Диаграмма компонент

Назначение диаграммы компонентов

Рассмотренные ранее диаграммы относились к логическому уровню представления.

Элементы логического представления не имеют самостоятельного материального воплощения.

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

Для описания таких сущностей используется физическое представление модели.

Назначение диаграммы компонентов

Программная система может считаться реализованной в том случае, когда она будет способна выполнять функции своего целевого предназначения.

Это возможно, если программный код будет реализован в форме исполняемых модулей, библиотек классов и процедур, стандартных графических интерфейсов, файлах баз данных.

Эти компоненты являются элементами физического представления системы.

Назначение диаграммы компонентов

В языке UML для физического представления моделей систем используются *диаграммы реализации* (implementation diagrams): диаграмма компонентов и диаграмма развертывания.

Диаграмма компонентов определяет архитектуру разрабатываемой системы.

Во многих средах разработки модуль или компонент соответствует файлу.

Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними.

Назначение диаграммы компонентов

Диаграмма компонентов разрабатывается для следующих целей:

• Визуализации общей структуры исходного кода программной системы.

• Спецификации исполняемого варианта программной системы.

• Обеспечения многократного использования отдельных фрагментов программного кода.

• Представление физической схемы базы данных.

Компоненты

Для представления физических сущностей в языке UML применяется специальный термин – *компонент* (component).

Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели.

Для графического представления компонента используется специальный символ:

Компоненты

Внутри прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация:

На данном рисунке отображены имя компонента, имя пакета и помеченное значение.

Имя компонента

Имя компонента подчиняется общим правилам именования элементов модели в языке UML.

В качестве простых имен принято использовать имена исполняемых файлов (.exe), имена динамических библиотек (.dll), имена Web-страниц (.html), имена текстовых файлов (.txt, .doc) или файлов справки (.hlp), имена файлов баз данных (DB) или имена файлов с исходными текстами программ (.h, .cpp, .java), скрипты (.pl, .asp) и др.

Имена компонентов будут определяться особенностями синтаксиса соответствующего языка программирования.

Виды компонентов

В языке UML выделяют три вида компонентов:

• **компоненты развертывания**, которые обеспечивают непосредственное выполнение системой своих функций: *динамически подключаемые библиотеки, Web-страницы и файлы справки*;

• **компоненты-рабочие продукты**. Как правило – это файлы с исходными текстами программ.

• **компоненты исполнения**, представляющие исполняемые модули – файлы с расширением exe.

Виды компонентов

Другой способ спецификации различных видов компонентов – явное указание стереотипа.

В языке UML для компонентов определены следующие стереотипы:

• **Библиотека** (library) – определяет первую разновидность компонента, который представляется в форме динамической или статической библиотеки.

• **Таблица** (table) – также определяет первую разновидность компонента, который представляется в форме таблицы базы данных.

Виды компонентов

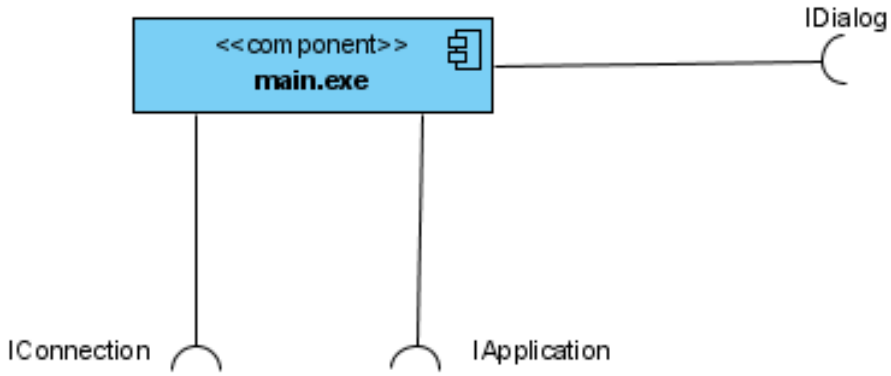
• **Файл** (file) – определяет вторую разновидность компонента, который представляется в виде файлов с исходными текстами программ.

• **Документ** (document) – определяет вторую разновидность компонента, который представляется в форме документа.

• **Исполнимый** (executable) – определяет третий вид компонента, который может исполняться в узле.

Интерфейсы

Имя интерфейса обязательно начинается с заглавной буквы «I» и записывается рядом с интерфейсом.



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

Интерфейсы

Другой способ представления интерфейса – изображение его в виде прямоугольника класса со стереотипом «интерфейс» и возможными секциями атрибутов и операций:

Этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

Зависимости

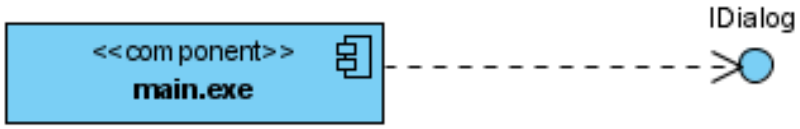
Отношение зависимости служит для представления связи, когда изменение одного элемента модели оказывает влияние или приводит к изменению другого элемента модели.

Отношение зависимости на диаграмме компонентов изображается пунктирной линией со стрелкой, направленной от клиента (зависимого элемента) к источнику (независимому элементу).

Зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

Зависимости

Пример зависимости компонента клиента от импортируемого интерфейса:

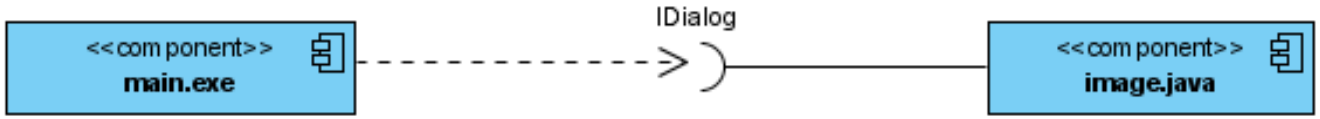


Наличие стрелки означает, что компонент не реализует соответствующий интерфейс, а использует его в процессе своего выполнения.

Зависимости

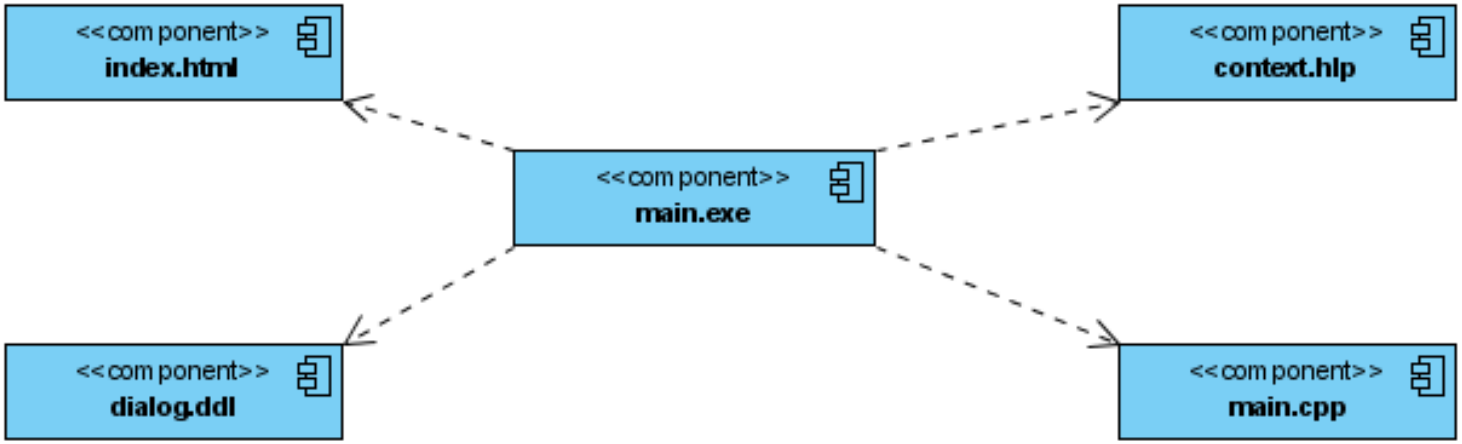
Одновременно на диаграмме может присутствовать другой компонент, который реализует этот интерфейс.

Пример: Компонент «main.exe» зависит от импортируемого интерфейса IDialog, который, в свою очередь, реализуется компонентом с именем «image.java»:



Зависимости

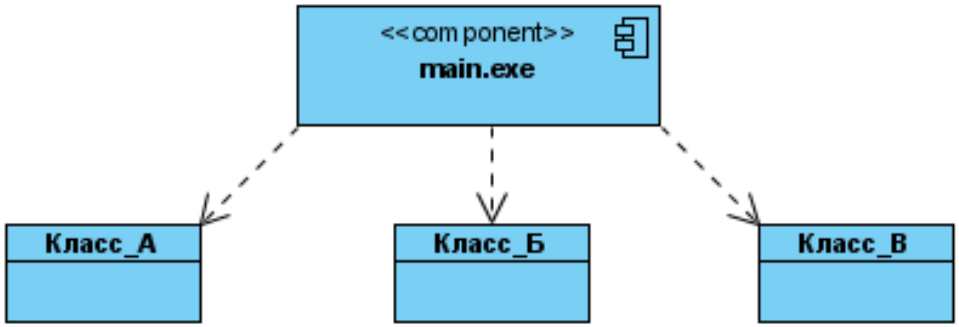
Другой случай: отношения зависимости между различными видами компонентов:



Внесение изменений в исходные тексты программ или динамические библиотеки приводят к изменениям самого компонента.

Зависимости

На диаграмме компонентов могут быть представлены отношения зависимости между компонентами и реализованными в них классами:



В данном случае из диаграммы компонентов не следует, что классы реализованы этим компонентом.

Рекомендации по построению диаграммы компонентов

• До начала разработки необходимо принять решение о выборе вычислительных платформ и операционных систем, о выборе баз данных и языков программирования.

• Необходимо решить, из каких физических частей (файлов) будет состоять программная система.

• После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных.

• Завершающий этап построения диаграммы компонентов связан с установлением взаимосвязей между компонентами и отношений реализации.

Диаграмма развертывания

Назначение диаграммы развертывания

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована.

Диаграмма развертывания (синоним – диаграмма размещения) применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы.

Назначение диаграммы развертывания

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime).

Представляются только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками.

Эта диаграмма завершает процесс ООАП для конкретной программной системы.

Назначение диаграммы развертывания

Цели, преследуемые при разработке диаграммы развертывания:

• Определить распределение компонентов системы по ее физическим узлам.

• Показать физические связи между всеми узлами реализации системы на этапе ее исполнения.

• Выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Узел

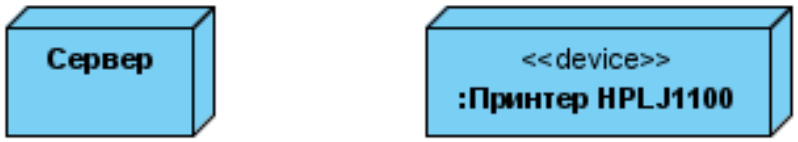
Узел (node) – физически существующий элемент системы, обладающий некоторым вычислительным ресурсом.

Вычислительный ресурс должен иметь по меньшей мере некоторый объем электронной или магнитооптической памяти и/или процессор.

В последней версии языка UML добавляются другие устройства: датчики, принтеры, модемы, цифровые камеры, сканеры и др.

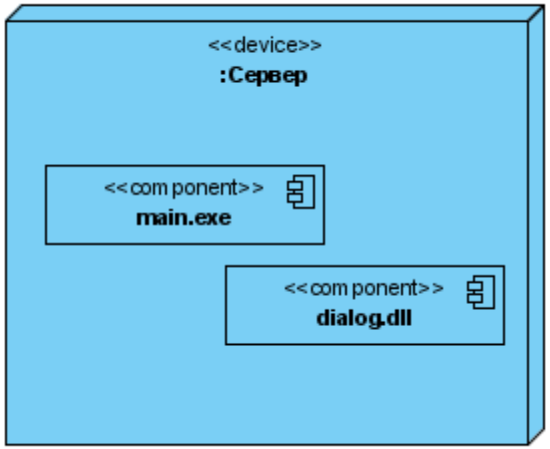
Узел

Графически на диаграмме развертывания узел изображается в форме трехмерного куба. Узел имеет собственное имя, которое указывается внутри данного куба:



Узел

Если необходимо явно указать компоненты, которые размещаются на отдельном узле, то допустимо использовать вложенные изображения компонентов:



Соединения

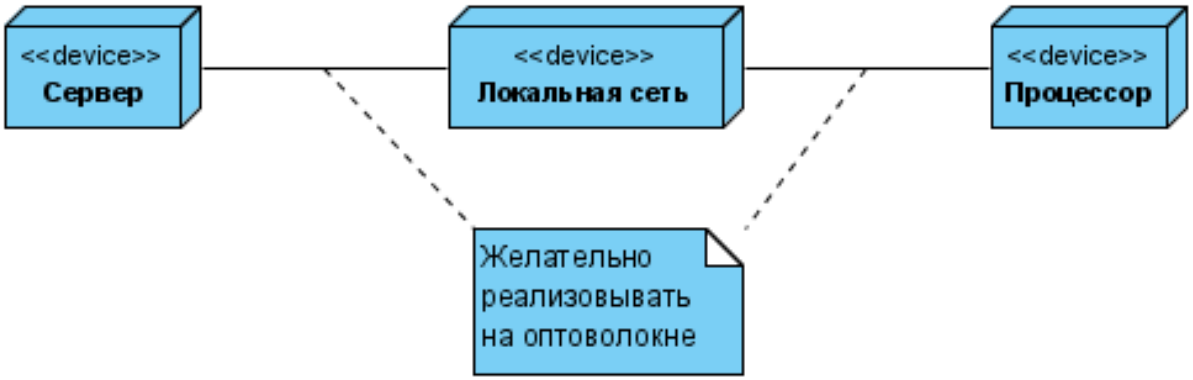
Кроме узлов на диаграмме развертывания указываются отношения между ними.

В качестве отношений выступают физические соединения между узлами и зависимости между узлами и компонентами.

Соединения изображаются отрезками линий без стрелок. Наличие такой линии указывает на необходимость организации физического канала для обмена информацией между узлами.

Соединения

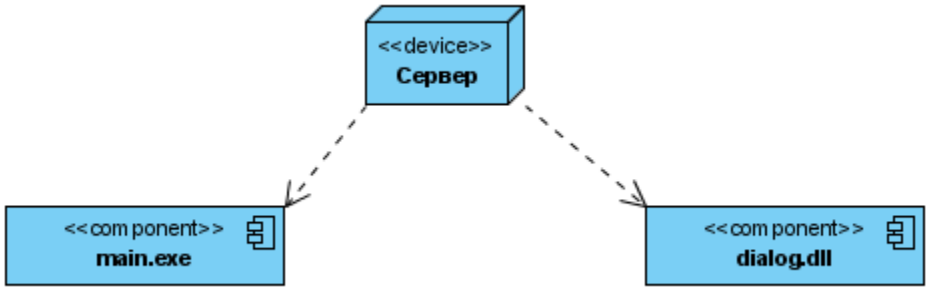
Характер соединения может быть дополнительно специфицирован примечанием:



Соединения

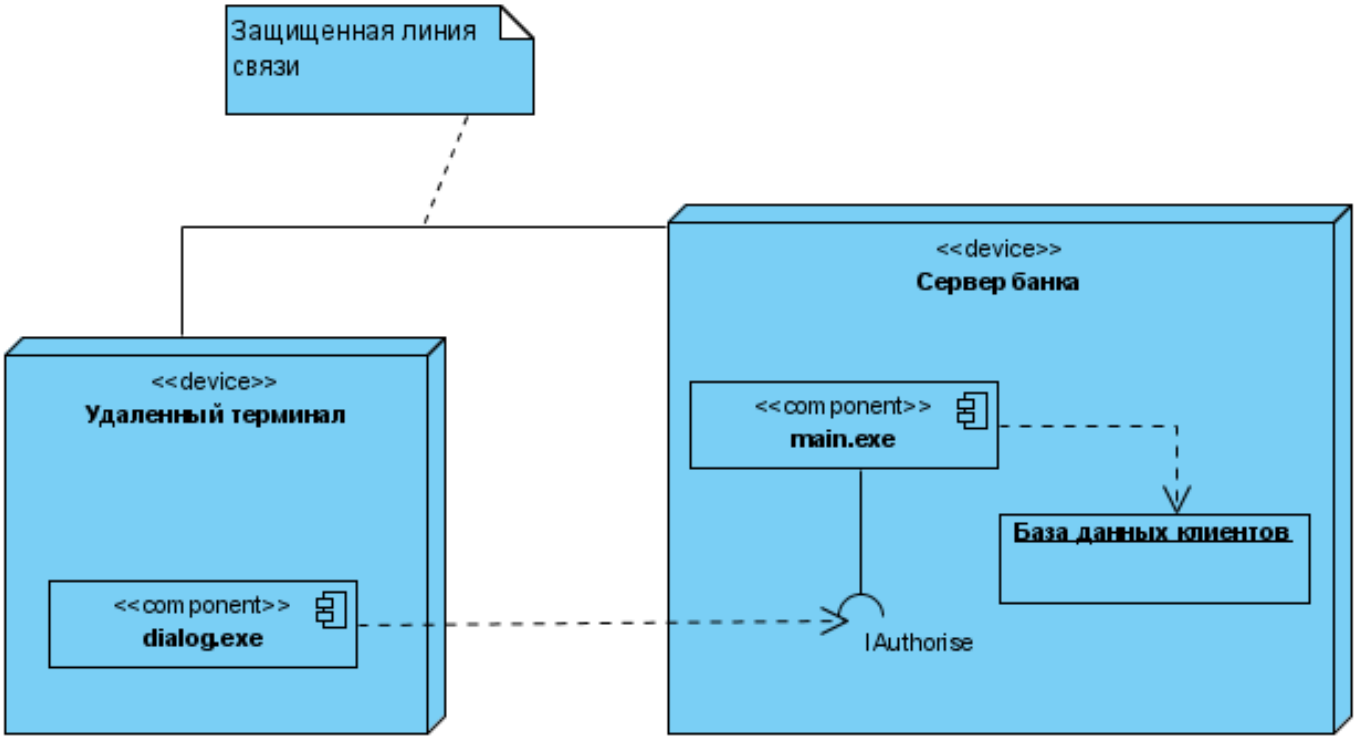
Кроме соединений на диаграмме развертывания могут присутствовать отношения зависимости между узлом и развернутыми на нем компонентами.

Вложенные изображения компонентов не всегда удобны и такую информацию можно представить в форме отношения зависимости:



Соединения

Более сложная диаграмма развертывания:



Рекомендации по построению диаграммы развертывания

• Разработка диаграммы развертывания начинается с идентификации всех аппаратных, механических и тругих типов устройств.

• Далее необходимо разместить все исполняемые компоненты по узлам системы.

• При разработке простых программ необходимость в диаграмме развертывания может вообще отсутствовать.

Рекомендации по построению диаграммы развертывания

Есть смысл строить диаграммы развертывания в следующих случаях:

• Моделирование программных систем, реализующих технологию доступа к данным «клиент-сервер».

• Моделирование неоднородных распределенных архитектур (корпоративные интрасети).

• Системы со встроенными микропроцессорами, которые могут функционировать автономно.