Основы UML

**Типы диаграмм**

Я зык UML (Unified Modeling Language) предоставляет девять типов диаграмм.

Как правило, при рассмотрении статических частей системы используются следующие четыре типа:

* диаграммы классов;
* диаграммы объектов;
* диаграммы компонентов;
* диаграммы развертывания.

Для работы с динамическими частями системы применяются пять типов, перечисленные ниже:

* диаграммы прецедентов;
* диаграммы последовательности;
* диаграммы кооперации;
* диаграммы состояний;
* диаграммы деятельности.

При разработке ПО чаще всего применяются **диаграммы классов** и **диаграммы последовательности** .

Я не буду описывать семантику и синтаксис сущностей, поскольку все современные среды моделирования в большой степени автоматизируют процесс их создания. Более подробно рассмотрю семантику отношений (связей), как наиболее важный элемент языка.

**Отношения**

*Отношением* (Relationship) называется связь между сущностями. В объектно-ориентированном моделировании особое значение имеют четыре типа отношений: *зависимости, обобщения, ассоциации и реализации*.

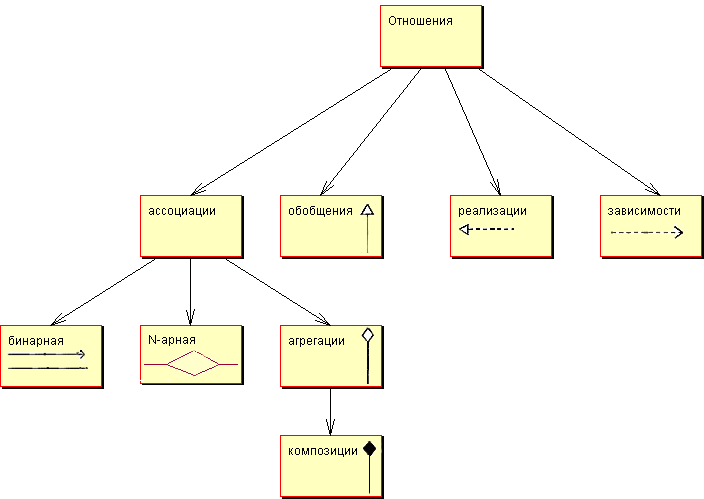


Рисунок 1 Виды отношений UML

Отношения изображаются в виде линий различного начертания, см. рисунок 1.

**Зависимость**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Зависимостью* (Dependency) называют отношение использования ("use a"), согласно которому изменение в спецификации используемого элемента может повлиять на другой элемент, его использующий. Причем обратное не обязательно. | Стрелка направлена от сущности, **которая использует** к **используемой сущности.** | Самым распространенным видом отношения зависимости является соединение между классами, когда один класс использует другой в качестве параметра операции.    Зависимость направлена от класса РасписаниеЗанятий к классу Курс, поскольку последний используется в операциях add() и remove() класса РасписаниеЗанятий. |

**Обобщения**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Обобщение* (Generalization) - это отношение между общей сущностью (суперклассом, или родителем) и ее конкретным воплощением (подклассом, или потомком). Обобщения называют отношениями типа "is a". | Отношение обобщения изображается в виде линии с большой не закрашенной стрелкой, направленной **на родителя** |  |

**Ассоциации**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Ассоциацией* (Association) называется структурное отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа.  Объект может иметь ассоциацию с самим собой. Это означает, что с объектом некоторого класса позволительно связать другие объекты из того же класса.  Ассоциация, связывающая два класса, называется бинарной.  Можно, создавать ассоциации, связывающие сразу несколько классов; они называются n-арными. | Ассоциация может быть двунаправленной (можно перемещаться от объектов одного класса к объектам другого) и однонаправленной (связь в одном направлении).  **Выбирается в зависимости от контекста** | Двунаправленная бинарная ассоциация:    Треугольник обозначает направление чтения имени  Также простой пример двунаправленной ассоциации между двумя классами: Книга и Библиотека.  Однонаправленная бинарная ассоциация:    Обычно означает, что один класс вызывает метод другого.  Тернарная ассоциация: |

*Имя*. Ассоциации может быть присвоено имя, описывающее природу отношения. Чтобы избежать возможных двусмысленностей в понимании имени, достаточно с помощью черного треугольника указать направление, в котором оно должно читаться.

*Роль*. Класс, участвующий в ассоциации, играет в ней некоторую роль. Можно явно обозначить роль, которую класс играет в ассоциации.

*Кратность.* Часто при моделировании бывает важно указать, сколько объектов может быть связано посредством одного экземпляра ассоциации. Это число называется кратностью (Multiplicity) роли ассоциации. Кратность можно задать равной единице (1), можно указать диапазон: "ноль или единица" (0..1), "много" (0..\*), "единица или больше" (1..\*). Разрешается также указывать определенное число.

Лучше избегать n-арных ассоциаций, поскольку при реализации они увеличивают связанность (сoupling), что противоречит General Responsibility Assignment Software Patterns.

Нужно избегать ненаправленных ассоциаций, поскольку при реализации в коде это будет означать циклическую зависимость.

**Агрегирование**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Агрегирование* (aggregation)– это отношение типа "часть/целое", в котором один из классов имеет более высокий ранг (целое) и состоит из нескольких меньших по рангу (частей). Агрегирование причислено к отношениям типа "has a".  Простое агрегирование - чисто концептуальное отношение, оно лишь позволяет отличить "целое" от "части".  Агрегирование является частным случаем ассоциации.  При простом агрегировании "часть" может принадлежать одновременно нескольким "целым". | Изображается в виде простой ассоциации с не закрашенным ромбом **со стороны "целого"**. | Можно привести такой пример агрегации: "у профессора есть студенты". Профессор объединяет студентов ("части") в целое ("группа профессора"). Но обе сущности могут обойтись друг без друга. |

Агрегирование может иметь дополнительные свойства, например, кратность.

Простое агрегирование не влияет на результат навигации по ассоциации между целым и его частями и не подразумевает наличия между ними какой-либо зависимости по времени жизни.

**Композиция**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Композицией* (composition) называется форма агрегирования с четко выраженным отношением владения, причем время жизни частей и целого совпадают.  Это означает, что в случае композиции "часть" в любой момент времени может принадлежать только одному "целому". | Изображается в виде простой ассоциации с закрашенным ромбом **со стороны "целого"**. | Например, в оконной системе класс Frame (Рама) принадлежит только одному классу Window (Окно)  Создав объект Frame в системе окон, вы должны присоединить его к объемлющему окну. Когда объект Window удаляется, он в свою очередь должен уничтожить принадлежащие ему объекты Frame. |

Агрегирование может иметь дополнительные свойства, например, кратность.

При композиции "целое" не может существовать без своих "частей" и отвечает за их диспозицию. То есть композит должен управлять созданием и уничтожением "частей".

**Реализация**

|  |  |  |
| --- | --- | --- |
| **Определение** | **Направление** | **Пример** |
| *Реализацией* (Realization) называется отношение между классификаторами, при котором один из них описывает контракт, а другой гарантирует его выполнение.  Реализации употребляются в двух ситуациях - в контексте интерфейсов и в контексте коопераций. | Изображается реализация в виде пунктирной линии с большой не закрашенной стрелкой, указывающей **на классификатор, который определяет контракт**. |  |

Чаще всего реализации используют для определения отношений между интерфейсом и классом или компонентом, который предоставляет объявленные в интерфейсе операции или услуги.

Реализацию можно использовать также для специфицирования отношений между прецедентом и реализующей его кооперацией.

Если класс или компонент реализуют интерфейс, это означает, что специфицированное интерфейсом поведение действительно будет обеспечено.

1. The Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, Ivar Jacobson
2. <http://habrahabr.ru/post/150041/>
3. <http://www.znannya.org/?view=association_relationship>