**Назначение языка**

*UML* - унифицированный *язык моделирования*. Из этих трех слов главным является *слово* " **язык** ". Что же такое язык? Не будем изобретать велосипед, а лучше заглянем в *глоссарий*, благо в Интернете их величайшее множество. Сделав это, мы скорее всего обнаружим *определение*, подобное приведенному ниже.

**Язык** - система знаков, служащая:

* средством человеческого общения и мыслительной деятельности;
* способом выражения самосознания личности;
* средством хранения и передачи информации.

Язык включает в себя набор знаков (словарь) и правила их употребления и интерпретации (грамматику).

К этому достаточно исчерпывающему определению нужно добавить, что языки бывают естественные и искусственные, формальные и неформальные. *UML* - язык формальный и искусственный, хотя, как мы увидим далее, этот ярлык к нему не совсем подходит. Искусственный он потому, что у него имеются авторы, о которых мы еще не раз упомянем в дальнейшем (в то же время, развитие *UML* непрерывно продолжается, что ставит его в один ряд с естественными языками). Формальным его можно назвать, поскольку имеются правила его употребления (правда, описание *UML* содержит и явно неформальные элементы, как мы, опять-таки, позже увидим). Еще один нюанс: *UML* - язык графический, что также немного путает ситуацию!

При описании формального искусственного языка, что мы уже видели на примерах описания языков программирования, как правило, описываются такие его элементы, как:

1. *синтаксис*, то есть определение правил построения конструкций языка;
2. *семантика*, то есть определение правил, в соответствии с которыми конструкции языка приобретают смысловое значение;
3. *прагматика*, то есть определение правил использования конструкций языка для достижения нужных нам целей.

Естественно, *UML* включает все эти элементы, хотя, как мы опять-таки увидим далее, в их описании тоже наблюдаются отличия от правил, принятых в языках программирования.

Второе *слово* в фразе, которой расшифровывается аббревиатура *UML* - *слово* " **моделирование** ". Да, *UML* - это *язык моделирования*. Причем *объектно-ориентированного моделирования*. Более подробно о смысле понятия "*моделирование*" мы поговорим чуть позже, а пока отметим, что *слово* это весьма многозначно. В английском языке есть целых два слова - **modeling** и **simulation**, которые оба переводятся как "*моделирование*", хотя означают разные понятия. *Modeling* подразумевает создание модели, лишь описывающей *объект*, а *simulation* предполагает получение с помощью созданной модели некоторой дополнительной информации об объекте. *UML* в первую *очередь* - *язык моделирования* именно в первом смысле, то есть средство построения *описательных моделей*. Как средство симулирования его тоже можно использовать, хотя для этой роли он подходит не так хорошо.

Третье *слово* в названии *UML* - *слово* " **унифицированный** ". Его можно понимать тоже неоднозначно. В литературе можно встретить описание эры "до *UML*" как "войны методов" моделирования, ни один из которых "не дотягивал" до уровня индустриального стандарта. *UML* как раз и стал таким единым универсальным стандартом для *объектно-ориентированного моделирования*, которое во времена его создания как раз "вошло в моду". "Единым" языком моделирования *UML* можно назвать еще и потому, что в его создании, как мы увидим далее, объединились усилия авторов трех наиболее популярных методов моделирования (и не только их).

Подводя итоги, кратко можно сказать, что *UML* - искусственный язык, который имеет некоторые черты естественного языка, и *формальный язык*, который имеет черты неформального. Это звучит не очень понятно, но это действительно так!

**Историческая справка**

Откуда взялся The *UML*? Если говорить коротко, то *UML* вобрал в себя черты нотаций Грейди Буча (Grady Booch), Джима Румбаха (Jim Rumbaugh), Айвара Якобсона (Ivar Jacobson) и многих других.

В не такие уж и далекие 80-е годы было множество различных методологий моделирования. Каждая из них имела свои достоинства и недостатки, а также свою нотацию. То смутное время получило название "войны методов". Проблема в том, что разные люди использовали разные нотации, и для того чтобы понять, что описывает та или иная *диаграмма*, зачастую требовался "переводчик". Один и тот же символ мог означать в разных нотациях абсолютно разные вещи!

К тому же примерно в это же время (начало 80-х) стартовала "объектно-ориентированная эра". Все началось с появлением семейства языков программирования SmallTalk, которые применяли некоторые понятия языка Simula-67, использовавшегося в 60-х годах. Появление объектно-ориентированного подхода в первую *очередь* было обусловлено увеличением сложности задач. *Объектно-ориентированный подход* внес достаточно радикальные изменения в сами принципы создания и функционирования программ, но, в то же время, позволил существенно повысить *производительность* труда программистов, *по*-иному взглянуть на проблемы и методы их решения, сделать программы более компактными и легко расширяемыми. Как результат, языки, первоначально ориентированные на традиционный подход к программированию, получили ряд объектноориентированных расширений. Одной из первых, в середине 80-х, была *фирма* Apple со своим проектом *Object Pascal*. Кроме этого, *объектно-ориентированный подход* породил мощную волну и абсолютно новых программных технологий, вершинами которой стали такие общепризнанные сегодня платформы, как Microsoft .*NET* Framework и *Sun* *Java*.

Но самое главное, что появление *ООП* требовало удобного инструмента для моделирования, единой нотации для описания сложных программных систем. И вот "три амиго", три крупнейших специалиста, три автора наиболее популярных методов решили объединить свои разработки. В 1991-м каждый из "трех амиго" начал с написания книги, в которой изложил свой метод ООАП. Каждая методология была *по*-своему хороша, но каждая имела и недостатки. Так, метод Буча был хорош в проектировании, но слабоват в анализе. *OMT* Румбаха был, наоборот, отличным средством анализа, но плох в проектировании. И наконец, Objectory Якобсона был действительно хорош с точки зрения *user experience*, на который ни метод Буча, ни *OMT* не обращали особого внимания. Основной идеей Objectory было то, что *анализ* должен начинаться с прецедентов, а не с *диаграммы классов*, которые должны быть производными от них.

К 1994-му существовало 72 метода, или частные методики. Многие из них "перекрывались", т. е. использовали похожие идеи, нотации и т. д. Как уже говорилось выше, чувствовалась острая потребность, "социальный заказ" - закончить "войну методов" и объединить в одном унифицированном средстве все лучшее, что было создано в области моделирования.

**Способы использования языка**

И вот Румбах присоединился к Бучу в Rational Inc. Они объединили свои нотации и создали первую версию *UML*. В 1995 году на конференции OOPSLA они представили его как *Unified* *Method*, который потом и получил название *UML*. Чуть позже к ним присоединился Якобсон, который добавил к результатам их труда элементы Objectory и начал работу над Rational *Unified* Process (*RUP*). В 1997 году *UML* был отправлен в *Object* *Management* *Group* (*OMG*) для стандартизации.

Начать хотелось бы с демонстрации известной картинки, которая уже более двух десятилетий "живет" в Интернете. Эта картинка прекрасно иллюстрирует типичный процесс создания продукта, или "решения" (поскольку продукт решает проблему заказчика), как любят говорить в Microsoft ([рис. 1.2](https://www.intuit.ru/studies/courses/1007/229/lecture/5952?page=2#image.1.2)).



Здесь мы видим все проблемы программной инженерии, в частности проблемы с коммуникацией и пониманием, вызванные отсутствием четкой спецификации создаваемого продукта. Так вот, авторы *UML* определяют его как графический *язык моделирования* общего назначения (т. е. его можно применять для проектирования чего угодно - от простой качели, как на рисунке, до сложного аппаратно-программного комплекса или даже космического корабля), предназначенный для **спецификации (модельной)**, **визуализации**, **проектирования** и **документирования** всех артефактов, создаваемых в ходе разработки.

Проектирование: возможен процесс, который часто называют "реверс-инжинирингом", - т. е. создание *UML*-модели из существующего кода приложения.

**Структура определения языка**

Это, наверное, самая короткая часть лекции. Здесь нам хотелось бы рассказать о том, как описан *UML* его авторами. Но прежде нужно поговорить о способах описания искусственных языков вообще (например, языков программирования).

Конечно, вы уже читали книги, в которых описывались языки программирования, и не могли не заметить, как авторы этих книг все время самоотверженно балансируют между точностью и понятностью описания. Велик соблазн описать язык формально точно, но такое описание своей сложностью может отпугнуть потенциального пользователя новой технологии. С другой стороны, "понятное", неформальное описание языка может получиться очень длинным и неполным и просто запутать читателя.

Как же определен *UML*? Довольно часто компиляторы и *IDE* языков программирования написаны с использованием этих же языков (вспомните хотя бы Turbo *Pascal*!). Подобный метод применяется и при описании *UML*. Авторы использовали так называемое четырехуровневое мета-*моделирование*. Первый уровень - это сами данные. Второй - это их модель, т. е., например, описание их в программе. Третий - *метамодель*, т. е. описание языка построения модели. Четвертый - мета-*метамодель*, т. е. описание языка, на котором описана *метамодель*. Для примера - следующий рисунок, позаимствованный из стандарта *UML*, показывает применение этого подхода к простым записям о котировках акций ([рис. 1.4](https://www.intuit.ru/studies/courses/1007/229/lecture/5952?page=3#image.1.4)).

**Терминология и нотация**

Теперь давайте поговорим о нотации. "*Нотация*" - это то, что в других языках называют "синтаксисом". Само *слово* "*нотация*" подчеркивает, что *UML* - язык графический и модели (а точнее диаграммы) не "записывают", а рисуют. Как уже говорилось выше, одна из задач *UML* -

служить средством коммуникации внутри команды и при общении с заказчиком. "В рабочем порядке" диаграммы часто рисуют на бумаге от руки, причем обычно - не слишком аккуратно. Поэтому при выборе элементов нотации основным принципом был отбор значков, которые хорошо смотрелись бы и были бы правильно интерпретированы в любом случае - будь они нарисованы карандашом на салфетке или созданы на компьютере и распечатаны на лазерном принтере.

Вообще же, в *UML* используется четыре вида элементов нотации:

1. фигуры,
2. линии,
3. значки,
4. надписи.

Разберем все *по* порядку. Фигуры используются "плоские" - прямоугольники, эллипсы, ромбы и т. д. Но есть одно *исключение* - как мы увидим далее, на диаграмме развертывания для обозначения узлов инфраструктуры применяется "трехмерное" изображение параллелепипеда. Это единственное *исключение* из правил. Внутри любой фигуры могут помещаться другие элементы нотации.

О линиях стоит сказать лишь то, что своими концами они должны соединяться с фигурами. На *UML* диаграммах вы не встретите линий, нарисованных "сами *по* себе" и не соединяющих фигуры. Применяется два типа линий - сплошная и пунктирная. Линии могут пересекаться, и хотя таких случаев следует *по* возможности избегать, в этом нет ничего страшного.

Вообще же стоит сказать, что *UML* предоставляет исключительную свободу - можно рисовать что угодно и как вздумается, лишь бы можно было понять смысл созданных диаграмм. В изображении фигур и значков тоже нет каких-то жестких требований, и разработчики CASE-средств для *UML*-проектирования вовсю используют эту свободу, применяя различные стили рисования, заливку фигур цветом, тени и т. д. Иногда это смотрится весьма симпатично, а иногда даже раздражает.

Кстати об инструментах рисования. Мы уже упоминали, что такое *ПО* существует, и далее мы рассмотрим этот вопрос более подробно (проведя сравнительные исследования), пока же скажем лишь о нескольких наиболее заметных программах этого класса. К таким пакетам можно отнести:

* IBM Rational Rose;
* Borland Together;
* Gentleware Poseidon;
* Microsoft Visio;
* Telelogic TAU G2.

**Контрольные вопросы**

* Как расшифровывается аббревиатура UML?
* Какая версия UML является текущей?
* Кто были авторами UML?
* Чем НЕ является UML?
* Какие программные средства, поддерживающие UML, вы знаете?
* Используются ли в UML "трехмерные" фигуры?