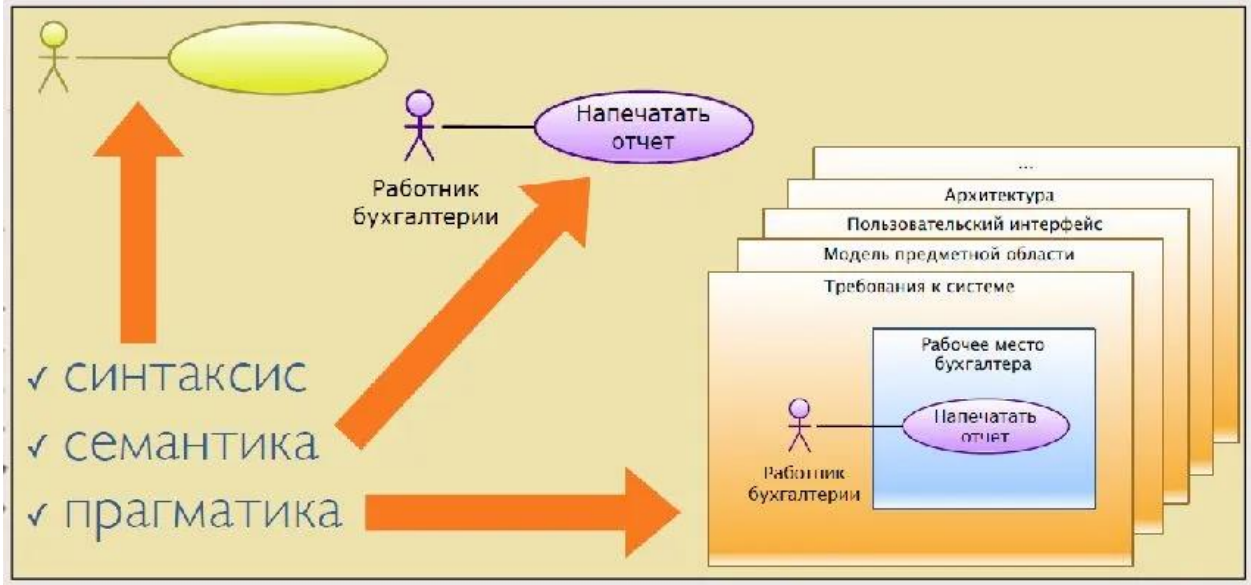
Что такое UML?



***Синтаксис (syntax)**, т.е. определение правил составления конструкций языка.*

***Семантика (semantics)**, т.е. определение правил приписывания смысла конструкциям языка.*

***Прагматика (pragmatics)**, т.е. определение правил использования конструкций языка для достижения определенных целей.*

Синтаксис — это самая простая часть описания алгоритмического языка. На уровне грамматики определяются корректные последовательности символов — лексемы. Если последовательность символов принадлежит языку, то она считается синтаксически правильной. Для программы это означает, что транслятор на ней не выдает ошибки. Но синтаксическая правильность не гарантирует даже осмысленности программы. Таким образом, синтаксис определяет лишь одну сторону языка.

Семантика — это соответствие между синтаксически правильными программами и действиями абстрактного исполнителя, то есть это смысл синтаксических конструкций. Семантика - в программировании - система правил истолкования отдельных языковых конструкций. Семантика определяет смысловое значение предложений алгоритмического языка.

Таким образом, применительно к UML, семантика и синтаксис определяют стиль изложения (построения моделей), который объединяет естественный и формальный языки для представления базовых понятий (элементов модели) и механизмов их расширения.

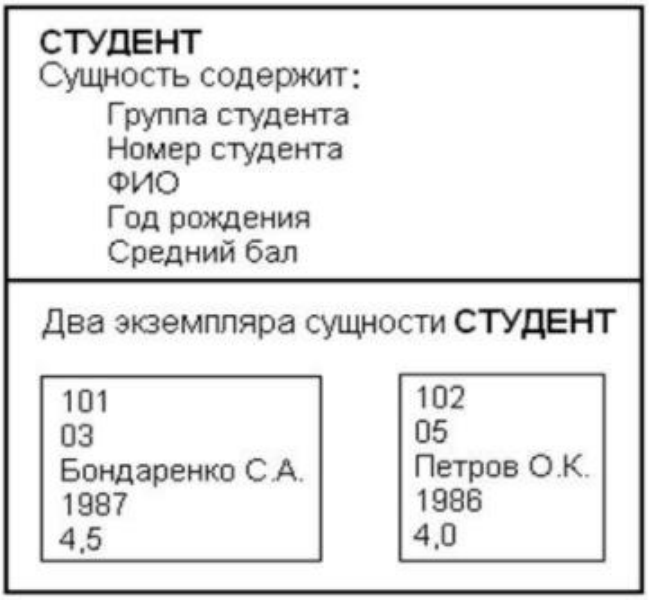
Прагматика — задает конкретизацию абстрактного вычислителя для данной вычислительной системы.

Модель и ее элементы

Модель UML (UML model)— это совокупность конечного множества конструкций языка, главные из которых — это сущности и отношения между ними.

Сами сущности и отношения модели являются экземплярами метаклассов метамодели.

Пример сущности СТУДЕНТ



Структурные сущности предназначены для описания структуры. Обычно к структурным сущностям относят следующие.

Объект (object) — сущность, обладающая уникальностью и инкапсулирующая в себе состояние и поведение.

Класс (class) — описание множества объектов с общими атрибутами, определяющими состояние, и операциями, определяющими поведение.

Интерфейс (interface) — именованное множество операций, определяющее набор услуг, которые могут быть запрошены потребителем и предоставлены поставщиком услуг.
Кооперация (collaboration) — совокупность объектов, которые взаимодействуют для достижения некоторой цели.
Действующее лицо (actor) — сущность, находящаяся вне моделируемой системы и непосредственно взаимодействующая с ней.
Компонент (component) — модульная часть системы с четко определенным набором требуемых и предоставляемых интерфейсов.
Артефакт (artifact) — элемент информации, который используется или порождается в процессе разработки программного обеспечения. Другими словами, артефакт — это физическая единица реализации, получаемая из элемента модели (например, класса или компонента).
Узел (node) — вычислительный ресурс, на котором размещаются и при необходимости выполняются артефакты.

Поведенческие сущности предназначены для описания поведения. Основных поведенческих сущностей всего две: состояние и действие (точнее, две с половиной, потому что иногда употребляется еще и деятельность, которую можно рассматривать как особый случай состояния).
Состояние (state) — период в жизненном цикле объекта, находясь в котором объект удовлетворяет некоторому условию и осуществляет собственную *деятельность* или ожидает наступления некоторого события.
Деятельность (activity) можно считать частным случаем *состояния*, который характеризуется продолжительными (по времени) не атомарными вычислениями.
Действие (action) — примитивное атомарное вычисление.
Состояния бывают самые разные. Кроме того, при моделировании поведения используется еще ряд вспомогательных сущностей, которые здесь не перечислены, потому что сосуществуют только вместе с указанными основными.
Несколько особняком стоит сущность — вариант использования, которой присущи как структурные, так и поведенческие аспекты.
Вариант использования (use case) — множество сценариев, объединенных по некоторому критерию и описывающих последовательности производимых системой действий, доставляющих значимый для некоторого *действующего лица* результат.
Приведенная классификация не является исчерпывающей. У каждой из этих сущностей есть различные частные случаи и вариации.
Группирующая сущность в UML одна — *пакет* — зато универсальная.
Пакет (package) — группа элементов модели (в том числе пакетов).
Аннотационная сущность тоже одна — *примечание*(comment) — зато в нее можно поместить все что угодно, так как содержание примечания UML не ограничивает.

СУЩНОСТИ UML

Отношения

В UML используются четыре основных типа отношений:

- зависимость (dependency);
- ассоциация (association);
- обобщение (generalization);
- реализация (realization).

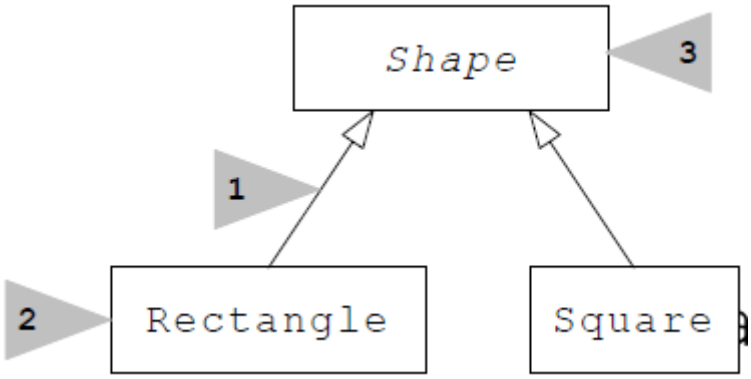
Зависимость — это наиболее *общий* тип отношения между двумя сущностями.

Отношение *зависимости* указывает на то, что изменение независимой сущности каким-то образом влияет на зависимую сущность.

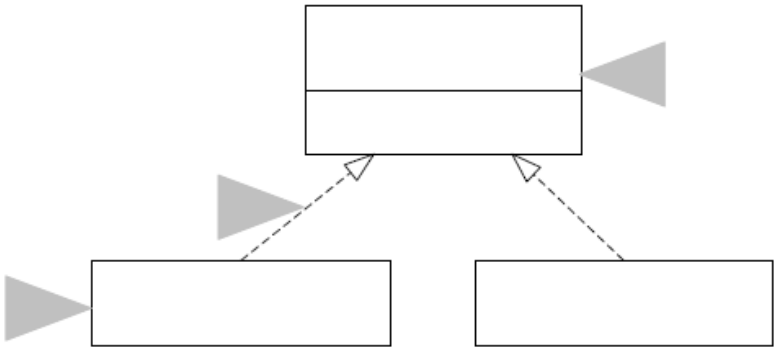
Графически отношение зависимости изображается в виде пунктирной линии со стрелкой, направленной от зависимой сущности к независимой (рис.). Как правило, семантика конкретной зависимости уточняется в модели с помощью дополнительной информации. Например, зависимость со стереотипом «use» означает, что зависимая сущность использует (скажем, вызывает операцию) независимую сущность.

Ассоциация — это наиболее часто используемый тип отношения между сущностями.
Отношение *ассоциации* имеет место, если одна сущность непосредственно связана с другой (или с другими — ассоциация может быть не только бинарной).
Графически ассоциация изображается в виде сплошной линии с различными дополнениями, соединяющей связанные сущности (рис.). На программном уровне непосредственная связь может быть реализована различным образом, главное, что ассоциированные сущности знают друг о друге. Например, отношение часть–целое является частным случаем ассоциации и называется отношением агрегации.

Обобщение — это отношение между двумя сущностями, одна их которых является частным (специализированным) случаем другой.
Графически обобщение изображается в виде линии с треугольной незакрашенной стрелкой на конце (1), направленной от частного (2) (подкласса) к общему (3) (суперклассу)



Отношение *реализации* используется несколько реже, чем предыдущие три типа отношений, поскольку часто подразумеваются по умолчанию.
Отношение *реализации* указывает, что одна сущность является реализацией другой.
Например, класс является реализацией интерфейса. Графически реализация изображается в виде пунктирной линии с треугольной незакрашенной стрелкой на конце, направленной от реализующей сущности к реализуемой.



Диаграммы

Диаграммы UML есть та основная накладываемая на модель структура,
которая облегчает создание и использование модели.

Диаграмма (diagram)— это графическое представление некоторой час-
ти графа модели.

Диаграмма вариантов использования (Use Case diagram)
- Диаграмма классов (Class diagram)
- Диаграмма объектов (Object diagram)
- Диаграмма состояний (State chart diagram)
- Диаграмма деятельности (Activity diagram)
- Диаграмма последовательности (Sequence diagram)
- Диаграмма кооперации (Collaboration diagram)
- Диаграмма компонентов (Component diagram)
- Диаграмма размещения (Deployment diagram)

Иерархия диаграмм

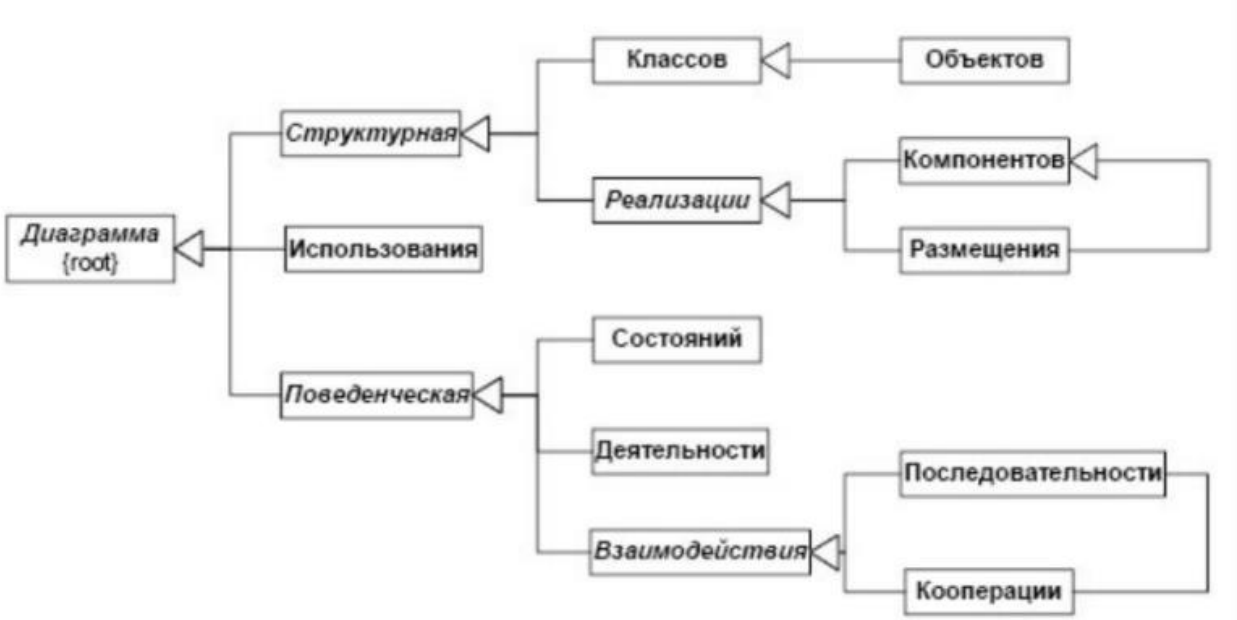


Диаграмма вариантов использования

Конструкция или стандартный элемент языка UML - вариант использования применяется для спецификации общих особенностей поведения системы без рассмотрения внутренней структуры этой сущности. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим лицом. Диаграмма вариантов использования может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами

Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой

Действующее лицо (actor) - это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок

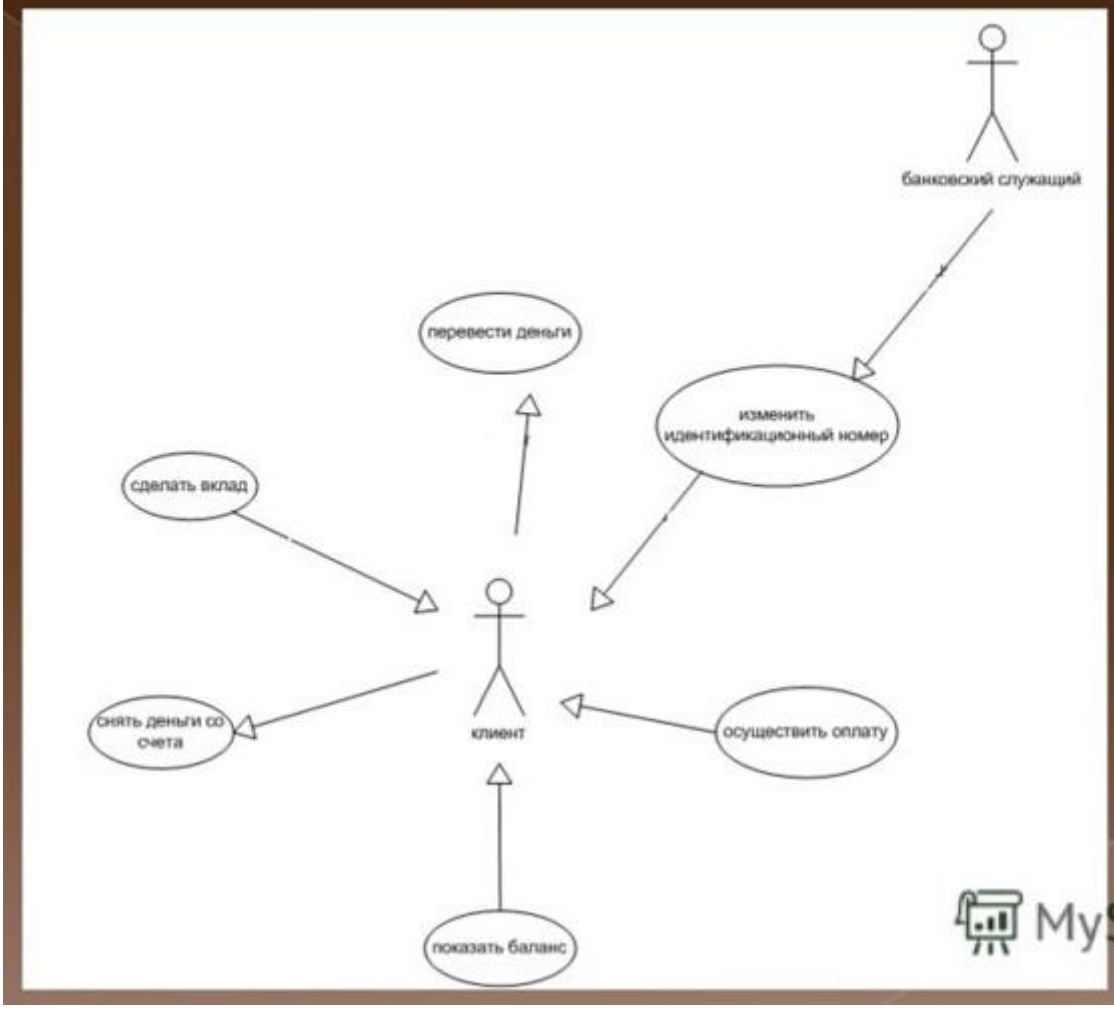
Действующие лица делятся на три основных типа:

Первый тип действующих лиц - это физические личности, или пользователи системы

Вторым типом действующих лиц является другая система

Наиболее распространенный тип действующего лица, третий, - это время

Пример диаграммы вариантов использования для банковского автомата



Конкретная цель диаграмм вариантов использования

документирование вариантов использования, действующих лиц и связей между ними

Правила разработки диаграммы вариантов использования:

Не моделируйте связи между действующими лицами. По определению действующие лица находятся вне сферы действия системы

Не соединяйте стрелкой два варианта использования. Диаграммы данного типа описывают только, какие варианты использования доступны системе, а не порядок их выполнения

Каждый вариант использования должен быть инициирован действующим лицом. Это означает, что всегда должна быть стрелка, начинающаяся на действующем лице и заканчивающаяся на варианте использования

Как обнаружить варианты использования?

1. Прочитать любую документацию заказчика.
2. Рассмотреть области использования системы.
3. Учесть мнение каждого из заинтересованных лиц проекта.
4. Учесть реакцию системы на внешние события.

Как убедиться, что обнаружены все варианты использования?

Для этого следует задать себе вопросы :

1. Присутствует ли каждое функциональное требование хотя бы в одном варианте использования?
2. Учли ли вы, как с системой будет работать каждое заинтересованное лицо?
3. Какую информацию каждое заинтересованное лицо будет передавать системе?
4. Какую информацию каждое заинтересованное лицо будет получать от системы?
5. Учли ли вы проблемы, связанные с эксплуатацией? Кто-то должен будет запускать готовую систему и выключать ее.
6. Учли ли вы все внешние системы, с которыми будет взаимодействовать данная?
7. Какой информацией каждая внешняя система будет обмениваться с данной?

Детали варианта использования, т.е. как будут происходить действия в нем, описывают в документе, называемом «Потоком событий». Этот документ подробно описывает, что будут делать пользователи системы, а что сама система.

Варианты использования не зависят от реализации. Создаваемый набор вариантов использования должен дать пользователям возможность увидеть всю систему целиком. Поэтому вариантов использования должно быть достаточно для того, чтобы полностью описать действия системы. Модель типичной системы состоит из 20 – 50 вариантов использования.

. Названия вариантов использования должны быть деловыми, а не техническими терминами, имеющими значение для заказчика.

Варианты использования обычно называют глаголами или глагольными фразами, описывая при этом, что пользователь видит как конечный результат процесса. Нужно заострить внимание на результате, который потребитель ожидает от системы, а не на действиях, которые надо предпринять для достижения этого результата.