РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

Тема 2. Описание структуры системы, структурные диаграммы

1. Диаграммы

Диаграмма в UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями), используемое для визуализации системы с разных точек зрения. В UML выделяют 8 типов диаграмм (рис. 1).

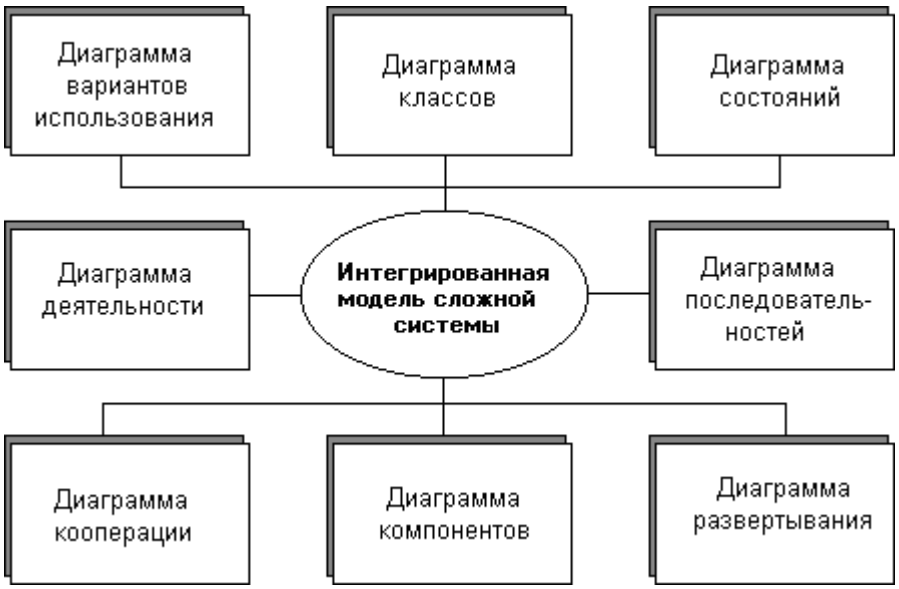

Рис. 1 Интегрированная модель сложной системы в нотации UML

На *диаграмме классов* (Class diagram) изображаются классы, интерфейсы, объекты и кооперации, а также их отношения. Используется при моделировании объектно-ориентированных систем.

На *диаграмме вариантов использования* (Use case diagram) представлены прецеденты и актеры (частный случай классов), а также отношения между ними. Они используются при моделировании поведения системы.

Диаграммы *последовательностей* (Sequence diagram) и *кооперации* (Collaboration diagram) являются частными случаями диаграмм взаимодействия. На диаграммах взаимодействия представлены связи между объектами (сообщения, которыми объекты могут обмениваться). Диаграммы взаимодействия относятся к динамическому виду системы. При этом диаграммы последовательности отражают временную упорядоченность сообщений, а диаграммы кооперации – структурную организацию обменивающихся сообщениями объектов. Эти диаграммы могут быть преобразованы друг в друга.

На *диаграммах состояний* (Statechart diagrams) представлен автомат, включающий состояния, переходы, события и виды действий. Диаграммы состояний используются при моделировании поведения интерфейса, класса или кооперации, зависящем от последовательности событий.

Диаграмма деятельности (Activity diagram) представляют переходы потока управления между объектами от одной деятельности к другой внутри системы.

Диаграмма компонентов (Component diagram) представляет зависимости между компонентами. Диаграммы компонентов отображаются на один или несколько классов, интерфейсов или кооперацій.

На *диаграмме развертывания* (Deployment diagram) представлена конфигурация обрабатывающих узлов системы и размещенных в них компонентов.

2. Диаграммы вариантов использования (use case diagram)

На диаграммах вариантов использования отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает общую функциональность системы. Пользователи, менеджеры проектов, аналитики, разработчики, специалисты по контролю качества и все, кого интересует система в целом, могут, изучая диаграммы вариантов использования, понять, что система должна делать.

2.1 Базовые элементы диаграммы вариантов использования

К базовым элементам рассматриваемой диаграммы относятся *вариант использования*, *актер* и *интерфейс*.

Вариант использования (рис. 2) применяется для спецификации общих особенностей поведения системы или другой сущности без рассмотрения ее внутренней структуры (например, оформление заказа на покупку товара, получение информации о кредитоспособности клиента, отображение графической формы на экране монитора).

Рис. 2 Графическое обозначение варианта использования

Актер – это внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач (рис. 3). При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой.

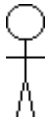

Рис. 3 Графическое обозначение актера

Имя актера должно быть достаточно информативным с точки зрения семантики, например клиент банка, продавец магазина, пассажир авиарейса, водитель автомобиля, сотовый телефон.

Так как в общем случае актер всегда находится вне системы, его внутренняя структура никак не определяется. Для актера имеет значение только его внешнее представление, т.е. то, как он воспринимается со стороны системы. Актеры взаимодействуют с системой посредством передачи и приема сообщений от вариантов использования. Сообщение представляет собой запрос актером сервиса от системы и получение этого сервиса. Это взаимодействие может быть выражено посредством ассоциаций между отдельными актерами и вариантами использования или классами. Кроме этого, с актерами могут быть связаны интерфейсы, которые определяют, каким образом другие элементы модели взаимодействуют с этими актерами.

Интерфейс служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры (рис. 4). В диаграммах вариантов использования интерфейсы определяют совокупность операций, обеспечивающих необходимый набор сервисов или функциональности для актеров. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации. Формально интерфейс эквивалентен абстрактному классу без атрибутов и методов с наличием только абстрактных операций.

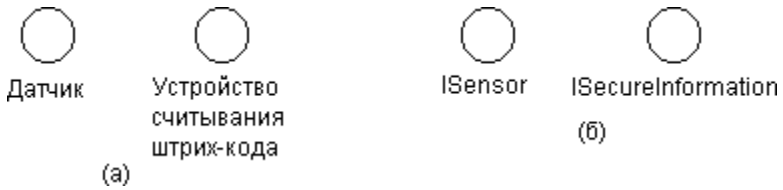

Рис. 4 Графическое изображение интерфейсов на диаграммах вариантов использования

Примечания в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта (рис. 5). В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Рис. 5 Примеры примечаний в языке UML

Применительно к диаграммам вариантов использования примечание может носить самую общую информацию, относящуюся к общему контексту системы.

2.2 Отношения на диаграмме вариантов использования

Для выражения отношений между актерами и вариантами использования применяются стандартные виды отношений.

Отношение ассоциации применительно к диаграммам вариантов использования служит для обозначения специфической роли актера в отдельном варианте использования (рис. 6).

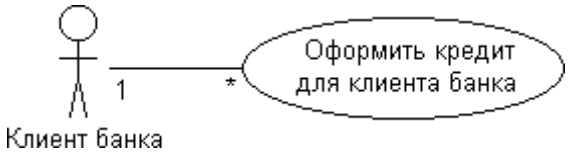

Рис. 6 Отношение ассоциации между актером и вариантом использования

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом “extend” (“расширяет”), как показано на рис. 7.

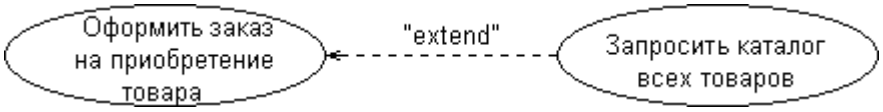

Рис. 7 Отношение расширения между вариантами использования

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

Отношение обобщения графически обозначается сплошной линией со стрелкой, которая указывает на родительский вариант использования (рис. 8).

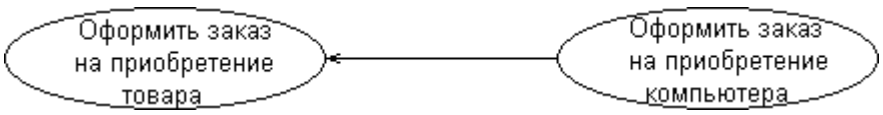

Рис. 8 Отношение обобщения между вариантами использования

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Отношение включения между двумя вариантами использования указывает, что поведение одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Графически данное отношение обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом “include” (“включает”), как показано на рис. 9.

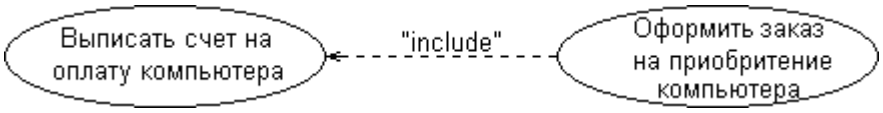

Рис. 9 Отношение включения между вариантами использования

3. 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.

3.1. 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.

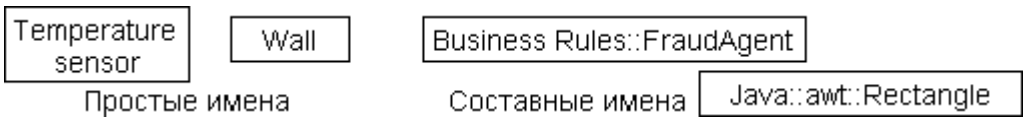

Рис. 10 Простые и составные имена

Атрибут – это именованное свойство класса, включающее описание множества значений, которые могут принимать экземпляры этого класса. Атрибуты представлены в разделе, расположенном под именем класса; при этом указываются их имена, и иногда начальное значение (рис. 11).

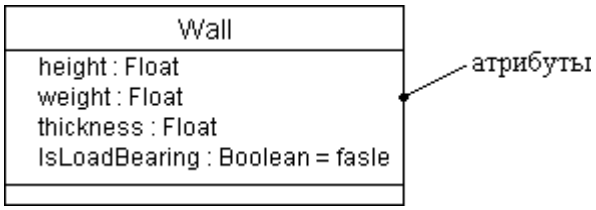

Рис. 11 Атрибуты и их класс

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. 12).

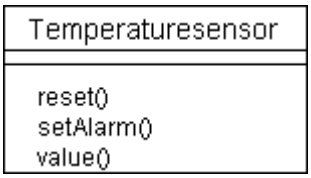

Рис. 12 Операции

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. 13).

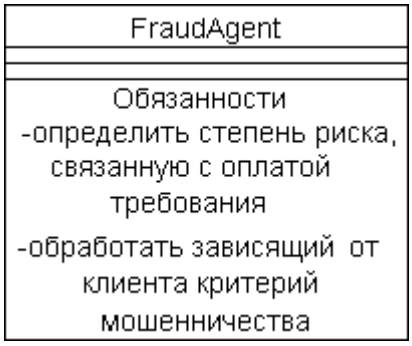

Рис. 13 Обязанности

3.2. Примеры диаграмм классов

На рис. 14 показана совокупность классов, взятых из информационной системы вуза. Этот рисунок содержит достаточное количество деталей для конструирования физической базы данных. В нижней части диаграммы расположены классы Студент, Курс и Преподаватель. Между классами Студент и Курс есть ассоциация, показывающая, что студенты могут посещать курсы. Более того, каждый студент может посещать любое количество курсов и на каждый курс может записаться любое число студентов.

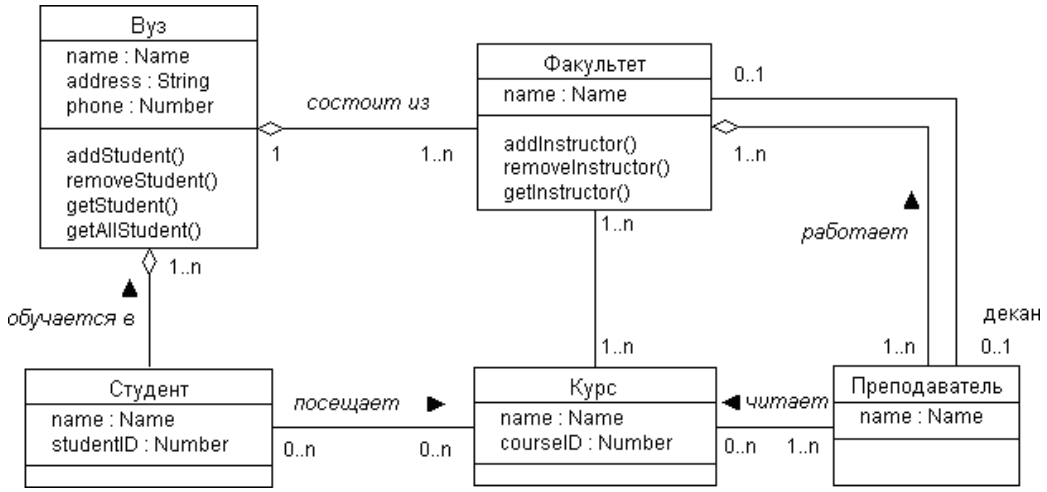

Рис. 14 Диаграмма классов для информационной системы ВУЗа

При моделировании возникает необходимость в указании количества объектов, связанных посредством одного экземпляра ассоциации. Это число называется *кратностью* (Multiplicity) роли ассоциации и записывается либо как выражение, значением которого является диапазон значений, либо в явном виде (рис. 15). Кратность указывает на то, столько объектов должно соответствовать каждому объекту на противоположном конце. Кратность можно задать равной единице (1), указать диапазон: “ноль или единица” (0..1), “много” (0..*), “единица или больше” (1..*). Разрешается также указывать определенное число (например, 3).

Рис. 15 Кратность

4. 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.

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

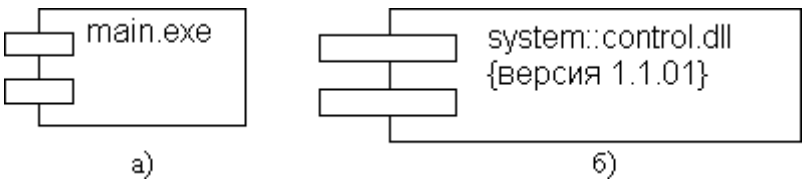

Рис. 16 Графическое изображение компонента в языке UML

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

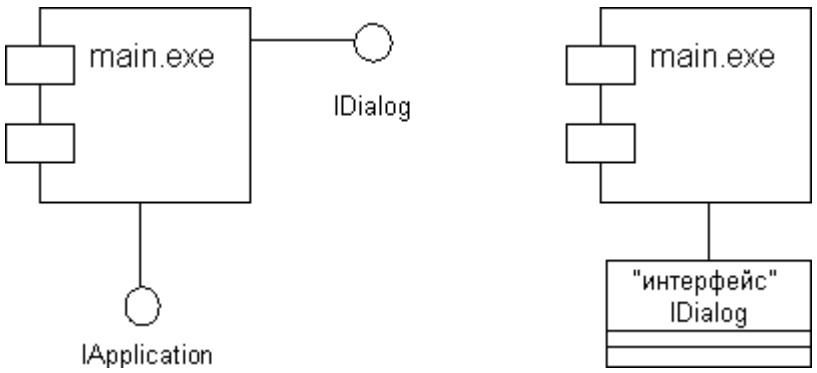

Рис. 17 Графическое изображение интерфейсов на диаграмме компонентов

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.

Литература

1. А.М. Вендров. CASE-технологии. Современные методы и средства проектирования информационных систем. М.: Финансы и статистика, 1998, -176с.
2. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. – М.: ДМК Пресс, 2001 – 176с.: ил.
3. Маклаков С.В. ERWin и BPWin. CASE средства разработки информационных систем. - М.:Диалог-МИФИ, 1999.
4. Джеймс Рамбо, Айвар Якобсон, Гради Буч «UML Специальный справочник». – СПб.: «Питер», 2002.
5. Бабушкин М., Иваненко С., Коростелев В. Web-сервер в действии. - Санкт-Петербург, Питер. – 1997.