**UML**

Unified Modeling Language

Универсальный язык визуального моделирования системы.

UML не задаёт методологии моделирования.

UML не привязан ни к одному из языков.

Основной идеей является возможность моделировать ПО в виде наборов взаимодействующих объектов.

В UML есть два аспекта:

* статическая структура
* динамическая структура

**Статическая** структура описывает какие типы объектов важны для моделирования системы и как они взаимосвязанны.

**Динамическая** структура описывает жизненный цикл этих объектов и то как они взаимодействуют друг с другом для обеспечения необходимой функциональности системы.

**Структура UML**

* строительные блоки
* общие механизмы
* архитектура

**Строительные блоки -** основные элементы, их отношения и диаграммы UML моделей.

**Общие механизмы -** общие UML пути для достижения определенной цели (спецификации, дополнения, принятые распределения, механизмы расширения).

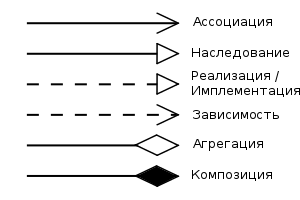
**Архитектура -** UML-разрезы архитектуры системы.

**Строительные блоки**

Сущности - элементы модели. Делятся на:

* структурные - имя существительное UML-модели: класс, интерфейс, кооперация, прецедент, активный класс, компонент
* поведенческие - глаголы UML-модели (взаимодействие, деятельность, автомат)
* группирующие - пакеты, которые используются для группировки семантически связанных элементов.
* аннотирующие - приметки, которые можно добавить к модели для записи специальной информации (типа метаданных).

**Отношения**



**Зависимость** обозначает такое отношение между классами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса, но не наоборот.

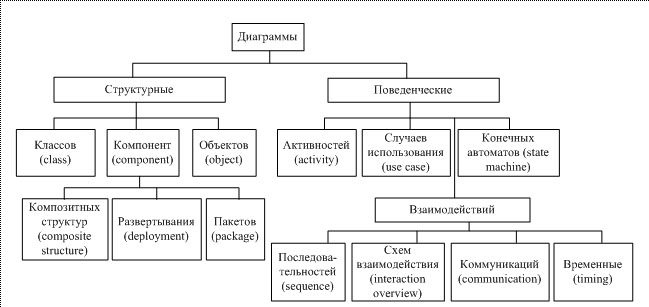
**Ассоциация** показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому. Является общим случаем композиции и агрегации.

**Агрегация** — это разновидность ассоциации при отношении между целым и его частями. Как тип ассоциации агрегация может быть именованной. Одно отношение агрегации не может включать более двух классов (контейнер и содержимое).

**Композиция** — более строгий вариант агрегации. Известна также как агрегация по значению. Композиция имеет жёсткую зависимость времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер будет уничтожен, то всё его содержимое будет также уничтожено.

**Обобщение (наследование) -** диаграмма классов, показывающая наследование двух подклассов от одного суперкласса. Обобщение (Generalization) показывает, что один из двух связанных классов (подтип) является частной формой другого (надтипа), который называется обобщением первого. На практике это означает, что любой экземпляр подтипа является также экземпляром надтипа. Например: животные — супертип млекопитающих, которые, в свою очередь, — супертип приматов, и так далее. Эта взаимосвязь легче всего описывается фразой «А — это Б» (приматы — это млекопитающие, млекопитающие — это животные).

**Реализация** — отношение между двумя элементами модели, в котором один элемент (клиент) реализует поведение, заданное другим (поставщиком). Реализация — отношение целое-часть. Графически реализация представляется так же, как и наследование, но с пунктирной линией. Поставщик, как правило, является абстрактным классом или классом-интерфейсом.



**USE-CASE диаграммы**

Назначение:

* верхний уровень описания системы
* описания возможных сценариев работы системы с точки зрения юзера

Прецеденты помогают в разработке ГУИ, но не описывают что именно юзер может делать с графическими компонентами.

Прецеденты разделяют:

* инициатора - тот, кто инициирует процесс
* исполнитель - тот, кто делает всю грязь

---

Основные понятия:

* актёр - юзер, инициатор или исполнитель цели
* прецедент - цель, которой хочет достичь юзер
* ассоциация - связь между актёром и целями
* включение - возможные варианты реализации цели (или их набора) для реализации общей цели
* расширение - новая цель, которая расширяет существующую

