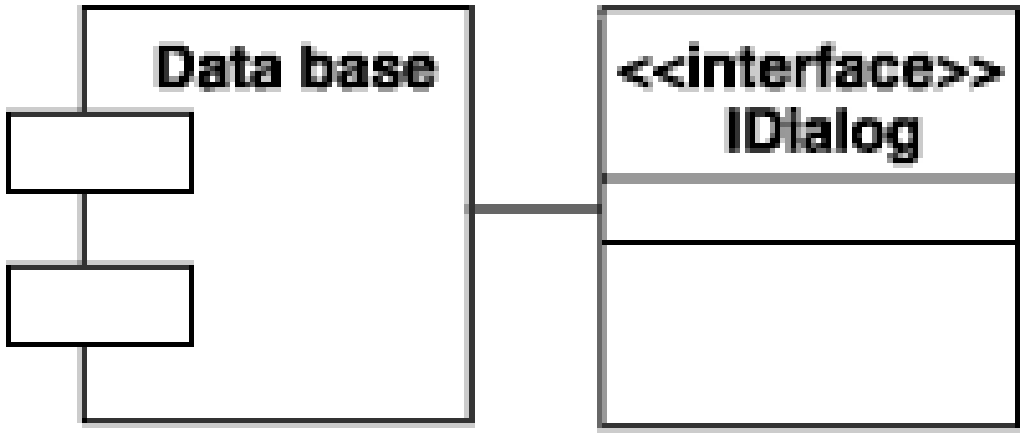


- имя *интерфейса*, рекомендуется начинать с заглавной буквы "I«
- имя *интерфейса* записывается рядом с окружностью.

Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

Элемент «Интерфейс»: текстовый стереотип

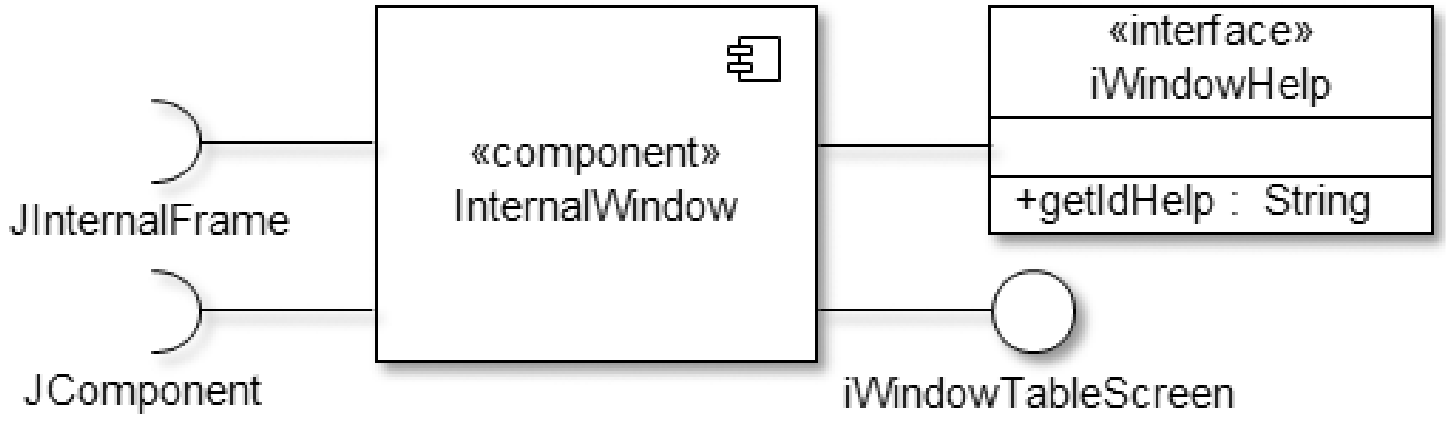
- Интерфейс на диаграмме компонентов может быть изображен в виде прямоугольника класса со стереотипом **<< interface>>** и секцией поддерживаемых операций.



Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса.

Пример отображения интерфейсов

- В некоторых языках программирования, в частности в Java, интерфейс представляет собой отдельный класс, включаемый и реализуемый (конкретизируемый) в части программного кода операций в составе других классов.
- На диаграмме компонентов интерфейс отображается так же, как и на диаграмме классов (слева от компонента необходимые для работы интерфейсы, справа - предоставляемые).



Связь интерфейса и компонента

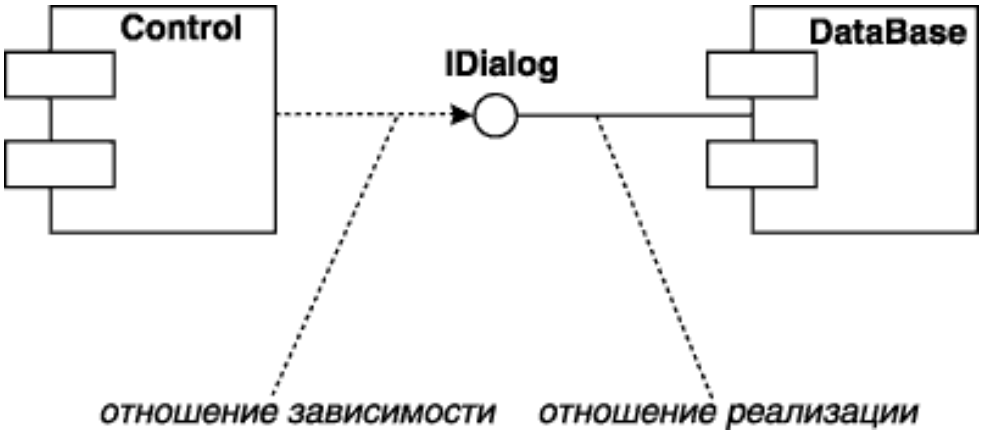
- Если компонент реализует некоторый интерфейс, то такой интерфейс называют экспортируемым или **поддерживаемым**, поскольку этот компонент предоставляет его в качестве сервиса другим компонентам.

- Если же компонент использует некоторый интерфейс, который реализуется другим компонентом, то такой интерфейс для первого компонента называется **импортируемым**.

  - *Особенность импортируемого интерфейса состоит в том, что на диаграмме компонентов это отношение изображается с помощью зависимости.*

Зависимости между компонентами

- *Отношение зависимости* на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента или зависимого элемента к источнику или независимому элементу модели.

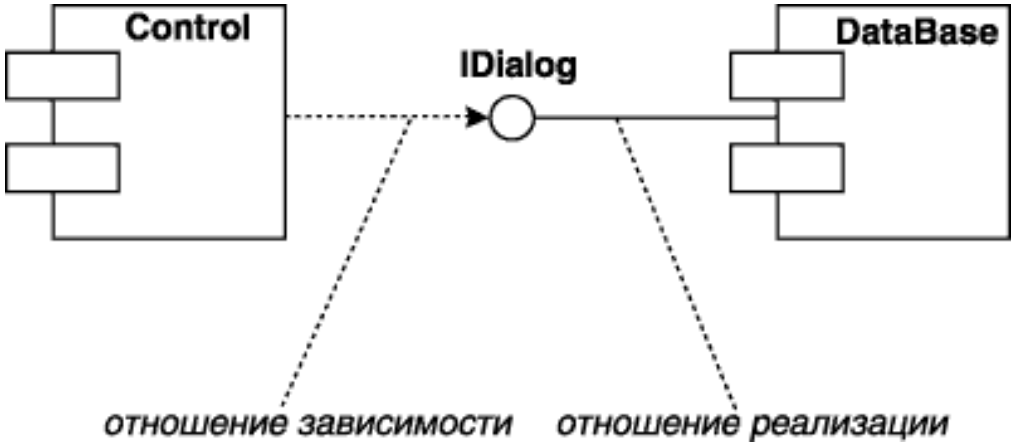


- Зависимости могут **отражать связи отдельных файлов** программной системы как на этапе компиляции, так и на этапе генерации объектного кода.
- Зависимость может указывать на наличие в независимом компоненте описаний классов, которые используются в зависимом компоненте для создания соответствующих объектов.

Роль зависимостей

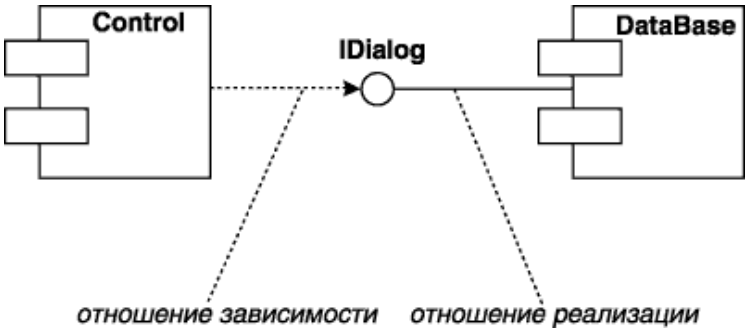
Применительно к диаграмме *компонентов* зависимости могут:

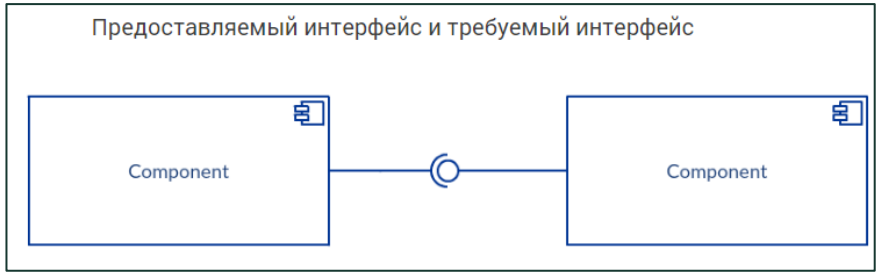
- связывать *компоненты* и импортируемые этим *компонентом интерфейсы;*
- связывать различные виды *компонентов* между собой.



В этом случае рисуют стрелку от компонента-клиента к импортируемому интерфейсу.

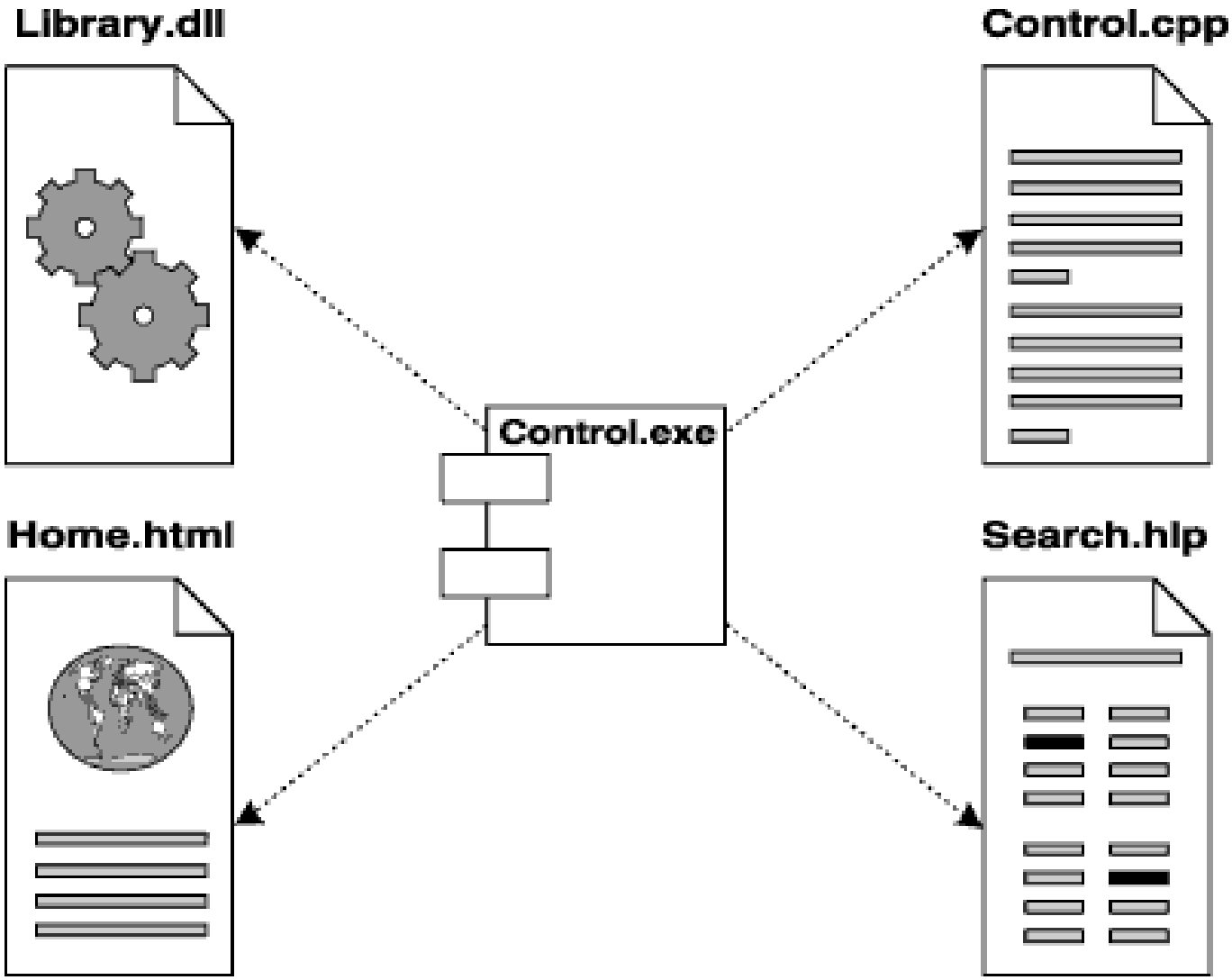
Требуемый и реализуемый интерфейсы



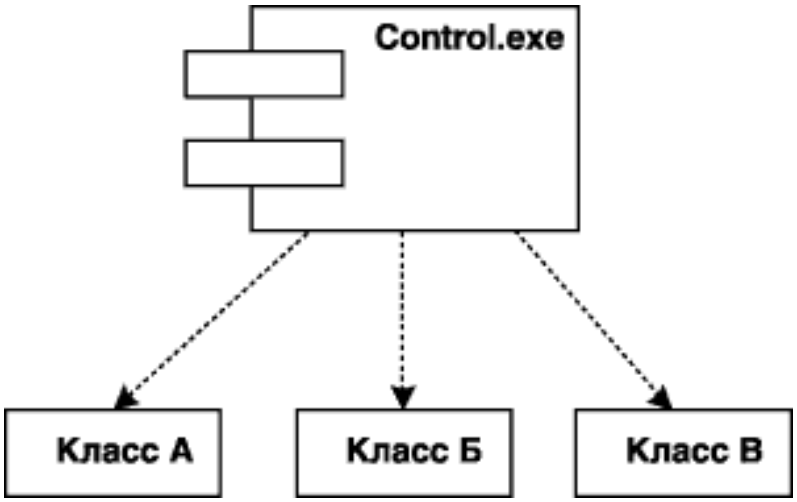


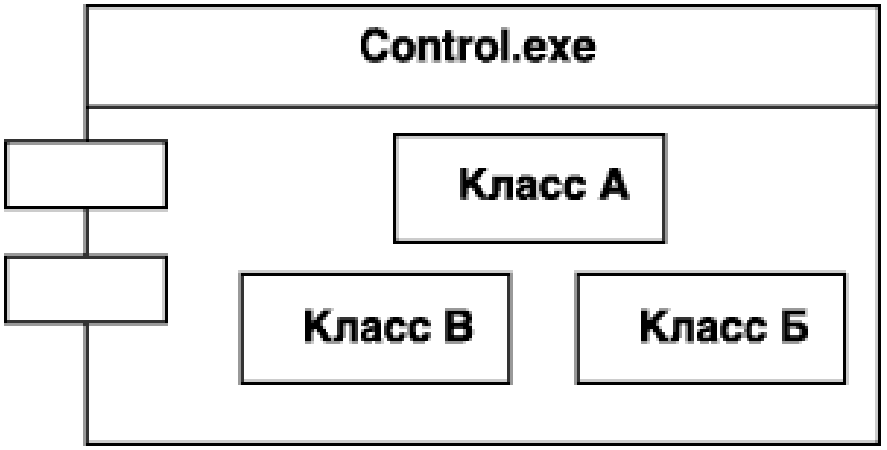
- Наличие пунктирной линии со стрелкой означает, что *компонент* не реализует соответствующий *интерфейс*, а использует его в процессе своего выполнения.
- При этом на этой же диаграмме может присутствовать и другой *компонент*, который реализует этот *интерфейс*. *Отношение* реализации *интерфейса* обозначается на диаграмме *компонентов* обычной линией без стрелки.

Отношение программного вызова и компиляции

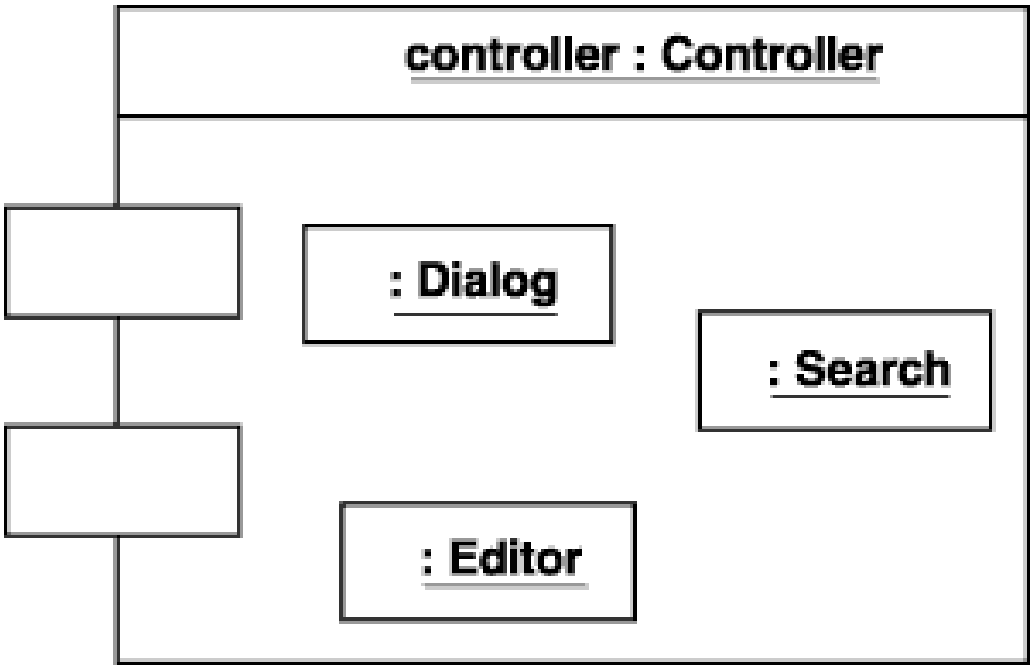


Зависимости между компонентами и реализованными в них классами





Компонент-экземпляр, реализующий отдельные объекты



Рекомендации по построению диаграммы компонентов

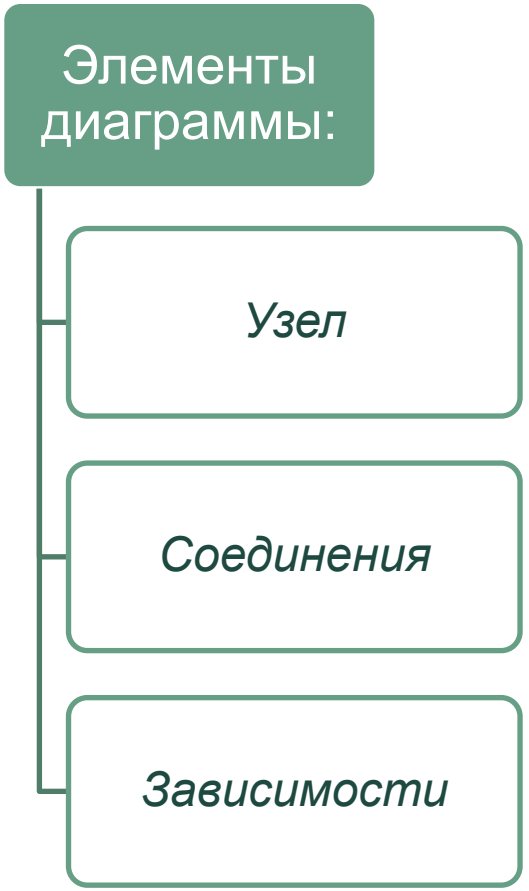
1. Определить состав системы, то есть решить из каких физических частей или файлов будет состоять программная система: *особо обратить внимание на возможность повторного использования кода за счет рациональной декомпозиции компонентов.*

   **Для повышения производительности программной системы рационально использовать вычислительных ресурсы: большую часть описаний классов, их операций и методов вынести в динамические библиотеки, оставив в исполняемых компонентах только самые необходимые для инициализации программы фрагменты программного кода.*
2. Дополнить модель интерфейсами и схемами *базы данных*. При разработке *интерфейсов* следует обращать внимание на согласование различных частей программной системы. Включение в модель схемы *базы данных* предполагает спецификацию отдельных таблиц и установление информационных связей между ними.
3. Установить и нанести на диаграмму взаимосвязи между *компонентами*, а также отношения реализации. Эти отношения должны иллюстрировать все важнейшие аспекты физической реализации системы, начиная с особенностей компиляции исходных текстов программ и заканчивая исполнением отдельных частей программы на этапе ее выполнения. Для этой цели можно использовать различные графические стереотипы *компонентов*.

Диаграмма развертывания

- ***Диаграмма развертывания (deployment diagram)*** - диаграмма, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.
- *Диаграмма развертывания* применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным *узлам* системы, 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.

Диаграммы развертывания: разработчики и элементы



Элемент диаграммы «Узел»

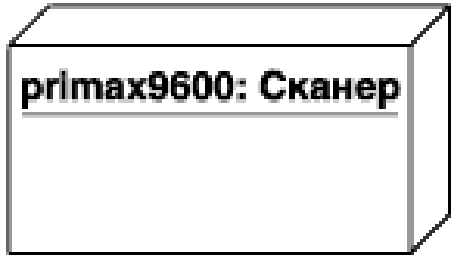
- ***Узел (node)*** представляет собой физически существующий элемент системы, который может обладать вычислительным ресурсом или являться техническим *устройством*.
- В качестве вычислительного ресурса *узла* может рассматриваться: один или несколько процессоров, а также объем электронной или магнитооптической памяти
- Механические или электронные *устройства:* датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Графическое изображение узла: стандартное

Представление узлов:

- на уровне типа: записывается только имя типа узла с заглавной буквы в форме **<Имя типа узла>;**
- на уровне экземпляра: записывается собственное имя *узла* и имя типа в форме **<*имя узла* : Имя типа узла>, п**ри этом вся запись подчеркивается.

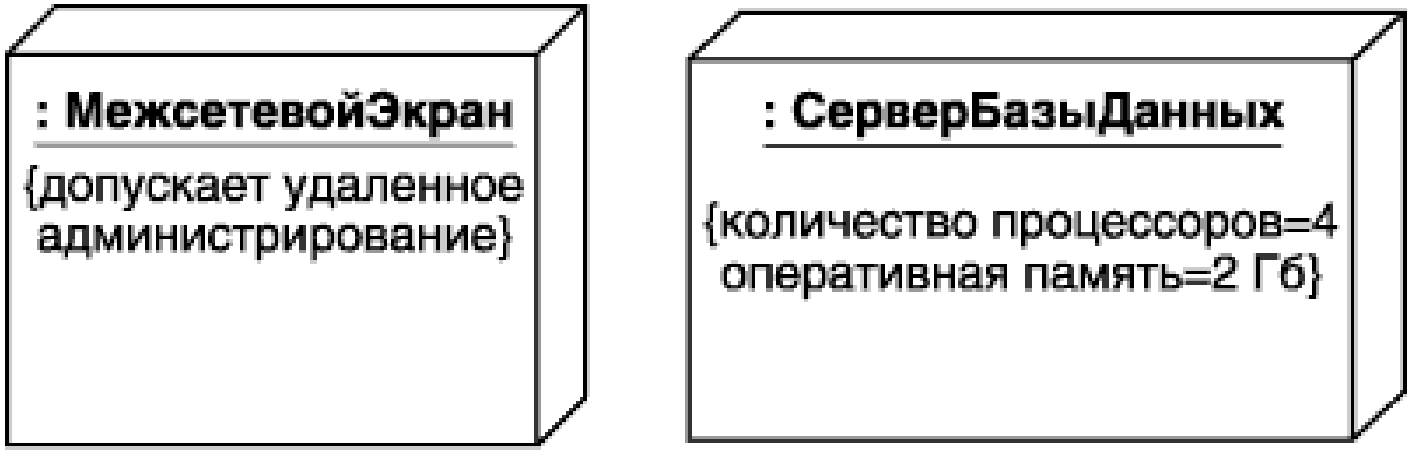
(а)



(б)

- Графически *узел* на *диаграмме развертывания* изображается в форме трехмерного куба.
- *Узел* имеет имя, которое указывается внутри этого графического символа.

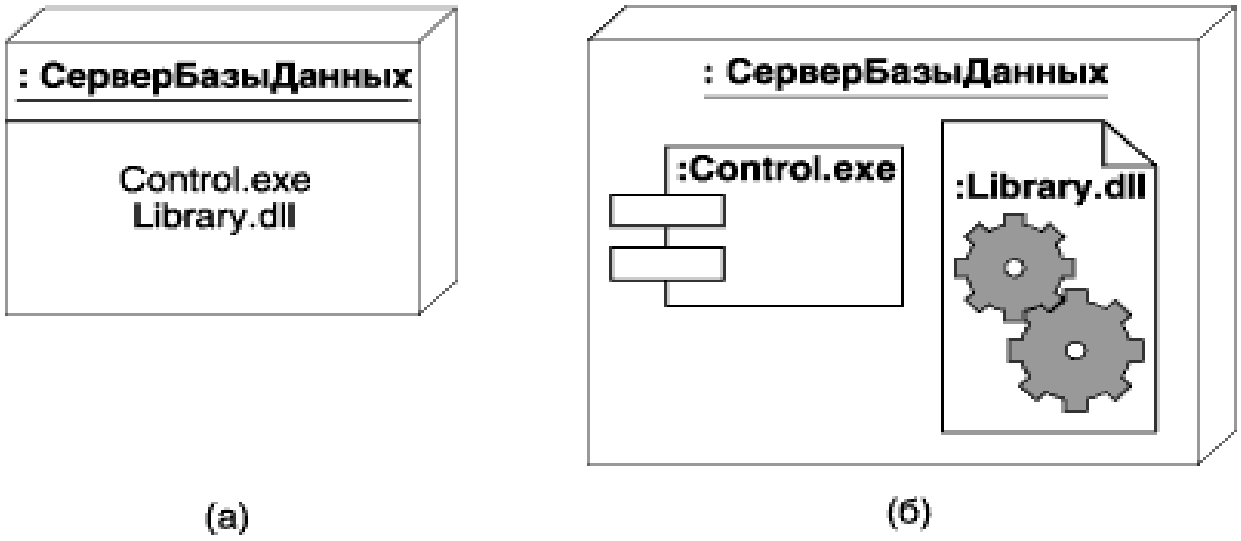
Графическое изображение узла: расширенное



(а) (б)

- Изображения *узлов* могут расширяться, чтобы включить дополнительную информацию о спецификации *узла*.
- Если дополнительная *информация* относится к имени *узла*, то она записывается под этим именем в форме помеченного значения

Указание компонентов на узлах



1 способ

- разделить графический символ *узла* на две секции горизонтальной линией: в верхней секции записывают имя *узла*, а в нижней - размещенные на этом *узле* компоненты

2 способ

- показать на *диаграмме развертывания узлы* с вложенными изображения (только исполняемые компоненты и динамические библиотеки)

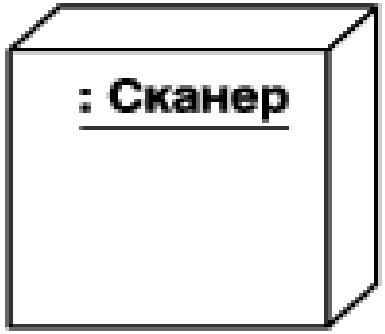
Текстовые стереотипы узла

- "*processor*" (*процессор*)
- "*sensor*" (датчик)
- "*modem*" (*модем*)
- "*net*" (*сеть*)
- "*printer*" (принтер)
- др.

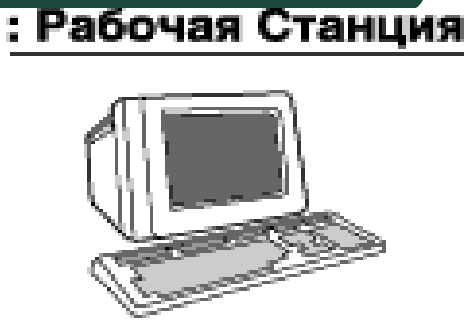
допускаются специальные условные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции.

Графические стереотипы узла

(а)



(б)



(в)

Ресурсоемкий узел (processor), под которым понимается *узел* с процессором и памятью, необходимыми для выполнения исполняемых компонентов.
Изображается в форме куба с боковыми гранями, окрашенными в серый цвет (а).

Устройство (device), под которым понимается *узел* без процессора и памяти (б).
Изображается в форме обычного куба.
Здесь не могут размещаться исполняемые компоненты программной системы.

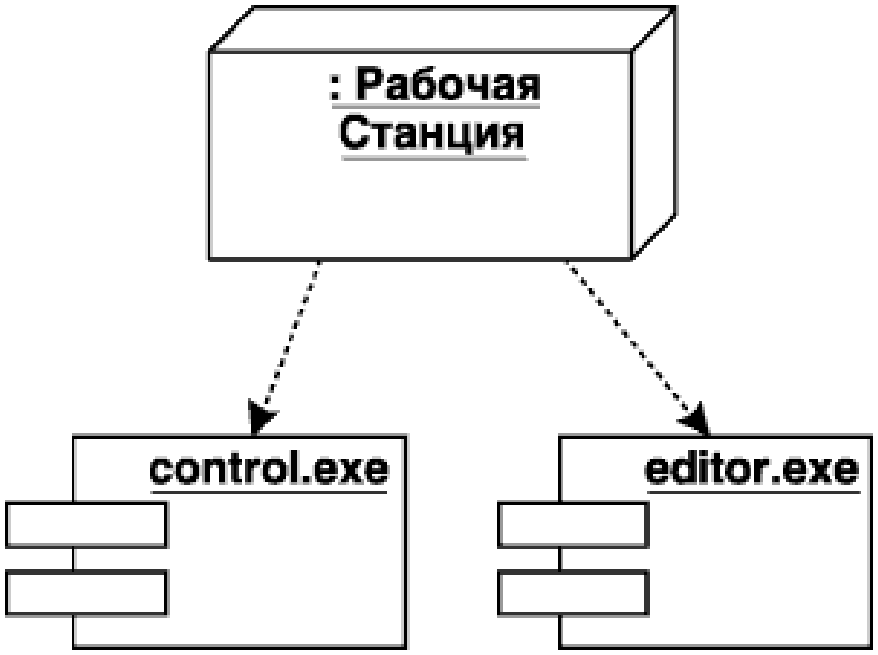
графические стереотипы, которые улучшают наглядность представления *диаграмм развертывания* (в форме рисунка внешнего вида компьютера или сканера).

Отношения: соединения и зависимости

- В качестве отношений выступают физические соединения между *узлами*, а также зависимости между *узлами* и компонентами, которые допускается изображать на *диаграммах развертывания*.
- Соединения являются разновидностью ассоциации и изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между соответствующими *узлами*.

Характер соединения может быть дополнительно специфицирован примечанием, стереотипом, помеченным значением или ограничением.

Зависимости между узлом и размещаемыми на нем компонентами



- альтернатива вложенному изображению компонентов внутри символа *узла*.
- при большом количестве развернутых на *узле* компонентов соответствующую информацию можно представить в форме отношения зависимости

Использование дополнительных стереотипов и изображений

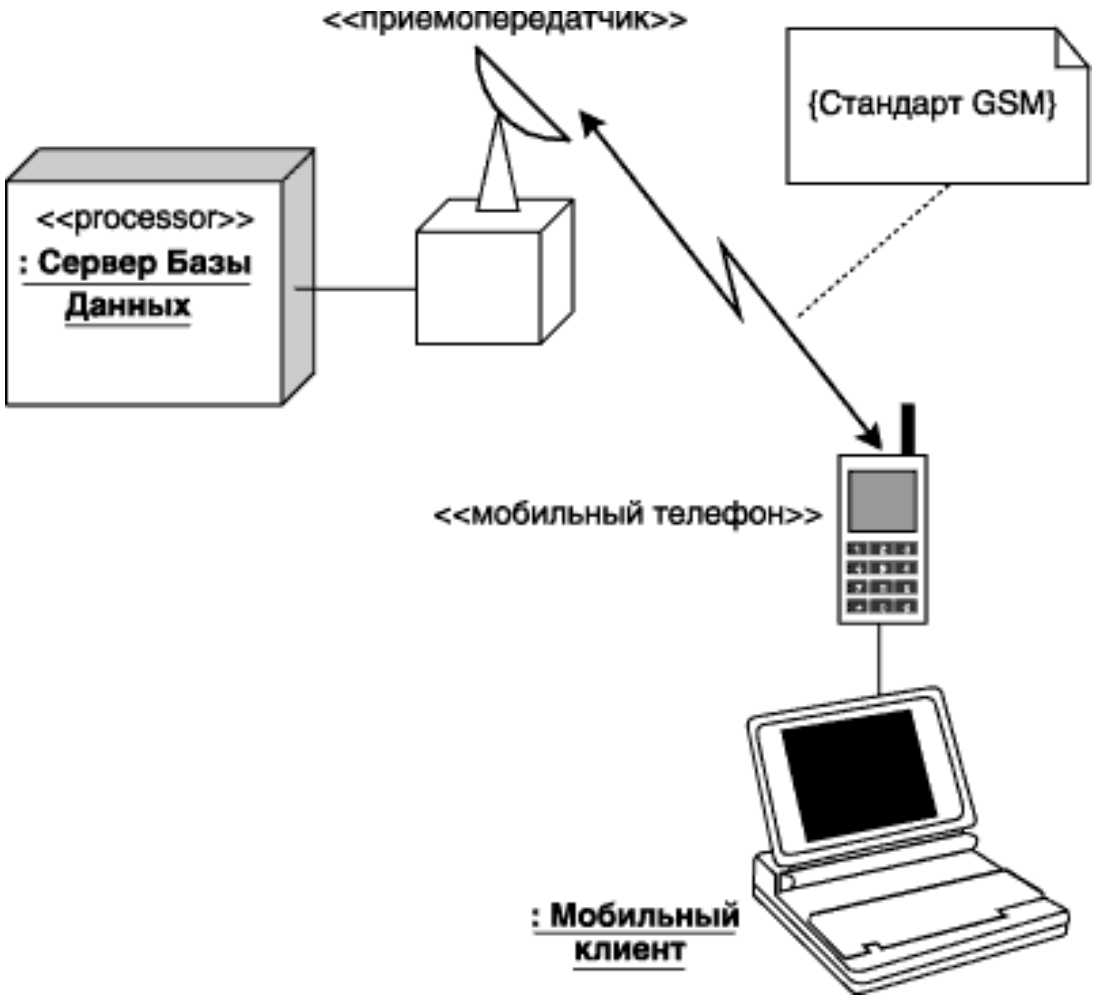


Диаграмма развертывания для системы мобильного доступа к корпоративной базе данных

Рекомендации по построению диаграммы развертывания

- 1. Необходима идентификация всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех функций. Вначале специфицируются вычислительные узлы системы, обладающие процессором и памятью (используются стереотипы, в том числе новые, задаются требования в форме ограничений и помеченных значений).
- 2. Необходимо размещение всех исполняемых компонентов диаграммы по узлам системы без исключения (можно внести в модель дополнительные узлы, содержащие процессор и память).
- 3. Диаграмма развертывания строится для следующих приложений:
  - *реализующих технологию доступа к данным "клиент-сервер;*
  - *имеющих неоднородную распределенную архитектуру (корпоративные интрасети);*
  - *основанных на системе реального времени со встроенными микропроцессорами, которые могут функционировать автономно.*
- 4. Если необходимо включить в модель ресурсы Интернета, то на диаграмме развертывания Интернет часто обозначается в форме "облачка" с соответствующим именем.

Спасибо за внимание

РТУ МИРЭА, кафедра ППИ