Лекция 4. Структурные схемы.

Рассмотрим структурные диаграммы.

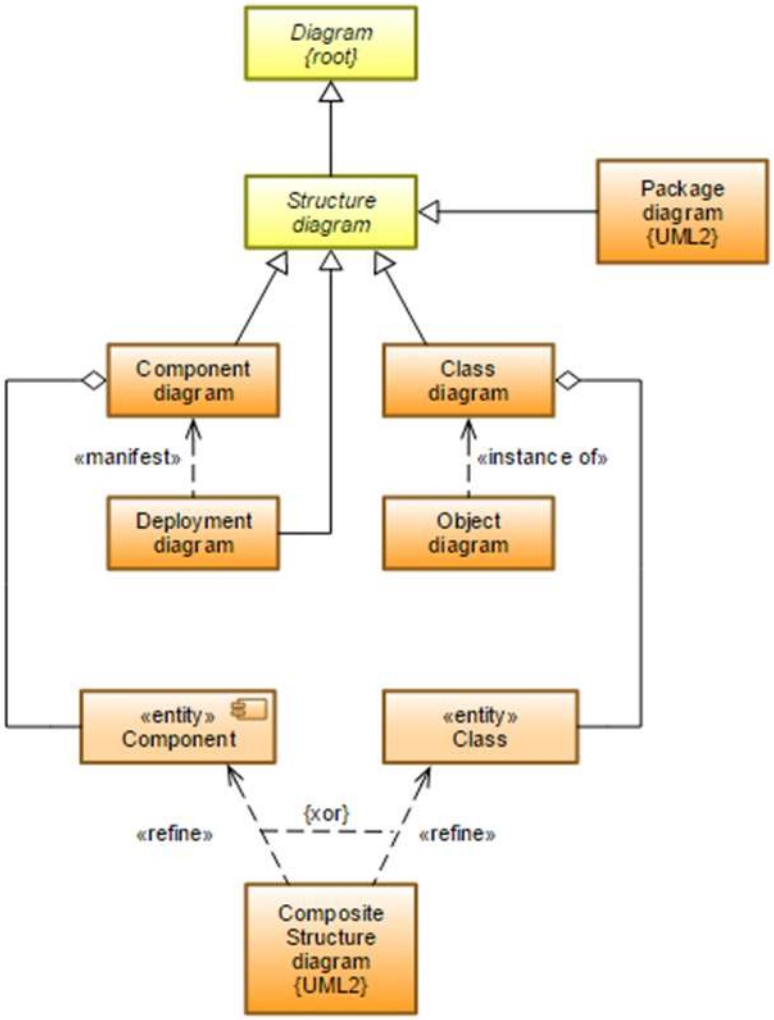

Рис. Иерархия типов диаграмм для UML 2 (часть 1)

Диаграммы классов(class diagram)

Диаграммы классов при моделировании объектно-ориентированных систем встречаются чаще других. На таких диаграммах отображается множество классов, интерфейсов, кооперaций и отношений между ними. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Кроме того, диаграммы классов составляют основу еще двух диаграмм – компонентов и развертывания.

Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы.

Компоненты диаграммы классов

Классом называется описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Например, класс “Стена” описывает объекты с общими свойствами: высотой, длиной, толщиной, и т.д. При этом конкретные стены будут рассматриваться как отдельные экземпляры класса «стена». У каждого класса есть имя, (простое или составное, к которому спереди добавлено имя пакета, в который входит класс). Имя класса в пакете должно быть уникальным. Класс реализует один или несколько интерфейсов.

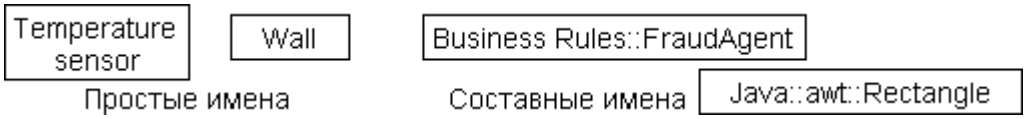

Рис. 40 Простые и составные имена

Атрибут – это именованное свойство класса, включающее описание множества значений, которые могут принимать экземпляры этого класса. Класс может иметь любое число атрибутов или не иметь их вовсе. В языках высокого уровня, таких, как C++, Java, атрибуты соответствуют переменным, объявленным в классе. Например, у любой стены есть высота, ширина и толщина. Атрибуты представлены в разделе, расположенном под именем класса; при этом указываются их имена, и иногда начальное значение (рис. 41).

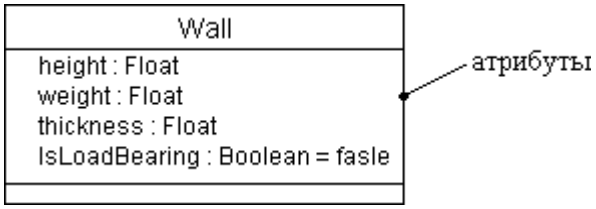

Рис. 41 Атрибуты и их класс

Операция – это некоторый сервис, который предоставляет экземпляр или объект класса по требованию своих клиентов (других объектов, в том числе и экземпляров данного класса). Класс может содержать любое число операций или не содержать их вовсе. В языках высокого уровня, таких, как C++, Java, операции соответствуют функциям, объявленным в классе. Операцию можно описать более подробно, указав имена и типы параметров, их значения, принятые по умолчанию, а также тип возвращаемого значения (рис. 42).

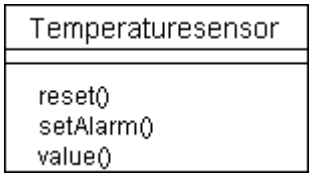

Рис. 42 Операции

При изображении класса необязательно сразу показывать все его атрибуты и операции. Их может быть много, однако для данного представления системы лишь небольшое подмножество атрибутов и операций имеет значение. В это случае класс сворачивают, и изображают

только некоторые из атрибутов и операций, а дополнительные атрибуты или операции обозначают многоточием. Для лучшей организации списков атрибутов и операций можно снабдить каждую группу дополнительным описанием (стереотипами).

Обязанности класса – это своего рода контракт, которому он должен подчиняться. Атрибуты и операции являются свойствами, посредством которых выполняются обязанности класса. Например, класс FraudAgent (агент по предотвращению мошенничества), который встречается в приложениях по обработке кредитных карточек, отвечает за оценку платежных требований – законные, подозрительные или подложные (рис. 43). Моделирование классов лучше всего начинать с определения обязанностей сущностей, которые входят в словарь системы. Число обязанностей класса может быть произвольным, но на практике хорошо структурированный класс имеет по меньшей мере одну обязанность; с другой стороны, их не должно быть и слишком много. При уточнении модели обязанности класса преобразуются в совокупность атрибутов и операций, которые должны наилучшим образом обеспечить их выполнение.

Рис. 43 Обязанности

Классы редко существуют автономно, они взаимодействуют между собой. Это значит, что, при моделировании системы, необходимо идентифицировать не только сущности, составляющие ее словарь, но и описать, как они соотносятся друг с другом. Существует основные три вида отношений между классами: *зависимости, обобщения, ассоциации* (рис. 44).

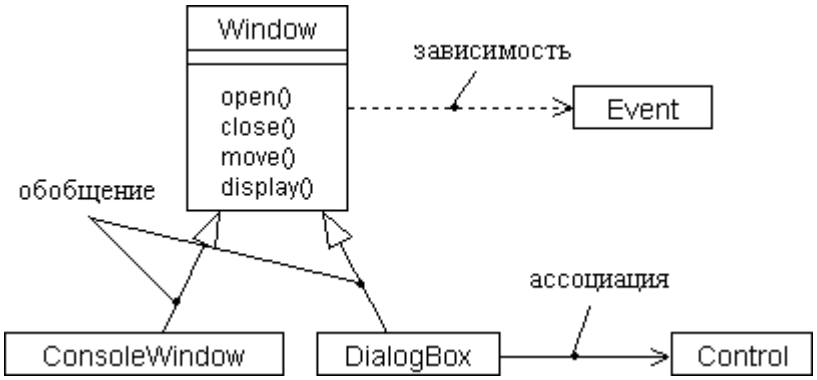

Рис. 44 Отношения

Кроме перечисленных отношений на диаграммах классов также применяются отношения агрегации и композиции.

Отношение агрегации применяется для представления системных взаимосвязей типа "часть-целое". Например, деление персонального компьютера на составные части: системный блок, монитор, клавиатуру и мышь (рис. 45).

Рис. 45 Отношения агрегации

Отношение композиции является частным случаем агрегации. Оно служит для выделения специальной формы отношения "часть-целое", при которой составляющие части в некотором смысле находятся внутри целого. Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т.е. с уничтожением целого уничтожаются и все его составные части. Например – окно интерфейса программы, состоящее из строки заголовка, кнопок управления размером, полос прокрутки, главного меню, рабочей области и строки состояния (рис. 46).

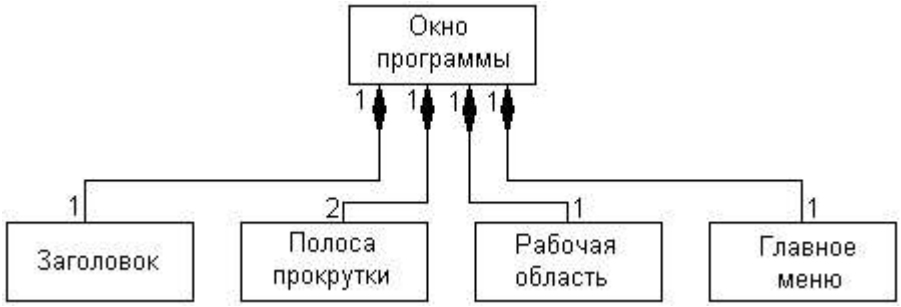

Рис. 46 Отношения композиции

Кроме описанных выше элементов, на диаграмме классов также могут присутствовать примечания, дополнения, стереотипы, помеченные значения, ограничения.

Стереотипом называют расширение словаря UML, позволяющее создавать новые виды строительных блоков, аналогичные существующим, но специфичные для данной задачи. Стереотип представлен в виде имени, заключенного в кавычки и расположенного над именем другого элемента. С помощью стереотипа создаётся новый строительный блок, который напоминает существующий, но обладает новыми свойствами (рис. 47).

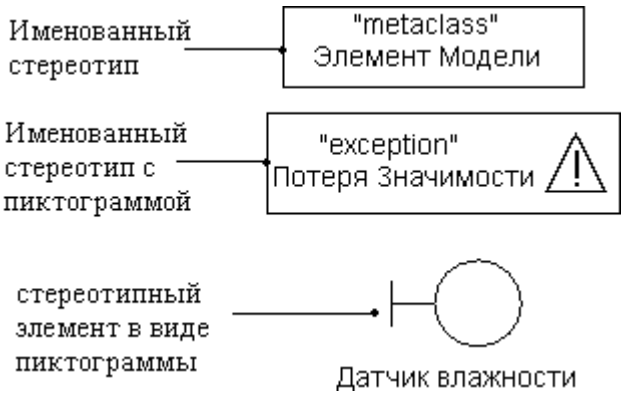

Рис. 47 Стереотипы

Помеченное значение позволяет включать новую информацию в спецификацию элемента. Например, при создании нескольких версий ПО необходимо отслеживать версию и автора какой-нибудь важной абстракции. Ни версия, ни автор не являются первичными концепциями UML, но их можно добавить к любому блоку, например, к классу, задавая для него новые помеченные значения (рис. 48).

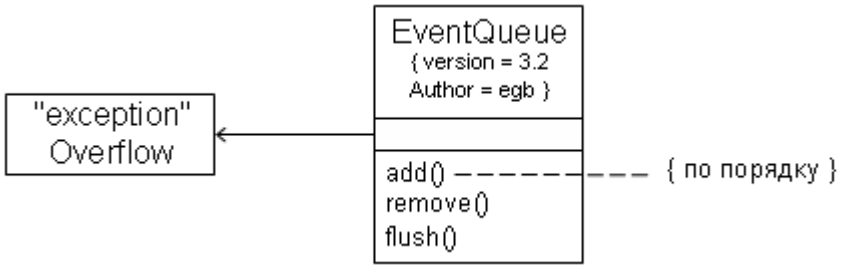

Рис. 48 Помеченное значение

Ограничение – это расширение семантики элемента UML, позволяющее создавать новые или изменять существующие правила (рис. 49).

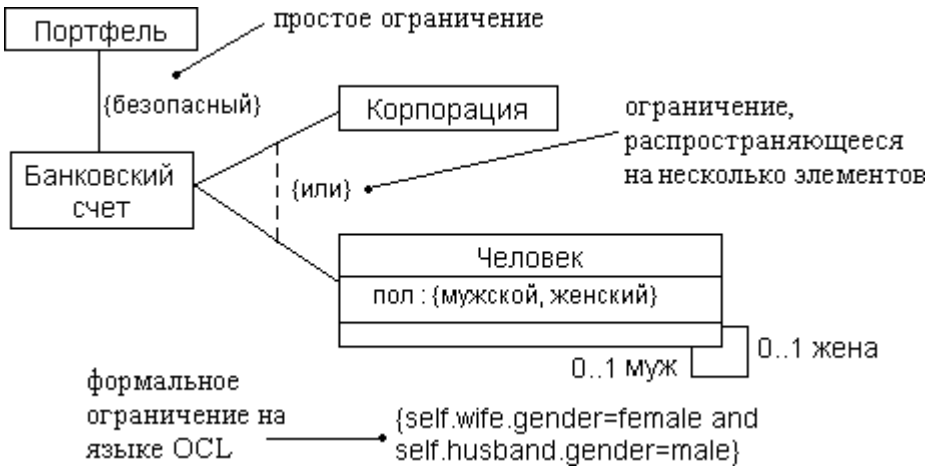

Рис. 49 Ограничения

Каждый элемент языка UML имеет свою семантику. С помощью ограничений можно создавать новую семантику или изменять существующие семантические правила. Например, на рис. 49 показано, что информация, передаваемая между классами “Портфель” и “Банковский счёт”, зашифрована. При моделировании систем реального времени часто

используются временные и пространственные ограничения.

Диаграммы компонентов (component diagram)

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

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.

Основные графические элементы диаграммы компонентов

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

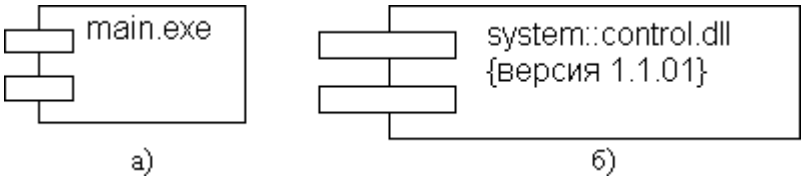

Рис. 63 Графическое изображение компонента в языке UML

В первом случае (рис. 63, а) с компонентом уровня экземпляра связывается только его имя, а во втором (рис. 63, б) – дополнительно имя пакета и помеченное значение.

В языке UML выделяют три вида компонентов:

- *компоненты развертывания*, которые обеспечивают непосредственное выполнение системой своих функций: динамически подключаемые библиотеки с расширением dll, Web-страницы на языке разметки гипертекста с расширением html и файлы справки с расширением hlp.
- *компоненты-рабочие продукты*: файлы с исходными текстами программ, например, с расширениями h или cpp для языка C++.
- *компоненты исполнения*, представляющие исполнимые модули – файлы с расширением exe.

Следующим элементом диаграммы компонентов являются *интерфейс*. Этот элемент уже рассматривался ранее, поэтому отметим только его особенности, которые характерны для представления на диаграммах компонентов. В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис. 64, а). Семантически линия означает реализацию интерфейса, а наличие интерфейсов у компонента означает, что данный компонент реализует соответствующий набор интерфейсов.

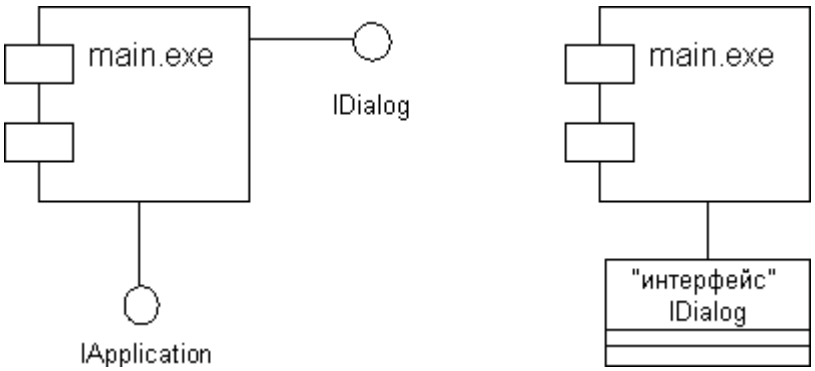

Рис. 64 Графическое изображение интерфейсов на диаграмме компонентов

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом "интерфейс" и возможными секциями атрибутов и операций (рис. 64, б). Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

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. 65). 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 IDialog, который, в свою очередь, реализуется компонентом с именем "image.java". Для второго компонента этот же интерфейс является экспортируемым.

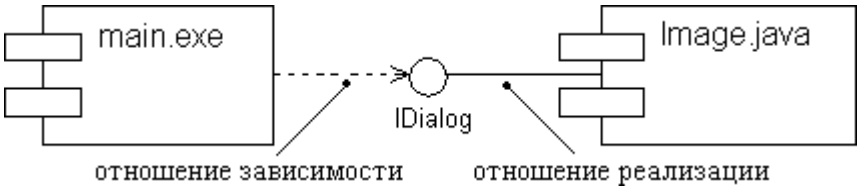

Рис. 65. Фрагмент диаграммы компонентов с отношением зависимости

На диаграмме компонентов также могут быть представлены отношения зависимости между компонентами и реализованными в них классами (рис. 66). Эта информация имеет важное значение для обеспечения согласования логического и физического представлений модели системы.

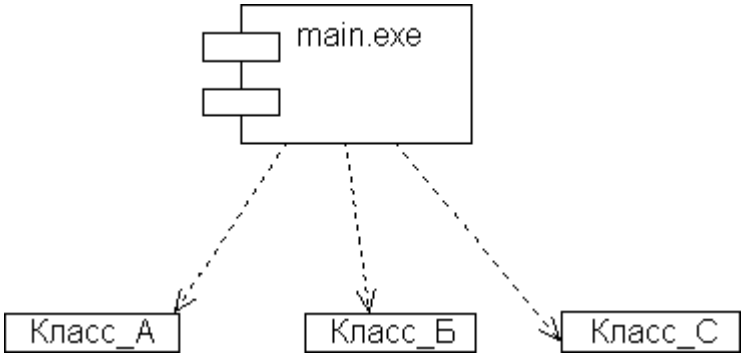

Рис. 66 Графическое изображение зависимости между компонентом и классами

Диаграммы развертывания (deployment diagram)

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* 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.

9.1 Элементы диаграммы компонентов

К основным элементам диаграммы развертывания относятся узлы и соединения.

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. Понятие узла также может включать в себя 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. Узел имеет собственное имя, которое указывается внутри этого графического символа. Сами узлы могут представляться как в качестве типов (рис. 67, а), так и в качестве экземпляров (рис. 67, б).

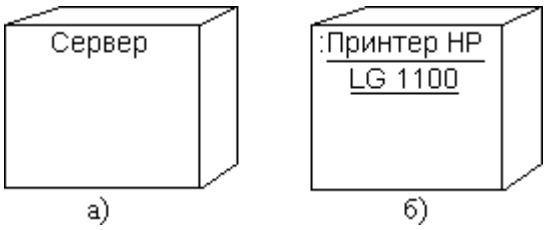

Рис. 67 Графическое изображение узла на диаграмме развертывания

Помеченное значение – это расширение свойств элемента UML, позволяющее вводить новую информацию в его спецификацию. У каждой сущности в UML есть фиксированный набор свойств: классы имеют имена, атрибуты и операции; ассоциации-имена и концевые точки (каждая со своими свойствами) и т.д. Помеченные значения позволяют добавлять новые свойства.

Например, как показано на рис. 68, в диаграмме развертывания можно указать число процессоров, установленных на узле каждого вида, или потребовать, чтобы каждому компоненту был приписан стереотип библиотеки, если его предполагается развернуть на клиенте или сервере.

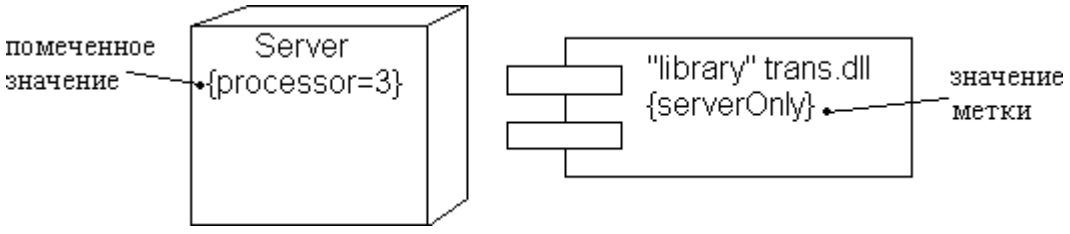

Рис. 68 Помеченные значения

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. 69).

Рис. 69 Графическое изображение узла-экземпляра с дополнительной информацией в форме помеченного значения

Соединения указывают отношения между узлами и являются разновидностью ассоциации. 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 (рис. 70). В рассмотренном примере 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.

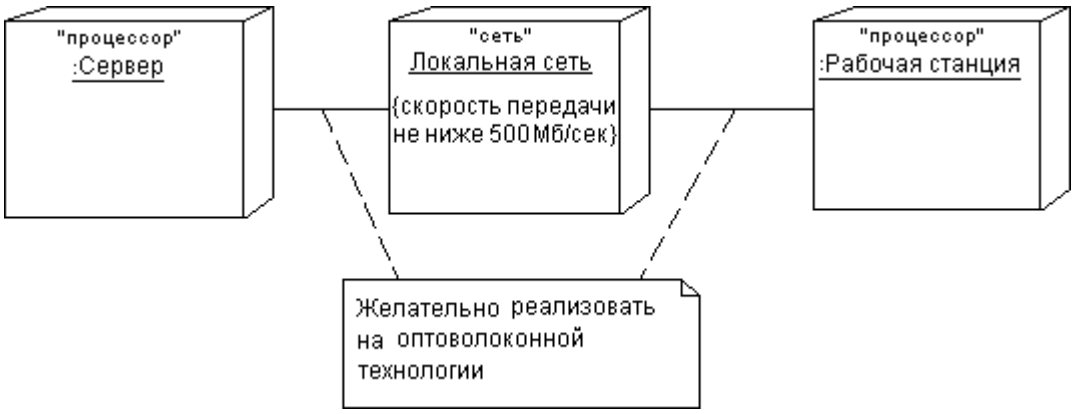

Рис. 70 Фрагмент диаграммы развертывания с соединениями между узлами

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 (рис. 71).

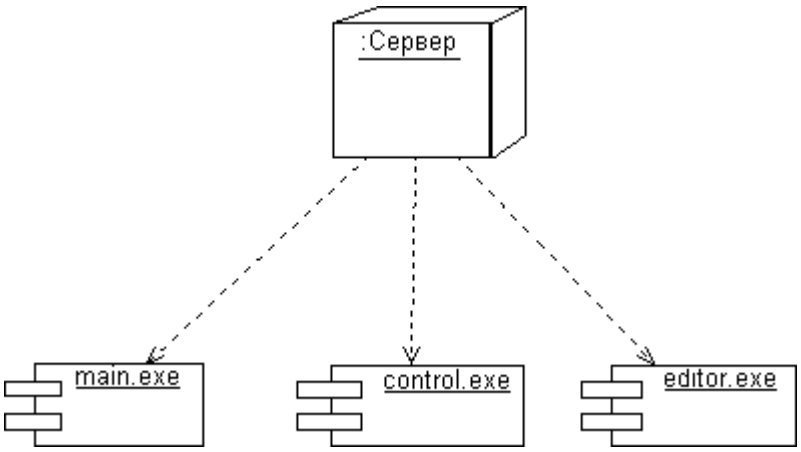

Рис. 71 Диаграмма развертывания с отношением зависимости между узлом и развернутыми на нем компонентами

9.2 Пример диаграммы развертывания

Рассмотрим фрагмент физического представления системы удаленного обслуживания клиентов банка (рис. 72).

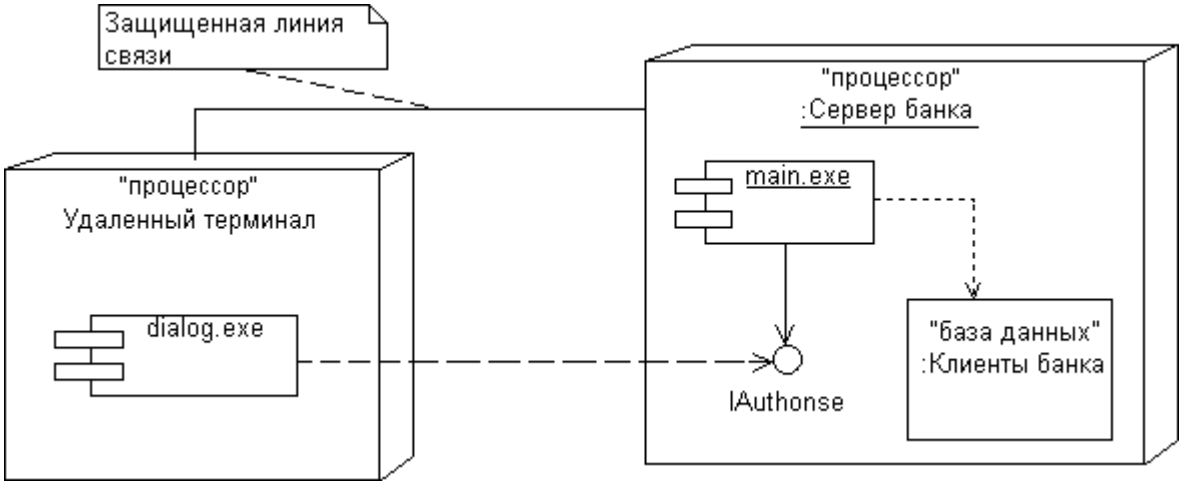

Рис. 72 Диаграмма развертывания для системы удаленного обслуживания клиентов банка

На диаграмме развертывания узлами системы являются удаленный терминал (узел-тип) и сервер банка (узел-экземпляр). Указана зависимость компонента реализации диалога "dialog.exe" на удаленном терминале от интерфейса lAuthorise, 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".