«Диаграммы компонентов и развертывания»

Все рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления.

Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения.

Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически.

Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы.

Однако для создания конкретной физической системы необходимо некоторым образом реализовать все элементы логического представления в конкретные материальные сущности.

Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Чтобы пояснить отличие логического и физического представлений, рассмотрим в общих чертах процесс разработки некоторой программной системы.

Ее исходным логическим представлением могут служить структурные схемы алгоритмов и процедур, описания интерфейсов и концептуальные схемы баз данных.

Однако для реализации этой системы необходимо разработать исходный текст программы на некотором языке программирования.

При этом уже в тексте программы предполагается такая организация программного кода, которая предполагает его разбиение на отдельные модули.

Тем не менее, исходные тексты программы еще не являются окончательной реализацией проекта, хотя и служат фрагментом его физического представления.

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

А это возможно, только если программный код системы будет реализован в форме исполняемых модулей, библиотек классов и процедур, стандартных графических интерфейсов, файлах баз данных. Именно эти компоненты являются необходимыми элементами физического представления системы.

Таким образом, полный проект программной системы представляет собой совокупность моделей логического и физического представлений, которые должны быть согласованы между собой.

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

Диаграмма компонентов, в отличие от ранее рассмотренных диаграмм, 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.

Диаграмма компонентов разрабатывается для следующих целей:

- Визуализации общей структуры исходного кода программной системы.
- Спецификации исполнимого варианта программной системы.
- Обеспечения многократного использования отдельных фрагментов программного кода.
- Представления концептуальной и физической схем баз данных.

В разработке диаграмм компонентов участвуют как системные аналитики и архитекторы, так и программисты.

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.

Диаграмма компонентов отражает общие зависимости между компонентами, рассматривая последние в качестве классификаторов.

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 может использоваться специальный символ — прямоугольник со вставленными слева двумя более мелкими прямоугольниками (рис. 1).

Внутри объемлющего прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация.

Изображение этого символа может незначительно варьироваться в зависимости от характера ассоциируемой с компонентом информации.

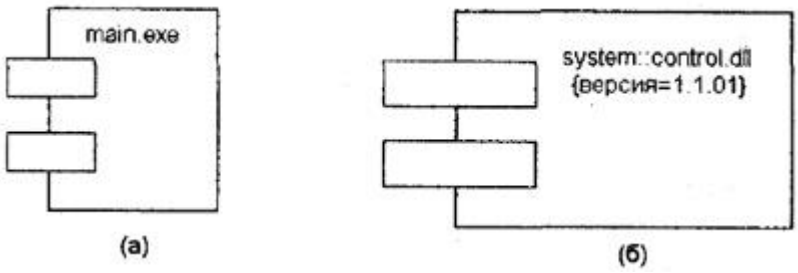

Рис. 1. Графическое изображение компонента в языке UML

Так, в первом случае (рис. 1, а) с компонентом уровня экземпляра связывается только его имя, а во втором (рис. 1, б) — дополнительно имя пакета и помеченное значение.

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

Так, верхний маленький прямоугольник концептуально ассоциируется с данными, которые реализует этот компонент (ранее он изображался в форме овала).

Нижний маленький прямоугольник ассоциируется с операциями или методами, реализуемыми компонентом.

В простых случаях имена данных и методов записывались явно в этих 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 некоторых знаков препинания.

Отдельный компонент может быть представлен на уровне типа или на уровне экземпляра.

Хотя его графическое изображение в обоих случаях одинаковое, правила записи имени компонента несколько отличаются. Если компонент представляется на уровне типа, то в качестве его имени записывается только имя типа с заглавной буквы.

Если же компонент представляется на уровне экземпляра, то в качестве его имени записывается <имя компонента ':' имя типаX При этом вся строка имени подчеркивается.

В качестве простых имен принято использовать имена исполняемых файлов (с указанием расширения exe после точки-разделителя), имена динамических библиотек (расширение dll), имена Web-страниц (расширение html), имена текстовых файлов (расширения txt или doc) или файлов справки (hip), имена файлов баз данных (DB) или имена файлов с исходными

текстами программ (расширения h, cpp для языка C++, расширение Java для языка Java), скрипты (pi, asp) и др.

Поскольку конкретная реализация логического представления модели системы зависит от используемого программного инструментария, то и имена компонентов будут определяться особенностями синтаксиса соответствующего языка программирования.

Виды компонентов

Поскольку компонент как элемент физической реализации модели представляет отдельный модуль кода, иногда его комментируют с указанием дополнительных графических символов, иллюстрирующих конкретные особенности его реализации.

Строго говоря, эти дополнительные обозначения для примечаний не специфицированы в языке UML.

Однако их применение упрощает понимание диаграммы компонентов, существенно повышая наглядность физического представления. Некоторые из таких общепринятых обозначений для компонентов изображены ниже (рис. 10.2).

В языке UML выделяют три вида компонентов.

- Во-первых, компоненты развертывания, которые обеспечивают непосредственное выполнение системой своих функций. Такими компонентами могут быть динамически подключаемые библиотеки с расширением dll (рис. 2, а), Web-страницы на языке разметки гипертекста с расширением html (рис. 2, б) и файлы справки с расширением hlp (рис. 2, в).
- Во-вторых, компоненты-рабочие продукты. Как правило — это файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++ (рис. 2, г).
- В-третьих, компоненты исполнения, представляющие исполнимые модули — файлы с расширением exe. Они обозначаются обычным образом.

Эти элементы иногда называют артефактами, подчеркивая при этом их законченное информационное содержание, зависящее от конкретной технологии реализации соответствующих компонентов. Более того, разработчики могут для этой цели использовать самостоятельные обозначения, поскольку в языке UML нет строгой нотации для графического представления примечаний.

Другой способ спецификации различных видов компонентов — явное указание стереотипа компонента перед его именем.

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.

- 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

Следующим элементом диаграммы компонентов являются интерфейсы.

Последние уже неоднократно рассматривались ранее, поэтому здесь будут отмечены те их, особенности, которые характерны для представления на диаграммах компонентов.

Напомним, что в общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис. 3, а).

При этом имя интерфейса, которое обязательно должно начинаться с заглавной буквы "I", записывается рядом с окружностью.

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

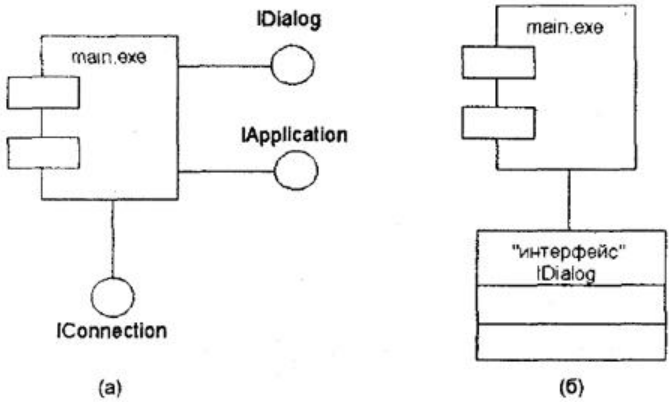

Рис. 3. Графическое изображение интерфейсов на диаграмме компонентов

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом "интерфейс" и возможными секциями атрибутов и операций (рис. 3, б).

Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

При разработке программных систем интерфейсы обеспечивают не только совместимость различных версий, но и возможность вносить существенные изменения в одни части программы, не изменяя другие ее части.

Таким образом, назначение интерфейсов существенно шире, чем спецификация взаимодействия с пользователями системы (актерами).

Характер использования интерфейсов отдельными компонентами может отличаться. Поэтому различают два способа связи интерфейса и компонента. Если компонент реализует некоторый интерфейс, то такой интерфейс называют экспортируемым, 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

В общем случае отношение зависимости также было рассмотрено ранее.

Напомним, что зависимость не является ассоциацией, а служит для представления только факта наличия такой 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 отражать связи модулей программы на этапе компиляции и генерации объектного кода.

В другом случае зависимость может отражать наличие в независимом компоненте описаний классов, которые используются в зависимом компоненте для создания соответствующих объектов.

Применительно к диаграмме компонентов зависимости могут связывать компоненты и импортируемые этим компонентом интерфейсы, а также различные виды компонентов между собой.

В первом случае рисуют стрелку от компонента-клиента к импортируемому интерфейсу (рис. 4).

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

Причем на этой же диаграмме может присутствовать и другой компонент, который реализует этот интерфейс.

Так, например, изображенный ниже фрагмент диаграммы компонентов представляет информацию о том, что компонент с именем "main.exe" зависит от импортируемого интерфейса I Dialog, который, в свою очередь, реализуется компонентом с именем "image.java".

Для второго компонента этот же интерфейс является экспортируемым.

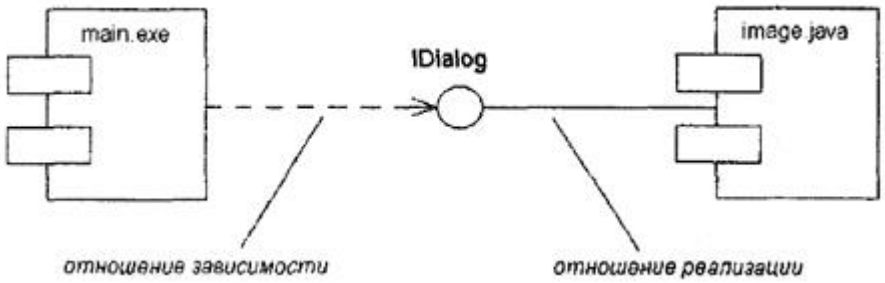

Рис. 4. Фрагмент диаграммы компонентов с отношением зависимости

Заметим, что изобразить второй компонент с именем "image.java" в форме варианта примечания нельзя именно в силу того факта, что этот компонент реализует интерфейс.

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

Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы. Разумеется, изменения в структуре описаний классов могут привести к изменению компонента.

Ниже приводится фрагмент зависимости подобного рода, когда некоторый компонент зависит от соответствующих классов.

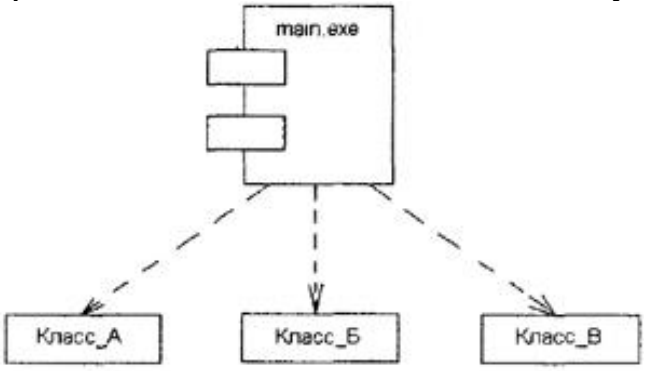

Рис. 6. Графическое изображение зависимости между компонентом и классами

Рекомендации по построению диаграммы компонентов

Разработка диаграммы компонентов предполагает использование информации как о логическом представлении модели системы, так и об особенностях ее физической реализации. До начала разработки необходимо принять решения о выборе вычислительных платформ и операционных систем, на которых предполагается реализовывать систему, а также о выборе конкретных баз данных и языков программирования.

После этого можно приступать к общей структуризации диаграммы компонентов. В первую очередь, необходимо решить, из каких физических частей (файлов) будет состоять программная система. На этом этапе следует обратить внимание на такую реализацию системы, которая обеспечивала бы не только возможность повторного использования кода за счет рациональной

декомпозиции компонентов, но и создание объектов только при их необходимости.

Речь идет о том, что общая производительность программной системы существенно зависит от рационального использования ею вычислительных ресурсов. Для этой цели необходимо 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.

После общей структуризации физического представления системы необходимо дополнить модель интерфейсами и схемами базы данных. При разработке интерфейсов следует обращать внимание на согласование (стыковку) различных частей программной системы. Включение в модель схемы базы данных предполагает спецификацию отдельных таблиц и установление информационных связей между таблицами.

Наконец, завершающий этап построения диаграммы компонентов связан с установлением и нанесением на диаграмму взаимосвязей между компонентами, а также отношений 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 виды графического изображения компонентов.

При разработке диаграммы компонентов следует придерживаться общих принципов создания моделей на языке UML. В частности, в первую очередь необходимо использовать уже имеющиеся в языке UML компоненты и стереотипы. Для большинства типовых проектов этого набора элементов может оказаться достаточно для представления компонентов и зависимостей между ними.

Если же проект содержит некоторые физические элементы, описание которых отсутствует в языке UML, то следует воспользоваться механизмом расширения. В частности, использовать дополнительные стереотипы для отдельных нетиповых компонентов или помеченные значения для уточнения их отдельных характеристик.

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

Конечно, если разрабатывается простая программа, которая может выполняться локально на компьютере пользователя, не задействуя никаких периферийных устройств и ресурсов, то в этом случае нет необходимости в разработке дополнительных диаграмм.

Однако при разработке корпоративных приложений ситуация представляется совсем по-другому.

Во-первых, сложные программные системы могут реализовываться в сетевом варианте на различных вычислительных платформах и технологиях доступа к распределенным базам данных.

Наличие локальной корпоративной сети требует решения целого комплекса дополнительных задач по рациональному размещению компонентов по узлам этой сети, что определяет общую производительность программной системы.

Во-вторых, интеграция программной системы с Интернетом определяет необходимость решения дополнительных вопросов при проектировании системы, таких как обеспечение безопасности, криптозащищенности и устойчивости доступа к информации для корпоративных клиентов.

Эти аспекты в немалой степени зависят от реализации проекта в форме физически существующих узлов системы, таких как серверы, рабочие станции, брандмауэры, каналы связи и хранилища данных.

Наконец, технологии доступа и манипулирования данными в рамках общей схемы "клиент-сервер" также требуют размещения больших баз данных в различных сегментах корпоративной сети, их резервного копирования, архивирования, кэширования для обеспечения необходимой производительности системы в целом.

Эти аспекты также требуют визуального представления с целью спецификации программных и технологических особенностей реализации распределенных архитектур.

Как было отмечено, первой из диаграмм физического представления является диаграмма компонентов.

Второй формой физического представления программной системы является диаграмма развертывания (синоним — диаграмма размещения).

Она применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение 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 лишь на этапе ее исполнения (runtime).

При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками.

Те компоненты, которые не используются 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 единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

Эта диаграмма, по сути, завершает процесс ООАП для конкретной программной системы и ее разработка, как правило, является последним этапом спецификации модели.

Итак, перечислим 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.

Для обеспечения этих требований диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками.

Далее рассмотрим отдельные элементы, из которых состоят диаграммы развертывания.

Узел

Узел (node) представляет собой некоторый физически существующий элемент системы, обладающий некоторым вычислительным ресурсом.

В качестве вычислительного ресурса узла может рассматриваться наличие по меньшей мере некоторого объема электронной или магнитооптической памяти и/или процессора.

В последней версии языка UML понятие узла расширено и может включать в себя не только вычислительные устройства (процессоры), но и другие механические или электронные устройства, такие как датчики, принтеры, модемы, цифровые камеры, сканеры и манипуляторы.

Графически на диаграмме развертывания узел изображается в форме трехмерного куба (строго говоря, псевдотрехмерного прямоугольного параллелепипеда).

Узел имеет собственное имя, которое указывается внутри этого графического символа.

Сами узлы могут представляться как в качестве типов (рис. 9, а), так и в качестве экземпляров (рис. 9, б).

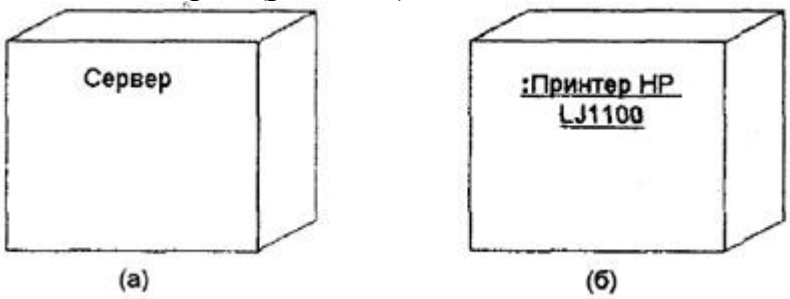

Рис. 9. Графическое изображение узла на диаграмме развертывания

В первом случае имя узла записывается без подчеркивания и начинается с заглавной буквы.

Во втором имя узла-экземпляра записывается в виде <имя узла ':' имя типа узла>.

Имя типа узла указывает на некоторую разновидность узлов, присутствующих в модели системы.

Например, аппаратная часть системы может состоять из нескольких персональных компьютеров, каждый из которых соответствует отдельному узлу-экземпляру в модели.

Однако все эти узлы-экземпляры относятся к одному типу узлов, а именно узлу с именем типа "Персональный компьютер".

Так, на представленном выше рисунке (рис. 9, а) узел с именем "Сервер" относится к общему типу и никак не конкретизируется.

Второй же узел (рис. 9, б) является анонимным узлом-экземпляром конкретной модели принтера.

Так же, как и на диаграмме компонентов, изображения узлов могут расширяться, чтобы включить некоторую дополнительную информацию о спецификации узла.

Если дополнительная информация относится к имени узла, то она записывается под этим именем в форме помеченного значения (рис. 10).

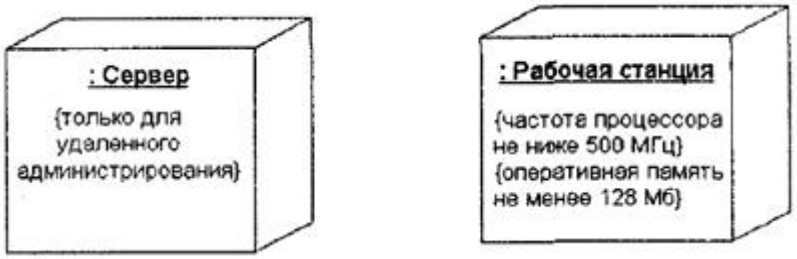

Рис. 10. Графическое изображение узла-экземпляра с дополнительной информацией в форме помеченного значения

Если необходимо явно указать компоненты, которые размещаются на отдельном узле, то это можно сделать двумя способами.

Первый из них позволяет разделить графический символ узла на две секции горизонтальной линией.

В верхней секции записывают имя узла, а в нижней секции — размещенные на этом узле компоненты (рис. 11, а).

Второй способ разрешает показывать на диаграмме развертывания узлы с вложенными изображениями компонентов (рис. 11, б).

Важно помнить, что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты.

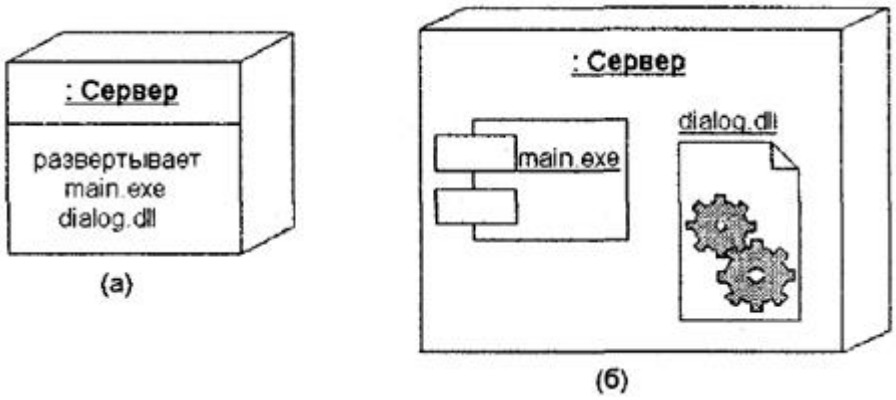

Рис. 11. Варианты графического изображения узлов-экземпляров с размещаемыми на них компонентами

В качестве дополнения к имени узла могут использоваться различные стереотипы, которые явно специфицируют назначение этого узла.

Хотя в языке UML стереотипы для узлов не определены, в литературе встречаются следующие их варианты: "процессор", "датчик", "модем", "сеть", "консоль" и др., которые самостоятельно могут быть определены разработчиком.

Более того, на диаграммах развертывания допускаются специальные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции.

Говоря о дополнительных графических изображениях для узлов диаграммы развертывания, прежде всего имеют в виду наглядность их представления.

Например, процессор можно изобразить как в виде общего узла (рис. 9), так и в форме изображения внешнего вида компьютера. Соответственно, консоль может быть изображена в виде клавиатуры. В любом из этих случаев разработчик должен обладать, в дополнение к основным, еще и художественными способностями.

Соединения

Кроме собственно изображений узлов на диаграмме развертывания указываются отношения между ними.

В качестве отношений выступают физические соединения между узлами и зависимости между узлами и компонентами, изображения которых тоже могут присутствовать 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, помеченным значением или ограничением.

Так, на представленном ниже фрагменте диаграммы развертывания (рис. 12) явно определены не только требования к скорости передачи данных в локальной сети с помощью помеченного значения, но и рекомендации по технологии физической реализации соединений в форме примечания.

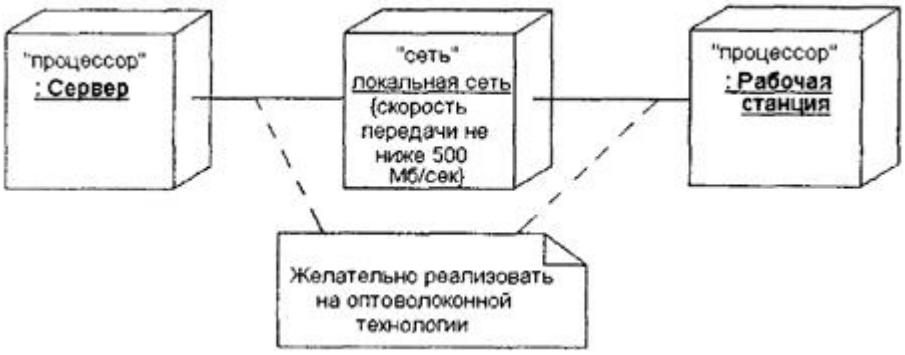

Рис. 12. Фрагмент диаграммы развертывания с соединениями меходу узлами

Кроме соединений на диаграмме развертывания могут присутствовать отношения зависимости между узлом и развернутыми на нем компонентами. Подобный способ является альтернативой вложенному изображению компонентов внутри символа узла, что не всегда удобно, поскольку делает этот символ излишне объемным.

Поэтому при большом количестве развернутых на узле компонентов соответствующую информацию можно представить в форме отношения зависимости (рис. 13).

Диаграммы развертывания могут иметь более сложную структуру, включающую вложенные компоненты, интерфейсы и другие аппаратные устройства.

На изображенной ниже диаграмме развертывания (рис. 14) представлен фрагмент физического представления системы удаленного обслуживания клиентов банка.

Узлами этой системы являются удаленный терминал (узел-тип) и сервер банка (узел-экземпляр).

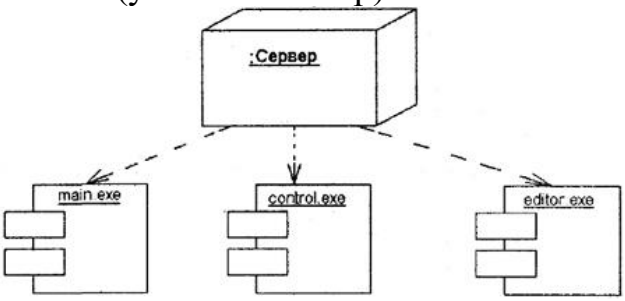

Рис. 13. Диаграмма развертывания с отношением зависимости между узлом и развернутыми на нем компонентами

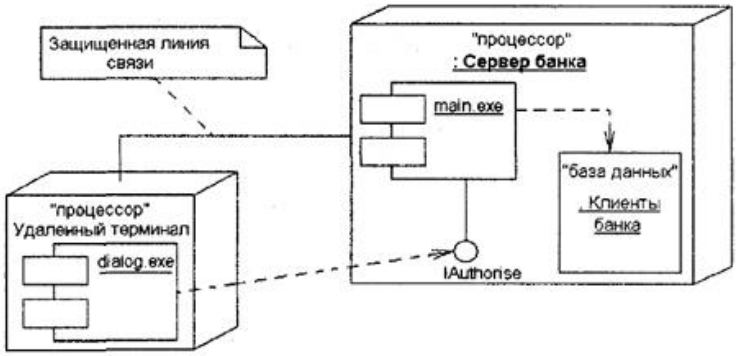

Рис. 14. Диаграмма развертывания для системы удаленного обслуживания клиентов банка

На этой диаграмме развертывания указана зависимость компонента реализации диалога "dialog.exe" на удаленном терминале от интерфейса IAuthorise, реализованного компонентом "main.exe", который, в свою очередь, развернут на анонимном узле-экземпляре "Сервер банка".

Последний зависит от компонента базы данных "Клиенты банка", который развернут на этом же узле.

Примечание указывает на необходимость использования защищенной линии связи для обмена данными в данной системе.

Другой вариант записи этой информации заключается в дополнении диаграммы узлом со стереотипом "закрытая сеть".