22. Представления UML

В языке UML нет четких границ между различными концепциями и конструкциями, но для удобства выделяют несколько представлений. **Представление модели** -- это просто подмножество конструкций, которое представляет один из аспектов моделируемой системы. На самых высоких уровнях абстракции различают три основных представления: 1) структурная классификация; 2) динамическое поведение и 3) управление моделью.

В табл. 1. приведены представления модели, используемые в языке UML, а также относящиеся к ним концепции и диаграммы.

 Структурная классификация описывает системные сущности и их отношения между собой. В число классификаторов, имеющихся в моделях UML, входят классы, варианты использования, компоненты и узлы. Классификаторы являются базой, на которой строится динамическое поведение системы. К представлениям классификации относятся статическое представление, представление вариантов использования и представление программной реализации.

**Динамическое поведение** описывает поведение системы во времени. Поведение можно определить как ряд изменений в мгновенных снимках системы, полученных со статической точки зрения. Представления моделей динамического поведения включают в себя представление конечных автоматов, представление деятельности и представление взаимодействия.

**Представление управления моделью** - это описание разбиения модели на иерархические блоки. Групповой организационный блок называется пакетом. Отдельные пакеты включают модели и подсистемы. Представление управления моделью организует все остальные представления моделей и позволяет осуществлять процесс разработки и управления конфигурацией.

Кроме того, есть несколько конструкций, которые обеспечивают пусть ограниченную, но полезную **функцию расширения возможностей языка UML**. Эти конструкции можно применять к элементам любых представлений модели. В число таких конструкций входят **ограничения** (constraints), **стереотипы** (stereotypes) и **именованные значения** (tagged values).

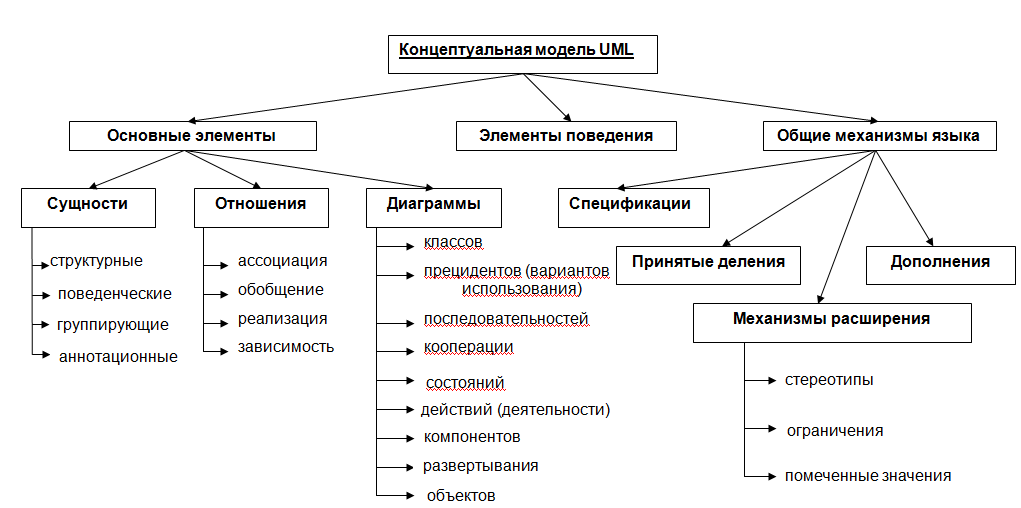
23. Связи и отношения в языке UML. Общая характеристика. Отношение обобщения. Отношение реализации.

Кроме внутреннего устройства или структуры классов на соответствующей диаграмме языка UML указываются различные **отношения (relationship)** между классами. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена семантикой этих типов отношений. Эта совокупность объединяется под общим названием **зависимость**.

**Базовыми отношениями (нотациями) или связями в языке UML являются:**

* Отношение ассоциации (association relationship);
* Отношение обобщения (generalization relationship);
* Отношение реализации (realization relationship);
* Отношение зависимости (dependency relationship).

Каждое из этих отношений имеет собственное графическое представление на диаграмме, которое отражает взаимосвязи между объектами соответствующих классов.



UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО систем. UML призван поддерживать процесс моделирования ПС на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

**Обобщение** - отношение специализации/обобщения, в котором объекты специализированного элемента (потомка, ребенка) могут заменять (наследовать) объекты обобщенного элемента (предка, родителя). Иначе говоря, потомок разделяет структуру и поведение родителя. Как показано в табл. 1. , обобщение изображается в виде сплошной стрелки с полым наконечником, указывающим на родителя. Такой тип взаимосвязи объектов используется при моделировании иерархии “общее/частное”и соответствует отношению **Является видом** (например, **Нападающий Является видом Игрока** ).

Класс может не иметь родителя. В этом случае он называется **базовым** или **корневым классом**. Класс также может не иметь дочернего класса. В этом случае он называется **листовым классом**.

**Реализация** - семантическое отношение между классификаторами, где один классификатор определяет контракт, который другой классификатор обязуется выполнять (к классификаторам относят классы, интерфейсы, компоненты, элементы Use Case, кооперации). Отношения реализации применяют в двух случаях: между интерфейсами и классами (или компонентами), реализующими их; между элементами Use Case и кооперациями, которые реализуют их. Как показано в табл. 3.3, реализация изображается в виде пунктирной стрелки с треугольным незакрашенным наконечником, который располагается у элемента, предоставляющего спецификацию. Хвост стрелки находится у элемента, который предоставляет программную реализацию.

24. Понятие отношения ассоциации. Его общее обозначение на диаграммах UML. Полюс ассоциации.

Отношение ассоциации является одним из фундаментальных понятий в языке UML, и, в той или иной степени, используется при построении всех графических моделей систем в форме канонических диаграмм.

**Ассоциация (association)** – это описание набора однородных связей между объектами двух типов. Она обозначает семантическое соединение классов. основная задача ассоциации состоит в обеспечении взаимодействия объектов, принадлежащих разным классам.

отношение ассоциации обозначается непрерывной линией. Общий вид обозначения отношения ассоциации показан на рис. 1. . Следует отметить, что обязательным элементом обозначения является **маршрут ассоциации**, состоящий из одного или нескольких непрерывных сегментов, которые обычно изображаются прямой линией. Остальные элементы используются по мере необходимости.

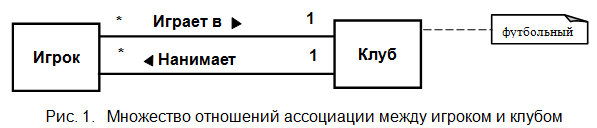
Непосредственно над линией указывается **имя ассоциации** – строка текста, представляющая собой уникальное имя среди имен ассоциаций и

классов, содержащихся в пакете-контейнере. Для наглядности можно показать направление ассоциации с помощью закрашенного треугольника.

У краев прямоугольника, обозначающего класс, находится **полюс ассоциации (association end)**, представляющий собой структурную часть ассоциации, которая определяет, какой объект в ней участвует. Полюса ассоциации имеют четкое местоположение и в принципе не взаимозаменяемы. Полюс содержит описание того, каким образом класс принимает участие в ассоциации. Для этого, между концом маршрута (линией) и классом размещается один или несколько указателей. В случае если указателей несколько, то они располагаются поочередно, как показано на рис. 1. – стрелка навигации, ромб агрегации или композиции, квалификатор и т. п. Дополнительно, над/под полюсом ассоциации может размещаться дополнительная информация – имя роли, кратность (множественность, мощность) и т. д.

1. Роль, кратность и ограничения ассоциации.

**Роль ассоциации**. Каждая ассоциация, как и любая связь, обладает направлением или в терминах UML - ролью (association role). Ролей может быть две, чтобы иметь возможность подчеркнуть возможные различия во "взаимоотношениях" между классами, например у "Начальника" могут быть сосредоточены все права, а у "Подчиненного" - только обязанности. В приведенном выше примере, у класса **"Игрок**" будет роль "Наемный работник", а у класса **"Клуб" –** "Наниматель", так как клуб нанимает игрока. кстати, этот факт, можно отобразить в виде второй ассоциации, имеющей обратное направление по отношению к первой, как показано на рис.1.



**кратность (множественность, мощность) ассоциации**. показывает количество конкретных объектов, которые могут быть связаны с данным партнером ассоциации. В общем случае, множественность показывает нижнюю и верхнюю границы количества объектов, которые могут участвовать в ассоциации. Кратность отдельного класса обозначается в виде интервала целых чисел, аналогично кратности атрибутов и операций классов, и обычно имеет следующий формат.

**<нижняя граница>..<верхняя граница>**

Если для верхней границы стоит символ ' **\*** ', то это символизирует, что она (верхняя граница) бесконечна. Если указывается одно конкретное число, то это означает, что количество объектов фиксировано. А если элементы списка кратности разделены запятыми, то запятая символизирует ЛОГИЧЕСКОЕ ИЛИ.

Например: 0..1 - 0 или 1

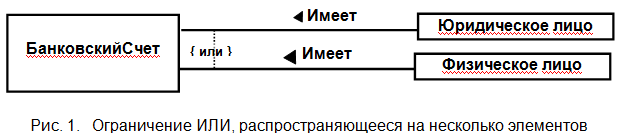
0..\* - 0 или более

\* - любое количество

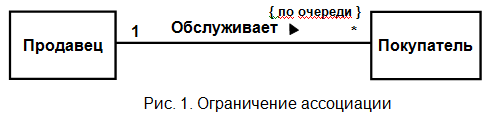
10 - 10

2, 4 - 2 или 4

3..5,10..20,100,200..\* - (от 3 до 5) или (от 10 до 20) или 100 или 200 и более

**Ограничения ассоциации**. Ограничения расширяют UML-блоков, позволяя добавлять новые правила или модифицировать уже существующие. Ограничение показывают как текстовую строку, заключенную в фигурные скобки и располагающуюся рядом с соответствующими элементами. На рис 1. показано ограничение на две ассоциации, располагающееся на пунктирной линии, соединяющей эти элементы (возможно размещение ограничения возле этой линии). 

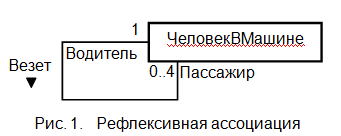
Иногда ассоциация между двумя классами должна удовлетворять некоторому правилу. С помощью нотации ограничений можно изобразить некоторые свойства ассоциации, например, порядок и изменчивость. на рис 1. показан пример, который определяет обслуживание покупателей продавцом по очереди.



1. Рефлексивные ассоциации. Классы ассоциации. Квалификаторы. Навигация.

**Рефлексивные ассоциации**. Иногда класс может находиться в ассоциации с самим собой. Этот вариант носит название **рефлексивной ассоциации** и возникает в том случае, если объекты класса выполняют несколько ролей. Например, сотрудник организации может быть одновременно и подчиненным и руководителем. Например, в университете декан подчиняется ректору, но руководит сотрудниками факультета (заведующими кафедр, преподавателями и т. п.). А человек находящийся в машине, может быть или пассажиром или водителем. Если он водитель, то он везет одного или нескольких пассажиров, или не везет ни одного.

На диаграмме этот вариант ассоциации отображается с помощью маршрута ассоциации, ведущей от прямоугольника класса к этому же прямоугольнику (рис 1. ).



**Классы ассоциаций**. Ассоциация, в свою очередь может иметь атрибуты и операции, т. е. в свою очередь являться классом. Поэтому в этом случае говорят о **классе ассоциаций**. Для его отображения используется обычное обозначение класса, к которому добавляется пунктирная линия, соединяющая этот класс с маршрутом (линией) ассоциации (рис. 1. ). При необходимости, класс ассоциации может связываться с другими классами.

**Играет в**

Рис. 1. Обозначение класса ассоциаций

**Клуб**

**Игрок**

**1**

**\***

футбольный

Нанимает

**1**

**\***

**Контракт**

**1**

**\***

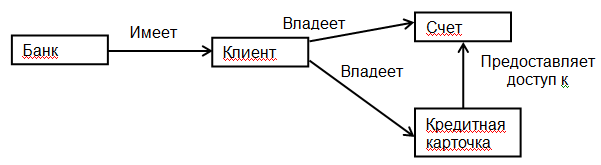
**Предлагается**

**Менеджер**

**Квалификатор**. Если кратность ассоциации описывается отношением "один ко многим", то выполнения отведенной ему в ассоциации роли должен выбрать конкретный объект другого класса. Он может это сделать на основе некоторого заданного атрибута, представляющего собой идентификатор (чаще всего – некоторый числовой идентификатор). Например, среди множества студентов (**\*** или конкретное число, например, **10 000**) вуза (1) можно выбрать конкретного студента по номеру его студенческого билета (:Word), который является уникальным. В UML идентифицирующая информация называется **квалификатором**. Он обозначается небольшим прямоугольником, который прилегает к обозначению класса, выполняющего поиск. Такой подход позволяет эффективно свести кратность "один ко многим" к случаю "один к одному". Пример использования нечислового квалификатора в отношении ассоциации показан на рис. 1. .



**Возможность навигации** указывает, можно ли с помощью выражений, описывающих класс, получить объект или множество объектов, ассоциированных с данным классом. Направление навигации определяется тем, что какой класс предоставляет операцию, а какой - пользуется ею. Иными словами, навигация происходит от класса-потребителя к классу-производителю (рис. 1. ).

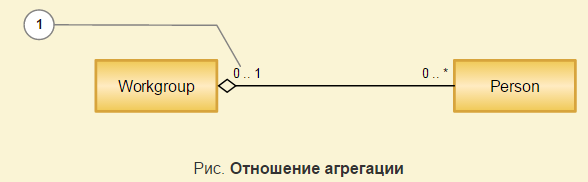


1. Агрегация и композит. Тернарная ассоциация и ассоциации более высокой арности

В UML используются два частных, но очень важных случая отношения ассоциации, которые называются агрегацией и композицией. В обоих случаях речь идет о моделировании отношения типа "часть ‒ целое". Ясно, что отношения такого типа следует отнести к отношениям ассоциации, поскольку части и целое обычно взаимодействуют.

Агрегация (aggregation) ‒ это ассоциация между классом A (часть) и классом B (целое), которая означает, что экземпляры (один или несколько) класса A входят в состав экземпляра класса B.

Это отмечается с помощью специального графического дополнения: на полюсе ассоциации, присоединенному к «целому», изображается незакрашенный ромб 1. Например, на следующем рисунке указано, что сотрудник является членом рабочей группы Workgroup.



При этом никаких дополнительных ограничений не накладывается: экземпляр класса Person (часть) может быть связан с другими объектами (т.е. класс Person может участвовать в нескольких агрегациях), создаваться и уничтожаться независимо от экземпляров класса Workgroup (целого).

***Композиция* (composition) ‒** это ассоциация между классом A (часть) и классом B(целое), которая дополнительно накладывает более сильные ограничения в сравнении с агрегацией: композиционно часть A может входить только в одно целое B, часть существует, только пока существует целое и прекращает свое существование вместе с целым.

Однако часть может быть отделена от целого до того, как оно будет удалено. В указанном случае композиция будет разрушена.

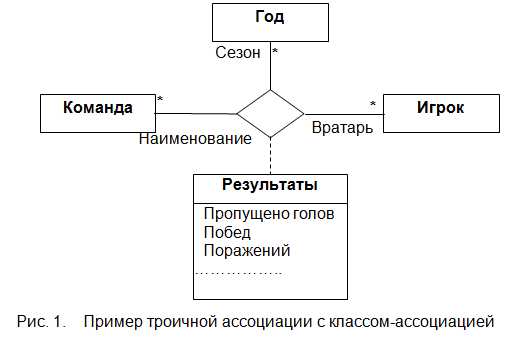
Графически отношение композиции отображается закрашенным ромбом 1∇. Для примера далее на рисунке приведен еще один взгляд на отношения между рабочими группами и сотрудниками в информационной системе отдела кадров. В этом случае, мы считаем, что в организации принята жесткая ("армейская") структура: каждый сотрудник входит ровно в одну рабочую группу и в каждой рабочей группе есть по меньшей мере один сотрудник. Для моделирования такой структуры используется композиция. Если же структура более аморфна: возможны "висящие в воздухе" сотрудники, бывают "пустые" рабочие группы и т.д., то для моделирования такой структуры более адекватным средством является агрегация (см. предыдущий рисунок).



**Тернарна ассоциация и ассоциации более высокой арности** в общем случае называются N-арной ассоциацией. Такая ассоциация связывает некоторым отношением 3 и более классов, при этом один класс может участвовать в ассоциации более чем один раз. Класс ассоциации имеет определенную роль в соответствующем отношении, что может быть явно указано на диаграмме. Каждый экземпляр N-арной ассоциации представляет собой N-арный кортеж значений объектов из соответствующих классов. Бинарная ассоциация является частным случаем N-арной ассоциации, когда значение N=2, и имеет свое собственное обозначение.

N-арная ассоциация графически обозначается ромбом, от которого ведут линии к символам классов данной ассоциации. В этом случае ромб соединяется с символами соответствующих классов сплошными линиями. Обычно линии проводятся от вершин ромба или от середины его сторон. Имя N-арной ассоциации записывается рядом с ромбом соответствующей ассоциации.

Порядок классов в N-арной ассоциации, в отличие от порядка множеств в отношении, на диаграмме не фиксируется. Некоторый класс может быть присоединен к ромбу пунктирной линией. Это означает, что данный класс обеспечивает поддержку свойств соответствующей N-арной ассоциации, а сама N-арная ассоциация имеет атрибуты, операции и/или ассоциации. Другими словами, такая ассоциация, в свою очередь, является классом с соответствующим обозначением в виде прямоугольника и является самостоятельным элементом языка UML - ассоциацией-классом (Association Class), как показано на рис. 1. . N-арная ассоциация не может содержать символ агрегации ни для какой из своих ролей.



Интервал записывается рядом с концом ассоциации и для N-арной ассоциации означает потенциальное число отдельных экземпляров или значений кортежей этой ассоциации, которые могут иметь место, когда остальные N-1 экземпляров или значений классов фиксированы.

1. Отношение зависимости

**Зависимость** — семантическое отношение между двумя предметами, в котором изменение в одном предмете (независимом предмете или поставщике) может влиять на семантику другого предмета (зависимого предмета или клиента). Другими словами, зависимость применяется тогда, когда необходимо показать, что одна сущность использует другую. Как показано в табл. 1. , зависимость изображается в виде пунктирной стрелки, которая связывает два элемента модели. Тот элемент модели, который находится у хвоста стрелки (клиент) зависит от элемента, который находится у ее наконечника (поставщик). У стрелки может находиться одно из ключевых слов, приведенных в табл. 1. , указывающее на вид данной зависимости, а также имя этой зависимости.

Пример отношения зависимости показан на рис 1. .

Класс **Заказ** зависит от класса **Книга**, так как класс **Книга** используется в операциях класса **Заказ**. Класс **Просмотр заказа** использует класс **Заказ**, причем, класс **Заказ** ничего не знает о **Просмотре заказа**. Таким образом, зависимость между этими классами может быть помечена стереотипом «friend» (быть дружественным), который расширяет простую зависимость, определенную в языке. Другие возможные виды зависимостей и соответствующие им стереотипы приведены в таблице 1. .

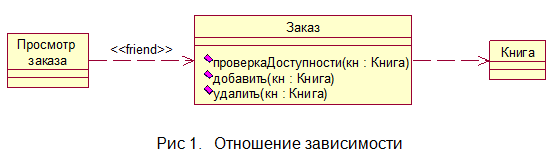


Таблица 1.

**Виды зависимостей**

|  |  |  |
| --- | --- | --- |
| **Зависимость** | **Функция** | **Ключевое слово** |
| Доступ (Access) | Пакет имеет доступ к содержимому другого пакета | «access» |
| Связывание  (Binding) | Присваивание значений параметрам шаблона для создания нового элемента модели | «bind» |
| Вызов (Call) | Метод одного класса вызывает операцию другого класса | «call» |
| Вывод (Derivation) | Один экземпляр может быть вычислен на основе информации, предоставленной другим экземпляром | «derive» |
| Дружественность (Friend) | Элемент имеет доступ к содержимому другого элемента, вне зависимости от его видимости | «friend» |
| Импорт (Import) | Пакет имеет доступ к содержимому другого пакета и добавляет имена из пространства имен этого пакета в пространство имен импортера | «import» |
| Создание экземпляра (Instantiation) | Метод одного класса создает экземпляры другого класса | «instantiate» |
| Параметр  (Parameter) | Отношение между операцией и ее параметрами | «parameter» |
| Реализация (Realization) | Соотнесение спецификации и ее реализации | «realize» |
| Уточнение  (Refinement) | Соотнесение элементов, находящихся на разных семантических уровнях | «refine» |
| Отправка (Send) | Отношение между объектом, принимающим сигнал, и объектом, который этот сигнал отправляет | «send» |
| Трассировка  (Trace) | Существование связи между элементами различных моделей (менее точной, чем прямое отображение) | «trace» |
| Использование (Usage) | Одному элементу для правильного функционирования необходимы услуги другого элемента (сюда входят вызов, конкретизация, параметр, оправка, другие зависимости) | «use» |