МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»

(ЮЗГУ)

Кафедра информационных систем и технологий

УТВЕРЖДАЮ

Проректор по учебной работе

_____ О. Г. Локтионова

«06» сентября 2017г.

ПОСТРОЕНИЕ ПРОГРАММНОЙ И ТЕХНОЛОГИЧЕСКОЙ МОДЕЛЕЙ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания по выполнению лабораторных работ по дисциплине «Проектирование информационных систем» для студентов направления подготовки бакалавров 09.03.02 Информационные системы 09.03.03 Прикладная информатика

Курск 2017

УДК 004.82 (075.8)

Составитель: Т.И.Лапина

Рецензент

Доктор технических наук, профессор *Р.А.Томакова*

Проектирование информационных систем: методические указания по выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: Т. И. Лапина, Курск, 2017. 21 с.: ил. 8, табл. 0, Библиогр.: с. 3.

Содержат краткие теоретические сведения о методах разработки функциональных моделей при проектированию информационных систем на основе использования нотаций языка UML.

Методические указания соответствуют требованиям программ по направлениям подготовки бакалавров: 09.03.02 Информационные системы, 09.03.03 Прикладная информатика.

Предназначены для студентов направления подготовки

бакалавров 09.03.02 Информационные системы, 09.03.03, Прикладная информатика дневной и заочной форм обучения.

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.

Усл. печ. л. Уч. – изд. л. .Тираж 100 экз. Заказ. Бесплатно.

Юго - Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

1. Цель работы

Изучение моделирования систем на логическом уровне с использованием диаграмм состояний унифицированного языка моделирования UML.

2. Основные теоретические положения

Диаграммы компонентов

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, вообще говоря, не имеют материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния и сообщения, не существуют материально. Они лишь отражают наше понимание статической структуры той или иной системы или динамические 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 от 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 (C++, Pascal, Basic/VBA, Java). При этом уже в тексте программы предполагается некоторая организация программного кода, определяемая синтаксисом языка программирования и предполагающая разбиение исходного кода 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 базовыми элементами физического представления системы в нотации языка UML.

Таким образом, полный проект программной системы представляет собой I 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:

- визуализация общей организации структуры исходного кода программной системы;
- спецификация исполнимого варианта программной системы;
- обеспечение многократного использования отдельных фрагментов программного кода;
- представление 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.

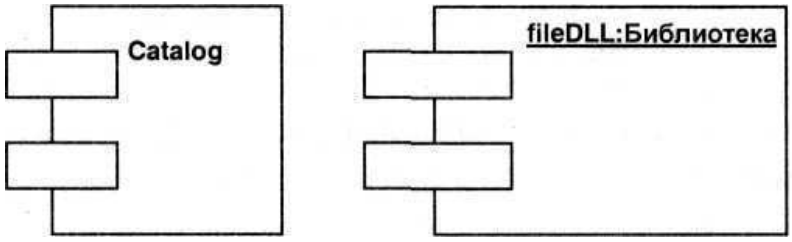
Примечание

Применительно к бизнес-системам программные компоненты следует понимать в более широком контексте, чтобы иметь возможность моделировать бизнес-процессы. В этом случае в качестве компонентов можно рассматривать отдельные организационные подразделения (отделы, службы), должности сотрудников (менеджер, кассир) или документы (счет, заказ, накладная), которые реально существуют в системе.

Компоненты

Для представления физических сущностей в языке UML применяется специальный термин - *компонент* (*component*). В метамодели языка UML компонент является потомком классификатора. Он предназначен для представления физической организации ассоциированных с ним элементов модели. Дополнительно компонент может иметь текстовый стереотип и помеченные значения, а некоторые компоненты - собственное графическое представление.

Компонент служит для общего обозначения элементов физического представления модели и может реализовывать некоторый набор интерфейсов. Для графического представления компонента используется специальный символ — прямоугольник со вставленными слева двумя прямоугольниками меньшего размера (рис. 10.1). Внутри большого прямоугольника записывается имя компонента и, возможно, некоторая дополнительная информация. Этот символ является базовым обозначением компонента в языке UML.



а б

Рис. 1. Графическое изображение компонента в языке UML

Графическое изображение компонента ведет свое происхождение от обозначения модуля программы, применявшегося некоторое время для отображения особенностей инкапсуляции данных и процедур. Так, верхний маленький прямоугольник концептуально ассоциировался с данными, которые реализует этот компонент (ранее он изображался в форме овала). Нижний маленький прямоугольник ассоциировался 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 типов, то в качестве его имени записывается только имя типа с заглавной буквы в форме

<имя типа>.

Если же компонент представляется на уровне примеров, то в качестве его имени записывается в форме

<имя компонента> : <имя типа>.

При этом вся строка имени подчеркивается.

Так, в первом случае (рис. 10.1, а) для компонента уровня типов указывается имя типа, а во втором (рис. 10.1, б) для компонента уровня примеров – собственное имя компонента и имя типа.

Примечание

Хотя правила именования объектов в языке UML требуют подчеркивания имени отдельных экземпляров, применительно к компонентам в литературе подчеркивание их имени часто опускают. В этом случае запись имени компонента со строчной буквы характеризует компонент уровня примеров.

В качестве собственных имен компонентов принято использовать имена исполняемых файлов (с расширением EXE), имена динамических библиотек (расширение DLL), имена Web-страниц (расширение HTML), имена текстовых файлов (расширения TXT или DOC) или файлов справки (HLP), имена файлов баз данных (DB) или имена файлов с исходными текстами программ (расширения H, CPP для языка C++, расширение JAVA для языка Java), скрипты (PL, ASP) и другие.

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

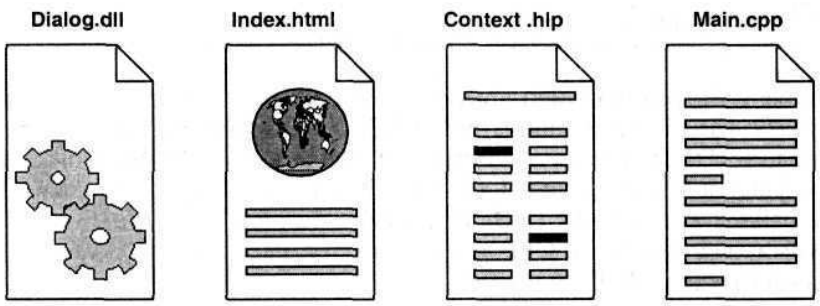
В отдельных случаях к простому имени компонента может быть добавлена информация об имени объемлющего пакета и о конкретной версии реализации данного компонента. Необходимо заметить, что в этом случае номер версии записывается как помеченное значение в фигурных скобках. В других случаях символ компонента может быть разделен на секции, чтобы явно указать имена реализованных в нем интерфейсов. Такое обозначение компонента называется *расширенным* и рассматривается далее в этой главе.

Виды компонентов

Поскольку компонент как элемент модели может иметь различную физическую реализацию, то иногда его изображают в форме специального графического символа, иллюстрирующего конкретные особенности реализации. Строго говоря, эти дополнительные обозначения не

специфицированы в нотации языка UML. Однако, удовлетворяя общим механизмам расширения языка UML, они упрощают понимание диаграммы компонентов, существенно повышая наглядность графического представления.

1) Для более наглядного изображения компонентов были предложены и стали общепринятыми следующие графические стереотипы: во-первых, стереотипы для компонентов развертывания, которые обеспечивают непосредственное выполнение системой своих функций. Такими компонентами могут быть динамически подключаемые библиотеки с расширением DLL (рис. 10.2, а), Web-страницы на языке разметки гипертекста с расширением HTML (рис. 10.2, б) и файлы справки с расширением HLP (рис. 2, в);



а б в г

Рис. 2. Варианты графического изображения компонентов

во-вторых, стереотипы для компонентов в форме рабочих продуктов. Как правило — это файлы с исходными текстами программ, как например, с расширениями H или CPP для языка C++.

Интерфейсы

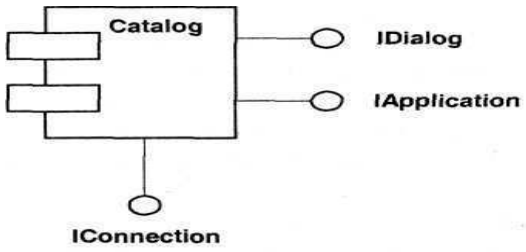
Следующим графическим элементом диаграммы компонентов являются интерфейсы. Последние уже рассматривались, поэтому здесь будут отмечены те их особенности, которые характерны для их представления на диаграммах компонентов.

В общем случае интерфейс графически изображается окружностью, которая соединяется с компонентом отрезком линии без стрелок (рис. 10.3, а). При этом имя интерфейса, которое рекомендуется начинать 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.

Другим способом представления интерфейса на диаграмме компонентов является его изображение в виде прямоугольника класса со стереотипом "интерфейс" и возможными секциями атрибутов и операций (рис. 10.3, б). Как правило, этот вариант обозначения используется для представления внутренней структуры интерфейса, которая может быть важна для реализации.

а



б

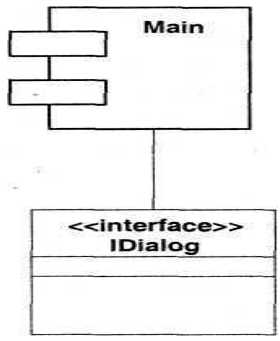


Рис. 3. Графическое изображение интерфейсов на диаграмме компонентов

В общем случае отношение зависимости также было рассмотрено ранее. Напомним, что отношение зависимости служит для представления факта наличия специальной формы связи между двумя элементами модели, 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).

В одних случаях зависимости могут отражать связи отдельных файлов программной системы на этапе компиляции и генерации объектного кода.

В других случаях 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 этом случае рисуют стрелку от компонента-клиента к импортируемому интерфейсу (рис. 10.4). Наличие такой стрелки 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.

Так, например, изображенный ниже фрагмент диаграммы компонентов (рис. 4) представляет информацию о том, что компонент с именем Main зависит от импортируемого интерфейса iDialog, который, в свою очередь, реализуется компонентом с именем Library.

При этом для второго компонента этот интерфейс является экспортируемым.

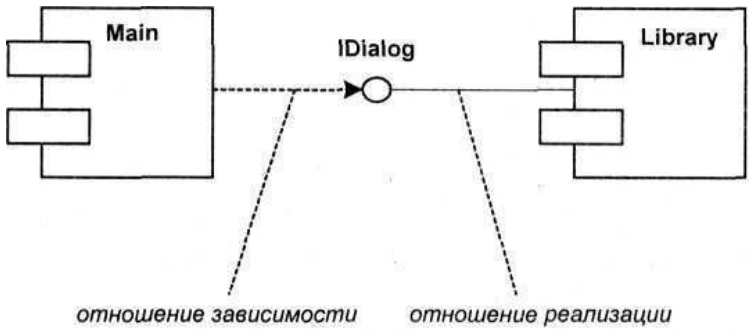


Рис. 4. Фрагмент диаграммы компонентов с отношениями зависимости

Пример построения диаграммы компонентов системы управления банкоматом

Вариант диаграммы компонентов программной системы управления банкоматом изображен на рис. 5.

Данная диаграмма содержит 6 компонентов, 2 из которых являются исполнимыми файлами, один представлен в форме динамической библиотеки, а 3 – в форме текстовых файлов.

При этом компоненты с именами MainATM.cpp и Transactions.cpp содержат исходный код на языке C++, что указывает на особенность их программной реализации. Примечания в форме помеченных значений уточняют реализуемые в этих файлах классы модели. Исполнимый компонент MainATM использует операции интерфейса lAuthorise, которые, в свою очередь, реализованы в исполнимом компоненте MainBank.

Следует отметить, что диаграмма компонентов строится только в том случае, если предполагается программная реализация разрабатываемой модели. В противном случае данный тип канонических диаграмм может вообще отсутствовать в модели, что характерно для проектов документирования и реинжиниринга бизнес-процессов.

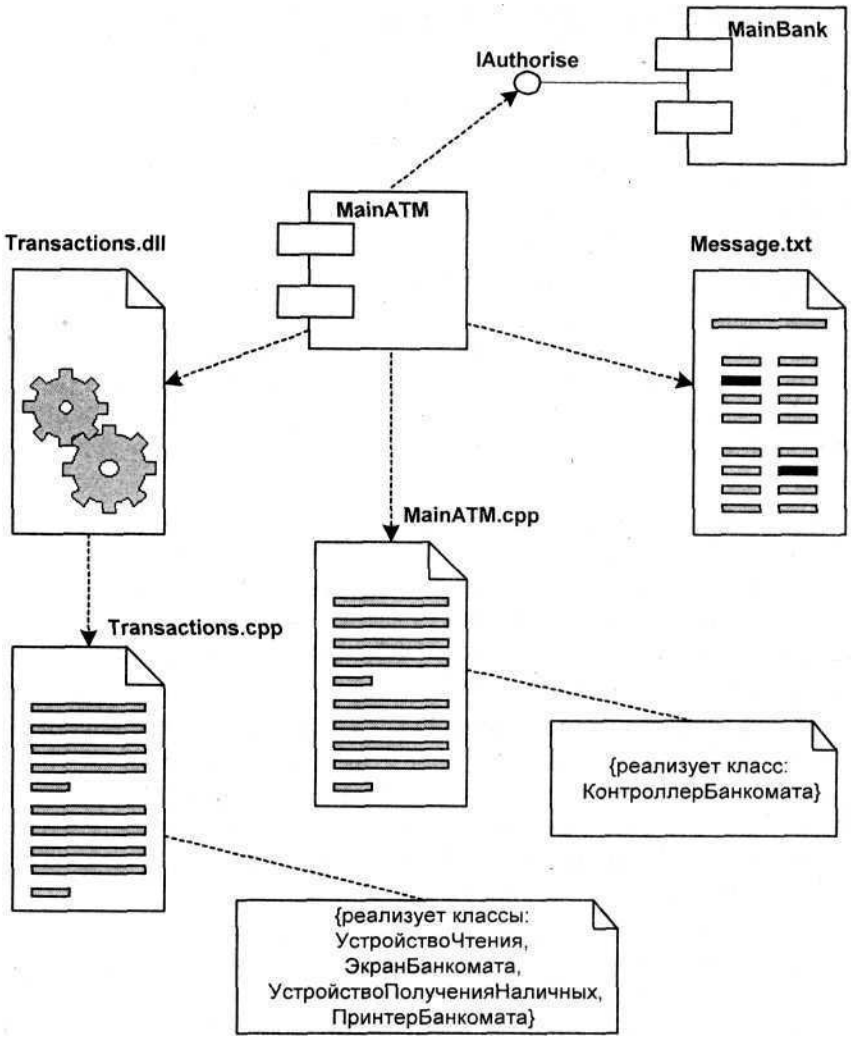


Рис.5. Диаграмма компонентов программной системы управления банкоматом

Создание диаграммы компонентов.

Выберем в качестве языка программирования C++ и для класса *Student* создадим соответствующие этому языку компоненты.

Создание диаграммы компонентов:

1. Дважды щелкните мышью по главной диаграмме компонентов в представлении компонентов.
2. На панели инструментов нажмите кнопку *Package Specification* (Спецификация пакета).

3. Поместите спецификацию пакета на диаграмму.
4. Введите имя спецификации пакета *Student* и укажите в окне спецификации язык C++.
5. На панели инструментов нажмите кнопку *Package Body* (Тело пакета).
6. Поместите тело пакета на диаграмму.
7. Введите имя тела пакета *Student* и укажите в окне спецификации язык C++.
8. На панели инструментов нажмите кнопку *Dependency* (Зависимость).
9. Проведите линию зависимости от тела пакета к спецификации пакета.

Результат показан на рис. 6.

Соотнесение классов с компонентами:

1. В логическом представлении браузера найдите класс *Student*.
2. Перетащите этот класс на спецификацию пакета компонента *Student* в представлении компонентов браузера. В результате класс *Student* будет соотнесен со спецификацией пакета компонента *Student*.
3. Перетащите класс *Student* на тело пакета компонента *Student* в представлении компонентов браузера. В результате класс *Student* будет соотнесен с телом пакета компонента *Student*.

Другой способ отображения классов на компоненты:

1. Двойным щелчком открыть окно спецификации свойств компонента.
2. Открыть вкладку *Realizes*.
3. Отобразить список всех классов модели – флажок *Show all classes*.
4. Щелкнуть правой кнопкой по классу в списке (вызвать контекстное меню).
5. Выполнить операцию *Assign* (Назначить).

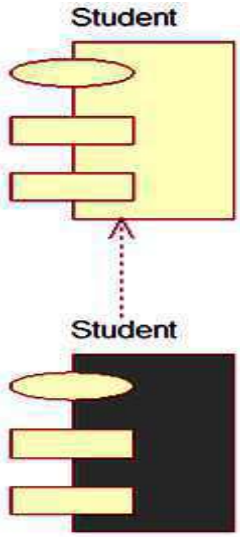


Рис. 6. Диаграмма компонентов

Цели, преследуемые при разработке диаграммы развертывания, следующие:

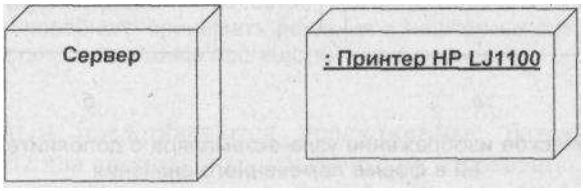
- указать размещение исполнимых компонентов программной системы по ее физическим узлам;
- показать физические связи между всеми узлами реализации системы на этапе ее исполнения;
- выявить узкие места системы и реконфигурировать ее топологию для достижения требуемой производительности.

Для обеспечения этих требований диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками. Далее рассмотрим отдельные элементы, из которых состоят диаграммы развертывания.

Узел (*node*) представляет собой некоторый физически существующий элемент системы, который может обладать некоторым вычислительным ресурсом. В качестве вычислительного ресурса узла может рассматриваться наличие одного или нескольких процессоров, а также некоторого объема оперативной памяти. В языке UML понятие узла расширено — оно 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 узел на диаграмме развертывания изображается в форме трехмерного куба (строго говоря, псевдотрехмерного прямоугольного параллелепипеда). Узел имеет имя, которое указывается внутри этого графического символа. Сами узлы могут представляться как в качестве типов (рис. 1, а), так и в качестве экземпляров (рис. б).



а б

Рис. 11. Графическое изображение узла на диаграмме развертывания

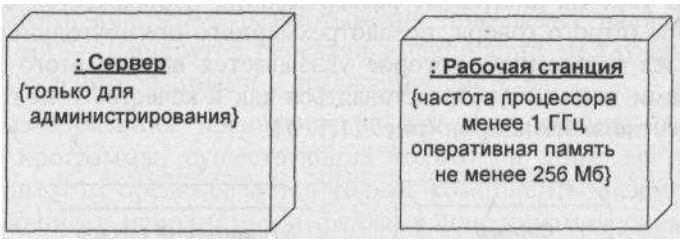
В первом случае имя узла записывается в форме: <имя типа узла> без подчеркивания и начинается с заглавной буквы. Во втором, имя узла-экземпляра записывается в виде <имя узла> : <имя типа узла>, а вся запись подчеркивается. Имя типа узла указывает на некоторую разновидность узлов, присутствующих в модели системы.

Например, аппаратная часть системы может состоять из нескольких персональных компьютеров, часть из которых выполняют функции сервера и соответствуют отдельным узлам-экземплярам в модели. Однако все эти узлы-экземпляры относятся к одному типу узлов, а именно — узлу с именем типа Сервер. Так, на представленном выше рисунке (рис. 1, а) узел с именем сервер относится к общему типу и никак не конкретизируется. Второй узел (рис. 1, 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. 2).

Если необходимо явно указать компоненты, которые размещаются или выполняются 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. 3, а).

Второй способ разрешает показывать на диаграмме развертывания узлы с вложенными изображениями компонентов (рис. 3, б). Важно помнить что в качестве таких вложенных компонентов могут выступать только исполняемые компоненты и динамические библиотеки.



а б

Рис. 2. Графическое изображение узла-экземпляра с дополнительной информацией в форме помеченного значения

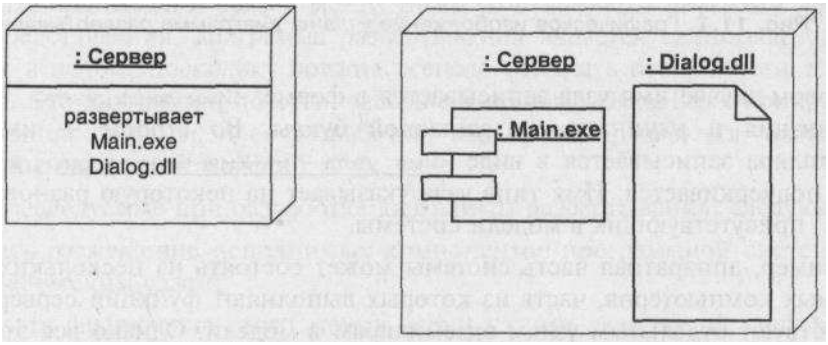


Рис. 3. Варианты графического изображения узлов-экземпляров с размещаемыми на них компонентами

В качестве дополнения к имени узла могут использоваться различные текстовые стереотипы, которые явно специфицируют назначение этого узла. Хотя в языке UML стереотипы для узлов не определены, разработчики предложили следующие текстовые стереотипы: «processor» (процессор), «sensor» (датчик), «modem»

(модем), «net» (сеть), «printer» (принтер) и другие, смысл которых достаточно очевиден.

На диаграммах развертывания допускаются специальные условные обозначения для различных физических устройств, графическое изображение которых проясняет назначение или выполняемые устройством функции. Однако пользоваться этой возможностью следует весьма осторожно, помня, что одно из основных достоинств языка UML следует из его названия -унификация графических элементов визуализации моделей.

На диаграмме развертывания кроме изображения узлов указываются отношения между ними. В качестве отношений выступают физические соединения между узлами, а также зависимости между узлами и компонентами, которые допускается изображать на диаграммах развертывания.

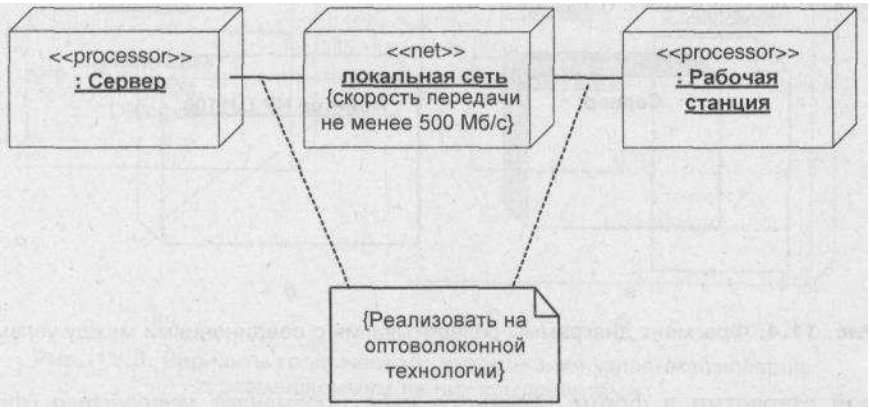


Рис. 5. Фрагмент диаграммы развертывания с соединениями между узлами

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. Так, на представленном ниже фрагменте диаграммы развертывания (рис. 5) 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.

Кроме соединений на диаграмме развертывания могут присутствовать отношения зависимости между узлом и размещаемыми 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..6).

Разработка так называемых встроенных систем реального времени предполагает не только создание программного кода, но и согласование между собой всех аппаратных средств и механических устройств. В качестве примера рассмотрим фрагмент модели управления удаленным механическим средством типа транспортной платформы. Такая платформа предназначена для перемещения в агрессивных средах, где присутствие человека невозможно в силу целого ряда физических причин.

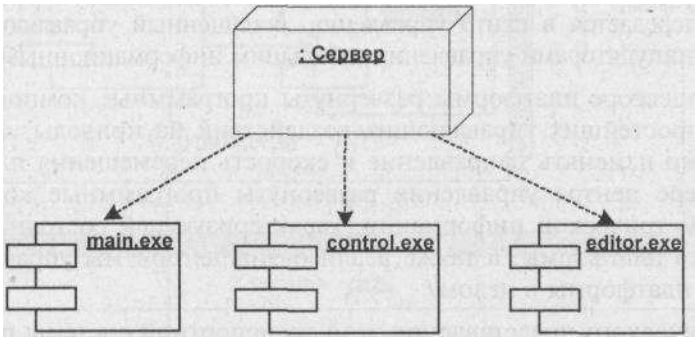


Рис. 16. Диаграмма развертывания с отношением зависимости между узлом и развернутыми на нем компонентами

Пример построения диаграммы развертывания системы управления банкоматом

Продолжая рассмотрение сквозного примера модели системы управления банкоматом, построим диаграмму развертывания для моделирования физической структуры соответствующей программной системы. В рамках разрабатываемой модели может быть построена единственная диаграмма развертывания, на которой следует изобразить отдельные узлы и физические каналы коммуникации между ними. Возможный вариант диаграммы развертывания уровня примеров для программной системы управления банкоматом изображен на рис..8.

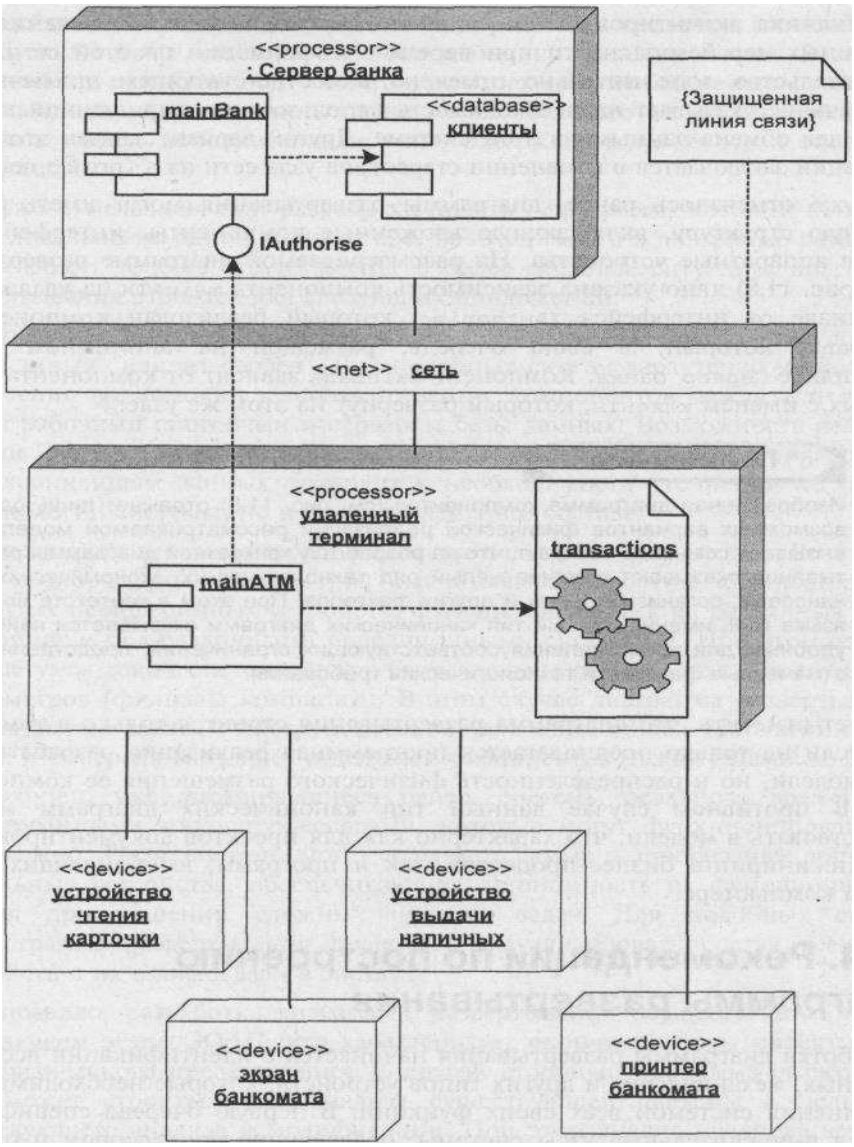


Рис. 8. Диаграмма развертывания системы управления банкоматом

3. Выполнение работы

1. Изучить теоретический материал.
2. Создать логическую модель системы в виде диаграммы компонентов в среде Rational Rose.
3. Создать диаграмму размещения в среде Rational Rose.

4. Содержание отчета

В качестве отчета о выполненной работе предъявите преподавателю:

1)на экране в среде Rational Rose: модель диаграммы компонентов и размещения по индивидуальному заданию;

2)отчет в печатном виде содержащий:

- задание,
- модель диаграммы компонентов и размещения по индивидуальному заданию.

5. Контрольные вопросы

1. Из каких основных элементов состоят диаграммы компонентов?
2. В каком случае используются диаграммы компонентов?
3. Чем отличаются диаграммы компонентов от диаграмм размещения?
4. Из каких основных элементов состоят диаграммы размещения?
5. В каком случае используются диаграммы размещения?
6. Чем отличаются диаграммы компонентов от диаграмм размещения?

6. Библиографический список

1. Золотов, С. Ю. Проектирование информационных систем [Электронный ресурс] : учебное пособие / С. Ю. Золотов. - Томск : Эль Контент, 2013. - 88 с. – Режим доступа : http://biblioclub.ru/index.php?page=book&id=208706
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. - М.: Финансы и статистика, 2002. - 352 с.
3. Вендров А.М. Практикум по проектированию программного обеспечения экономических информационных систем: Учеб. пособие. - М.: Финансы и статистика, 2002. - 192 с.