Моделирование на UML

Лекция 1

Определение UML

Что такое UML?

UML является аббревиатурой полного названия Unified Modeling Language. Правильный перевод этого названия на русский язык – унифицированный язык моделирования.

Таким образом, обсуждаемый предмет характеризуется тремя словами, каждое из которых является точным термином.

UML – это язык

Главным словом в этом сочетании является слово "язык".

Язык – это знаковая система для хранения и передачи информации.

- *формальные*, правила употребления которых строго и явно определены,
- *неформальные*, употребление которых основано на сложившейся практике
- *естественные*, появляющиеся как бы сами собой в результате неперсонифицированных усилий массы людей,
- *искусственные*, являющиеся плодом видимых усилий определенных лиц.

UML можно охарактеризовать как **формальный искусственный язык**. Признаком искусственности служит наличие трех общепризнанных авторов – Гради Буча, Ивара Якобсона и Джеймса Рамбо.

UML – это язык моделирования

Слово "моделирование“ имеет множество смысловых оттенков и сложившихся способов употребления. В частности, английские слова modeling и simulation оба переводятся словом "моделирование", хотя означают разные вещи. В первом случае речь идет о составлении модели, которая используется только для описания моделируемого объекта или явления. Во втором случае подразумевается составление модели, которая может быть использована для получения существенной информации о моделируемом объекте или явлении.

В отношении разработки программного обеспечения так сложилось, что результаты фаз анализа и проектирования, оформленные средствами определенного языка, принято называть *моделью*. Деятельность по составлению моделей естественно назвать *моделированием*. Именно в этом смысле UML является языком моделирования.

Таким образом, **модель UML – это, прежде всего, описание объекта или явления**, а также и кое-что другое, а именно все, что авторам UML удалось включить в язык, не нарушая принципа унификации.

UML – это унифицированный язык моделирования

Авторы языка характеризуют эпоху до UML как период "войны методов". UML является не первым языком моделирования. К моменту его появления насчитывались десятки других, различающихся системой обозначений, степенью универсальности, способами применения и т.д. Авторы языков и теоретики программирования препирались между собой, выясняя, чей подход лучше, а разработчики всю эту "войну методов" равнодушно игнорировали, поскольку ни один из методов не дотягивал до уровня индустриального стандарта.

Толчком к изменению ситуации послужило следующее. Во-первых, массовое распространение получил *объектно-ориентированный подход* к разработке программных систем, в результате чего возникла потребность в соответствующих средствах. Другими словами, появления чего-то подобного UML с нетерпением ждали практики. Во-вторых, три крупнейших специалиста в этой области, авторы наиболее популярных методов, решились объединить усилия именно с целью унификации своих разработок в соответствии с социальным заказом.

Авторы UML при поддержке и содействии всей международной программистской общественности смогли свести воедино (унифицировать) большую часть того, что было известно и до них. В результате унификации получилась теоретически изящная и практически полезная вещь – UML.

Если попытаться проследить историю возникновения и развития элементов UML, как на уровне основополагающих идей, так и на уровне технических деталей, то пришлось бы назвать сотни имен и десятки организаций. Мы не будем этого делать, достаточно привести картинку, иллюстрирующую историю развития UML.

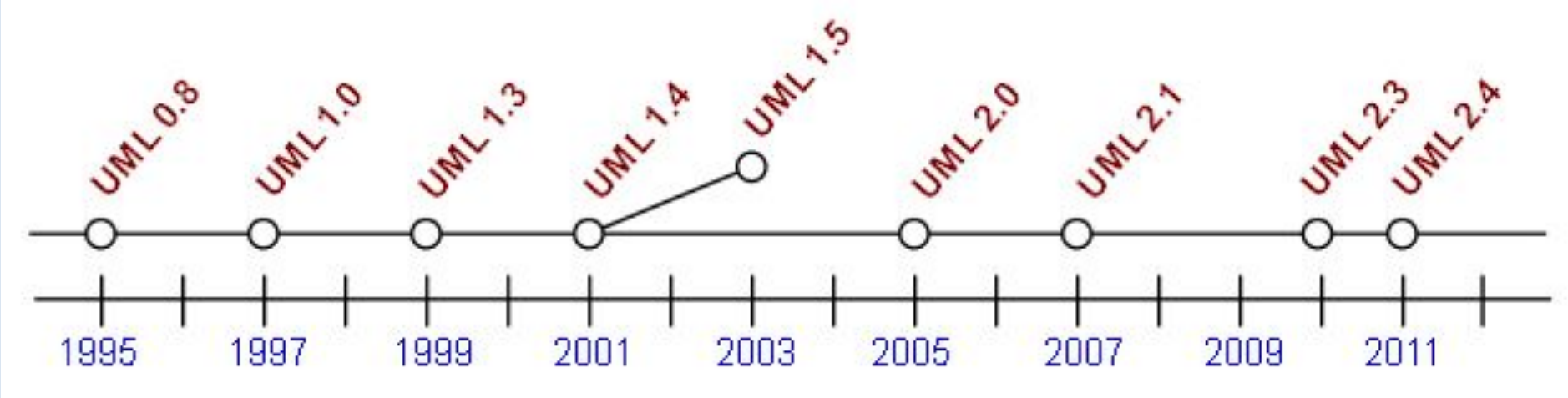


Рис. **История развития UML**

Назначение UML

UML предназначен для моделирования. Сами авторы UML определяют свое детище следующим образом.

Язык UML – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем.

Спецификация

В типичных случаях в процессе разработки приложений участвуют по меньше мере два действующих лица: *заказчик* (конкретный человек или группа лиц, или организация) и *разработчик* (это может быть программист-одиночка, временная команда проекта или целая организация, специализирующаяся на разработке программного обеспечения). Из-за того, что действующих лиц двое, очень многое зависит от степени их взаимопонимания.

Одним из ключевых этапов разработки приложения является определение того, каким требованиям должно удовлетворять разрабатываемое приложение. В результате этого этапа появляется формальный или неформальный документ (артефакт), который называют по-разному, имея в виду примерно одно и то же: постановка задачи, требования, техническое задание, внешние спецификации и др.

Спецификация – это декларативное описание того, как нечто устроено или работает.

Необходимо принимать во внимание три толкования спецификаций.

- То, которое имеет в виду действующее лицо, являющееся источником спецификации (например, заказчик).
- То, которое имеет в виду действующее лицо, являющееся потребителем спецификации (например, разработчик).
- То, которое объективно обусловлено природой специфицируемого объекта.

Эти три трактовки спецификаций могут не совпадать, и сплошь и рядом не совпадают, причем значительно. Заказчик может не осознавать своих объективных потребностей, или неверно их интерпретировать. Разработчик может не разбираться в предметной области заказчика и интерпретировать формулировки спецификаций совершенно превратным образом. Если же в формулировке спецификаций участвует разработчик, то злоупотребление технической терминологией может совершенно дезориентировать заказчика.

Основное назначение UML – предоставить, с одной стороны, достаточно формальное, с другой стороны, достаточно удобное, и, с третьей стороны, достаточно универсальное средство, позволяющее до некоторой степени снизить риск

Визуализация

Модели UML допускают представление в форме картинок, причем эти картинки наглядны, интуитивно понятны, практически однозначно интерпретируются и легко составляются.

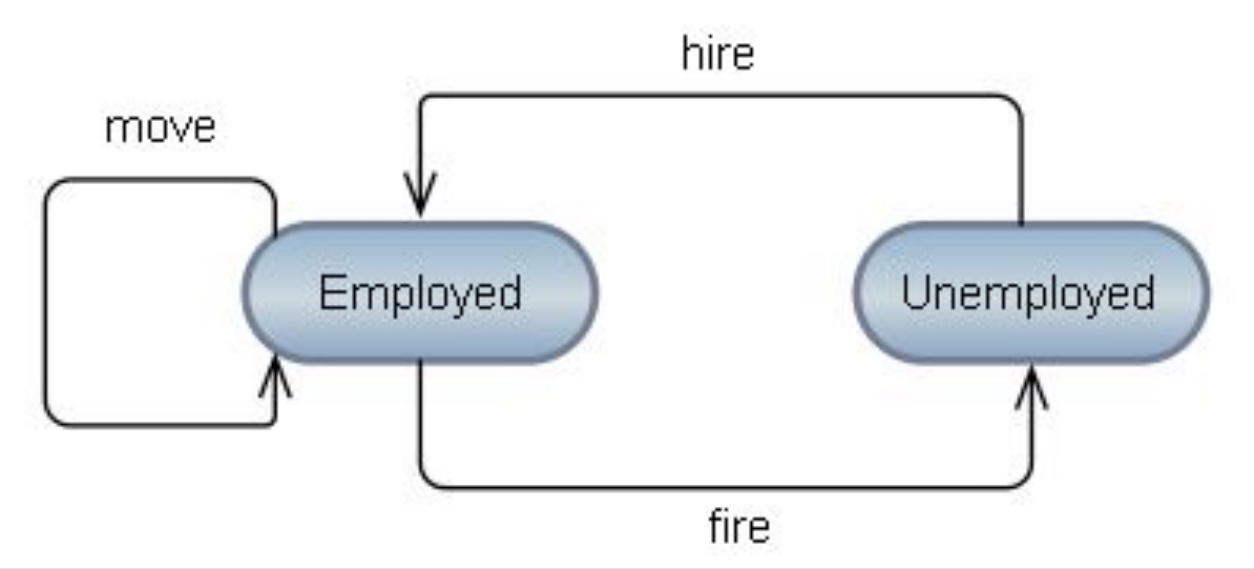


Рис. **Жизненный цикл работника на предприятии**

Второе по важности назначение UML состоит в том, чтобы служить адекватным средством коммуникации между людьми. Графическое представление модели UML не тождественно самой модели. Это важное обстоятельство часто упускается из виду при первом знакомстве с UML.

Проектирование

В оригинале данное назначение UML определено с помощью слова construct, которое мы передаем термином "проектирование". Речь идет о том, что UML предназначен не только для описания абстрактных моделей приложений, но и для непосредственного манипулирования артефактами, входящими в состав этих приложений, в том числе такими, как программный код. Другими словами, одним из назначений UML является создание таких моделей, для которых возможна **автоматическая генерация программного кода** (или фрагментов кода) соответствующих приложений. Более того, природа моделей UML такова, что возможен и обратный процесс: автоматическое построение модели по коду готового приложения.

Инструменты, поддерживающие UML, все время совершенствуются, так что в перспективе третье предназначение UML может выйти и на первое место.

Некоторым уставшим от бесконечной отладки разработчикам может показаться, что стоит изучить UML, и все проблемы программирования будут решены. К сожалению, это не так.

Документирование

Модели UML являются артефактами, которые можно хранить и использовать как в форме электронных документов, так и в виде твердой копии. В последних версиях UML с целью достижения более полного соответствия этому назначению сделано довольно много. В частности, специфицировано представление моделей UML в форме документов в формате XMI. Другими словами, модели UML не являются вещью в себе, которой можно только любоваться – это документы, которые можно использовать самыми разными способами, начиная с печати картинок и заканчивая автоматической генерацией человекочитаемых текстовых описаний.

- XMI (XML Metadata Interchange) – внешний формат данных, основанный на языке XML (схема и набор правил использования тэгов), предназначенное для сериализации моделей и обмена ими.

Чем НЕ является UML

Во-первых, **UML не является языком программирования**. Дело не в том, что UML язык графический, а подавляющее большинство практических языков программирования являются текстовыми языками. Гораздо важнее то, что для моделей UML не определена *операционная семантика*, то есть, не определен способ выполнения моделей на компьютере.

Во-вторых, **UML не является спецификацией инструмента** (хотя инструменты имеются, например, Magic Draw, Rational Rose Enterprise, Visual Paradigm, Enterprise Architect, StarUML и др.). Сам язык никоим образом не навязывает то, как его нужно поддерживать инструментальными средствами. Решение всех вопросов, связанных с реализацией UML на компьютере полностью отдано на откуп разработчикам инструментов.

В-третьих, **UML не является моделью процесса разработки приложений** (хотя модель процесса разработки необходима и имеется множество различных моделей, предложенных разными авторами). Конечно, у авторов UML есть собственная модель процесса – Rational Unified Process (RUP), которую они не могли не иметь в голове, разрабатывая язык, но, тем не менее, ими сделано все для того, чтобы устранить прямое влияние RUP на UML и сделать UML пригодным для использования в любой модели процесса или даже без оной.

Способы использования UML

UML предназначен для решения различных задач, соответственно он может быть использован и практически используется по-разному.

Способы использования UML:

Рисование картинок. Графические средства UML можно и нужно использовать безотносительно ко всему остальному. Даже рисование диаграмм карандашом на бумаге позволяет упорядочить мысли и зафиксировать для себя существенную информацию о моделируемом приложении или иной системе.

Обмен информацией. Сообщество людей, применяющих и понимающих UML, стремительно растет. Если вы будете использовать UML, то вас будут понимать другие, и вы будете понимать других "с полувзгляда".

Спецификация систем. Это важнейший способ использования UML. И хотя не во всех случаях UML оказывается абсолютно адекватным средством спецификации, мы надеемся, что по мере развития языка все меньше будет оставаться таких исключений, где UML неприменим.

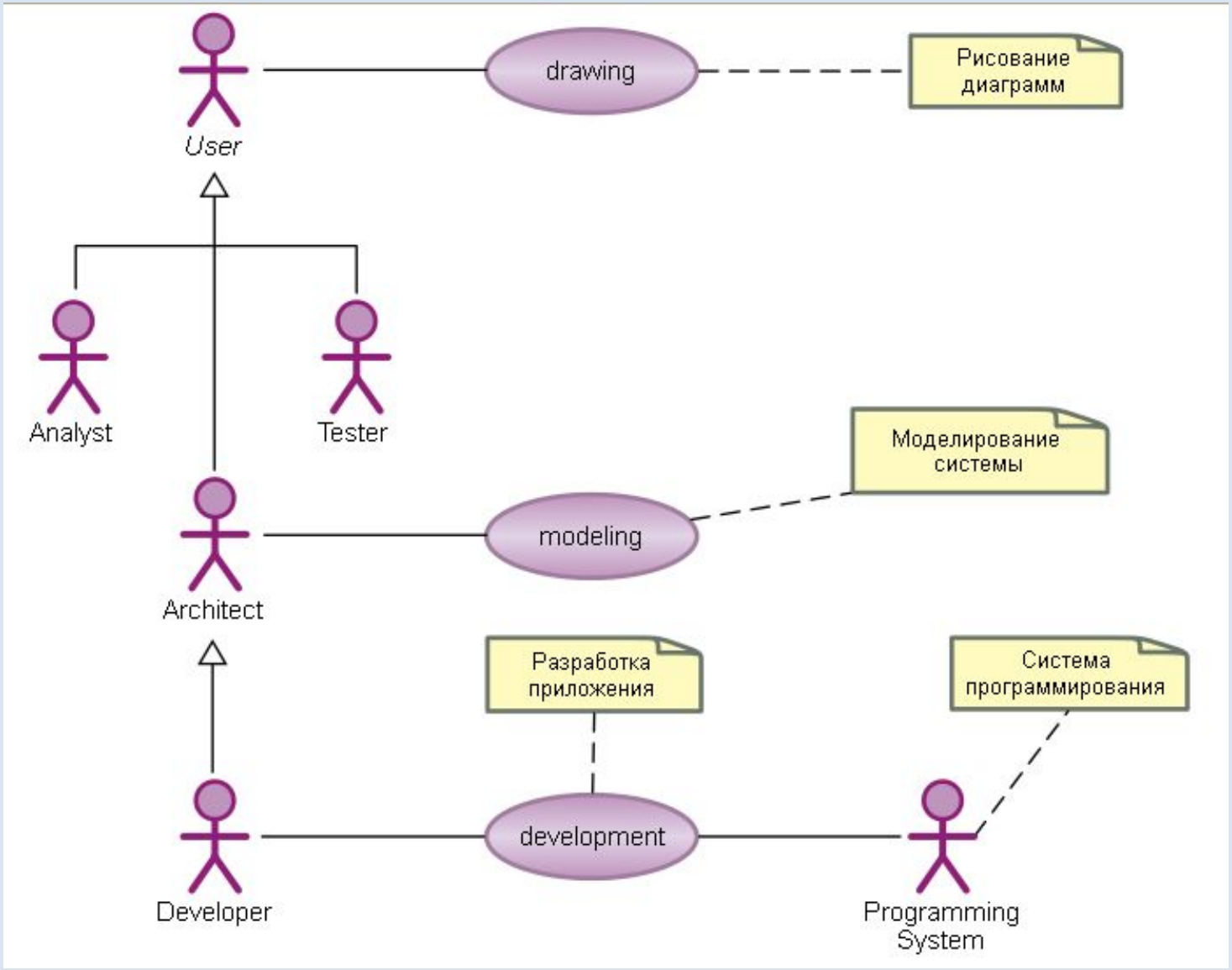
Повторное использование архитектурных решений. Повторное использование ранее разработанных решений – ключ к повышению эффективности. К сожалению, модели UML пока что повторно используются в весьма ограниченных масштабах.

Генерация кода. Генерировать код нужно и можно, но возможности имеющихся инструментов не стоит переоценивать.

Имитационное моделирование. Возможности построения моделей UML, из которых путем вычислительных экспериментов можно было бы извлекать информацию о моделируемом объекте, пока что уступают возможностям специализированных систем, сконструированных для этих целей.

Верификация моделей. Было бы замечательно, если бы по модели можно было бы делать формальные заключения об ее свойствах: модель непротиворечива, согласована, эффективна и т. п. Кое-что UML позволяет проверить, но, конечно, очень мало. Здесь уместно привести аналогию с традиционными системами программирования: они позволяют быстро и надежно избавиться от синтаксических ошибок, но с логическими ошибками дело обстоит гораздо хуже. Может быть, в будущем...

Инструментальная поддержка



Рассмотрим, как соотносится сегодняшняя практика использования UML с деклари-рованным назначением языка.

Можно выделить три основных варианта использования UML.

Рис. **Инструментальная поддержка**

Вариант использования drawing ("Рисование диаграмм") подразумевает изображение диаграмм UML с целью обдумывания, обмена идеями между людьми, документирования и тому подобного. Значимым для пользователя User результатом в этом случае является само изображение диаграмм. Вообще говоря, в этом варианте использования языка поддерживающий инструмент не очень нужен. Иногда рисование диаграмм от руки фломастером с последующим фотографированием цифровым аппаратом может оказаться практичнее.

Вариант использования modeling ("Моделирование систем") подразумевает создание и изменение модели системы в терминах тех элементов моделирования, которые предусматриваются метамоделью UML. Значимым результатом в этом случае является машинно-читаемый артефакт с описанием модели. Мы будем для краткости называть такой артефакт просто моделью, деятельность по составлению модели называть моделированием, а субъекта моделирования называть архитектором Architect.

Вариант использования development ("Разработка приложений") подразумевает детальное моделирование, реализацию и тестирование приложения в терминах UML. Значимым для пользователя Developer результатом в этом случае является работающее приложение, которое может быть скомпилировано в язык, поддерживаемый конкретной системой программирования Programming System или сразу интерпретировано средой выполнения инструмента. Этот вариант использования наиболее сложен в реализации.

Современные инструменты поддерживают указанные варианты использования далеко не в равной степени. Все инструменты умеют (плохо или хорошо) визуализировать все типы диаграмм UML, некоторые инструменты позволяют построить модель, допускающую какое-то дальнейшее использование, но только немногие инструменты могут генерировать исполняемый код и то, отнюдь, не для всех диаграмм. Имеется множество практических и организационных причин, по которым указанные выше варианты использования неравноправны и в разной степени поддержаны в современных инструментах.

Метод определения UML

В основу описания UML положен метод раскрутки, то есть использование определяемого языка для определения этого языка. А именно, основные конструкции UML формально определены с помощью UML. Конечно, с чего-то раскрутку нужно начать, и это описано в UML неформально, с помощью текстов на естественном (английском) языке.

В описании UML используются три языковых уровня.

- *Мета-метамодель*, то есть описание языка, на котором описана метамодель.
- *Метамодель*, то есть описание языка, на котором описываются модели.
- *Модель*, то есть описание самой моделируемой предметной области.

Весь текст описания UML каждой версии находится в свободно распространяемых документах, доступных по адресу www.omg.org.

Структура определения языка

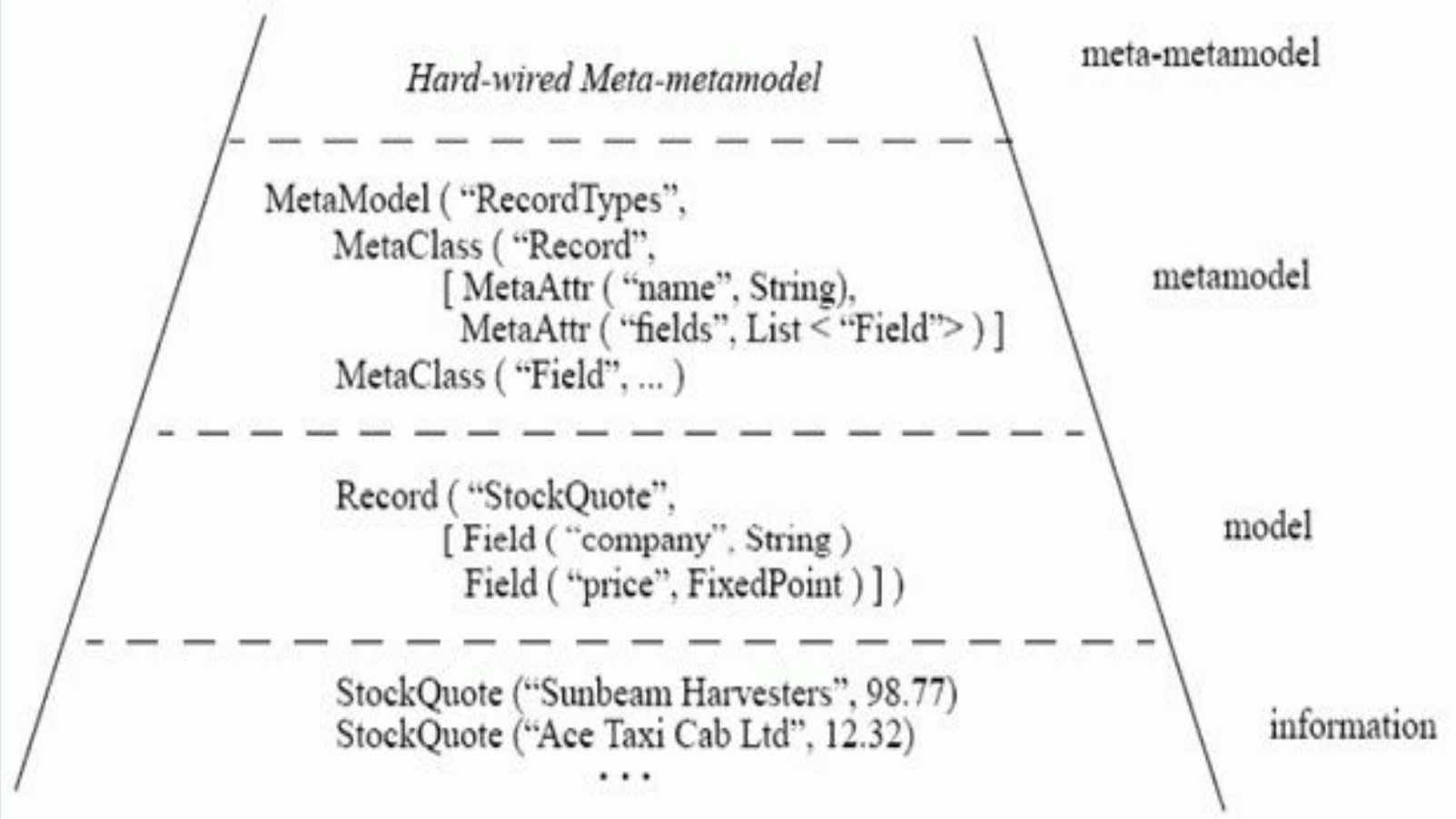
четырехуровневое мета-моделирование.

Первый уровень - сами данные.

Второй - это их модель, т. е. описание их в программе.

Третий - метамодель, т. е. описание языка построения модели.

Четвертый - мета-метамодель, т. е. описание языка, на котором описана метамодель.



Уровни визуального проектирования

При визуальном моделировании программного обеспечения используются следующие уровни абстракции:

- *предметная область;*
- *модель;*
- *метамодель;*
- *метаметамодель.*

Предметная область (domain):

- *тот фрагмент действительности, куда создаваемое ПО будет встроено:*

  - бизнес-процессы компании, для которой создается информационная система,

  -электромеханическая среда для встроенного ПО и т. д.;

- *архитектурные решения ПО, которые должны быть тщательно проработаны, обсуждены с разными специалистами и ими понятны; с этой целью они и подвергаются визуализации.*

Модель (model) *- это упрощенное описание предметной области, созданное для удобства выполнения там действий, работы.*

При визуальном моделировании ПО строятся следующие модели:

модели анализа (analysis models), формализующие результаты изучения программистами того контекста, где будет работать их будущее ПО;

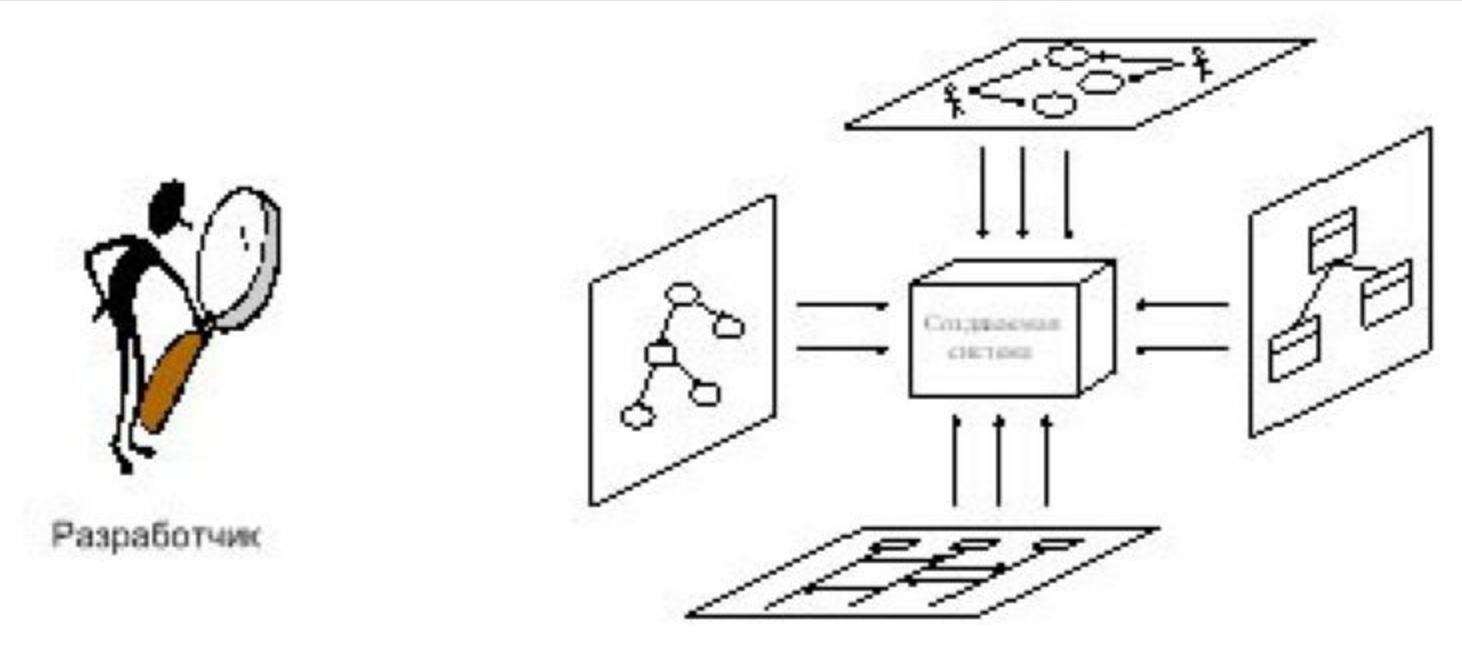
модели проектирования (design models), в которых фиксируются архитектурные решения будущего ПО - его структура, внешние и внутренние интерфейсы, принципиальные вопросы реализации с учетом средств разработки, платформ исполнения и т.д.

Модели анализа должны "плавно" переходить в модели проектирования.

Метамодель (metamodel) *- специальный язык, который существенно упрощает разработку моделей, содержащий описание всех абстракций, которые нужны при моделировании.*

Метаметамодель (meta-metamodel) *- язык описания языков (метамоделей).*

Множество моделей ПО.

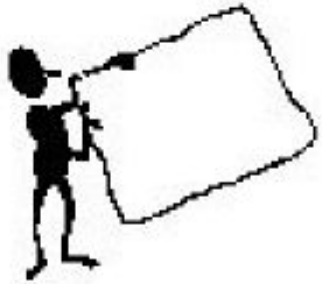


Анализ: ПО - реализует определенную бизнес-функциональность, нужную заказчику.

Проектирование: ПО - принципы реализации ПО.

Тестирование: ПО - черный ящик, реализующий некоторый набор пользовательской функциональности.

Развертка: ПО - набор файлов, хранилищ данных и т. д.



Продавец

Менеджер

Заказчик

Разработчик

Над одним проектом работают:

программисты,

инженеры,

тестеры,

менеджеры,

заказчик,

спонсоры,

бизнес-аналитики,

пользователи,

продавцы-маркетологи,

технические писатели

и многие другие.

Разные виды деятельности при разработке ПО и разные категории специалистов, задействованные в программном проекте, - все это приводит к созданию и использованию различных моделей, выполненных с разных точек зрения.

Точка зрения моделирования (viewpoint) *- это определенный взгляд на систему, который осуществляется для выполнения какой-то определенной задачи кем-либо из участников проекта.*

Правильно выбранная и ясно сформулированная точка зрения на систему, которая не "плывет" при моделировании, - это один из основных критериев того, что модель действительно принесет пользу.

Характеристики точки зрения моделирования:

-- цель моделирования (зачем создается модель)

-- целевая аудитория (для кого предназначается модель)

Терминология и нотация

В **качестве основных графических элементов были выбраны такие, которые было бы легко использовать во всех случаях**.

Типов элементов нотации пять:

- фигура (shape);
- линия (line);
- значок (icon);
- текст (text);
- рамка (frame).

Модель и ее элементы

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

Рассматривая модель UML с наиболее общих позиций, можно сказать, что это граф, в котором вершины и ребра нагружены дополнительной информацией и могут иметь сложную внутреннюю структуру. **Вершины этого графа называются сущностями, а ребра – отношениями**.

Сущности

Для удобства обзора сущности в UML можно подразделить на четыре группы:

- структурные;
- поведенческие;
- группирующие;
- аннотационные.

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

Объект **(object) 1 – сущность, обладающая уникальностью и инкапсулирующая в себе состояние и поведение.**

- ***Класс*** **(class) 2 – описание множества объектов с общими атрибутами, определяющими состояние, и операциями, определяющими поведение.**
- ***Интерфейс*** **(interface) 3 – именованное множество операций, определяющее набор услуг, которые могут быть запрошены потребителем и предоставлены поставщиком услуг.**
- ***Кооперация*** **(collaboration) 4 – совокупность объектов, которые взаимодействуют для достижения некоторой цели.**
- ***Действующее лицо*** **(actor) 5 – сущность,**

- ***Компонент*** **(component) 6 – модульная часть системы с четко определенным набором требуемых и предоставляемых интерфейсов.**
- ***Артефакт*** **(artifact) 7 – 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) 8 – вычислительный ресурс, на котором размещаются и при необходимости выполняются артефакты.**

На следующем рисунке приведена стандартная нотация в минимальном варианте для структурных сущностей.

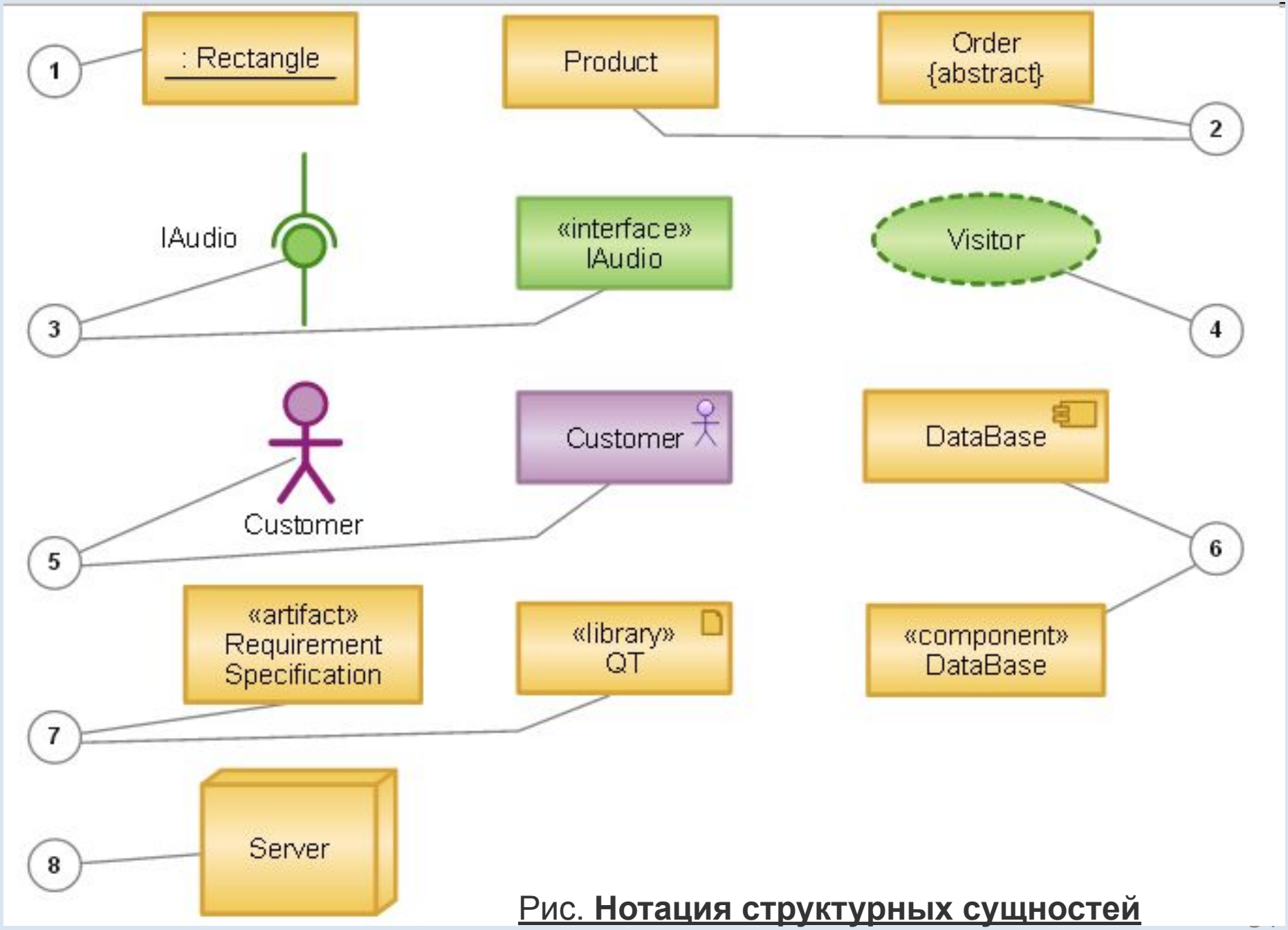


Рис. **Нотация структурных сущностей**

Поведенческие сущности предназначены для описания поведения.

Основных поведенческих сущностей всего две: состояние(1) и действие(3). Сущность «деятельность»(2) можно рассматривать как особый случай состояния.

Несколько особняком стоит сущность – вариант использования (4).

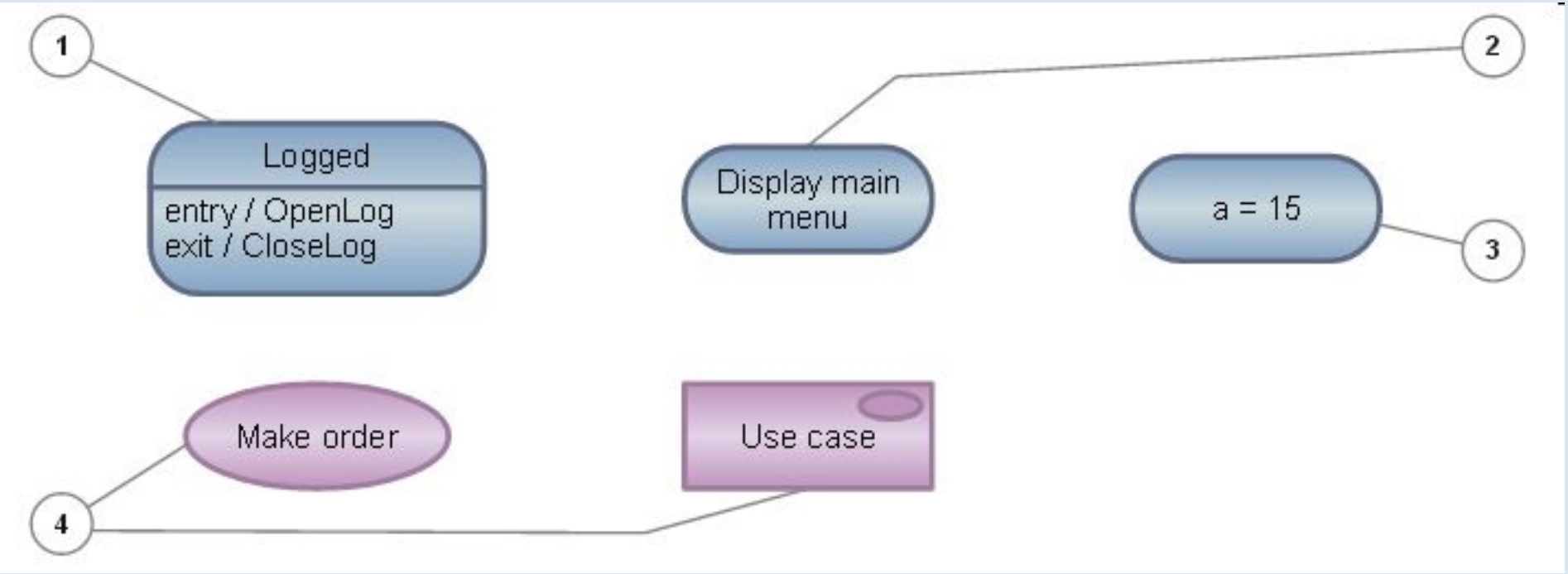


Рис. **Нотация поведенческих сущностей**

Группирующая сущность в UML одна – пакет – зато универсальная.
Пакет **(package) 1 – группа элементов модели (в том числе пакетов).**

Аннотационная сущность тоже одна – комментарий.
Комментарий **(comment) 2 – произвольное по формату и содержанию описание одного или нескольких элементов модели.**

Рис. **Нотация группирующей и аннотационной сущностей**

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.***

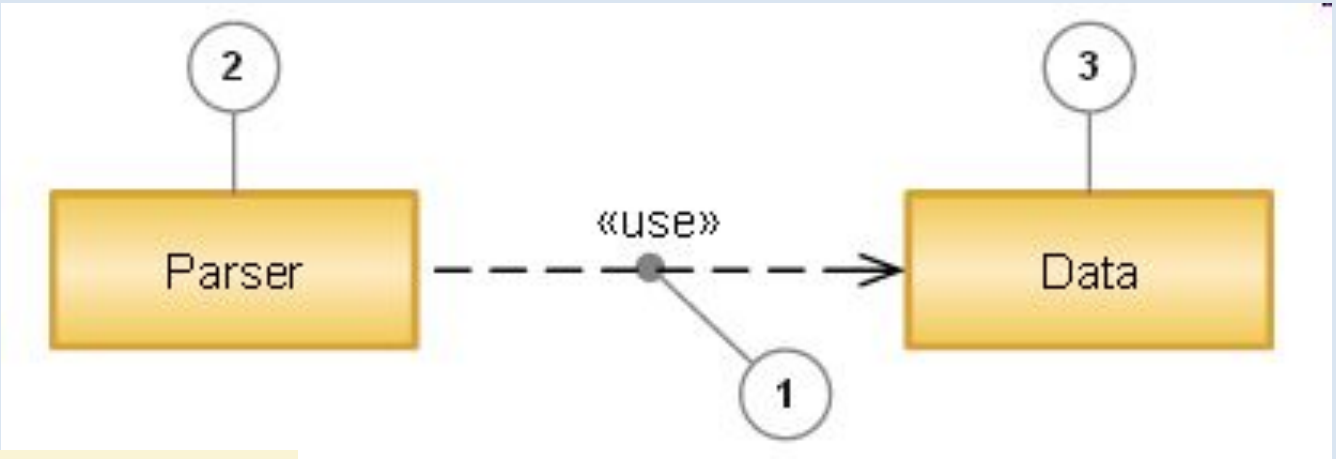


Рис. **Отношение зависимости**

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 (1) с различными дополнениями, соединяющей связанные сущности.

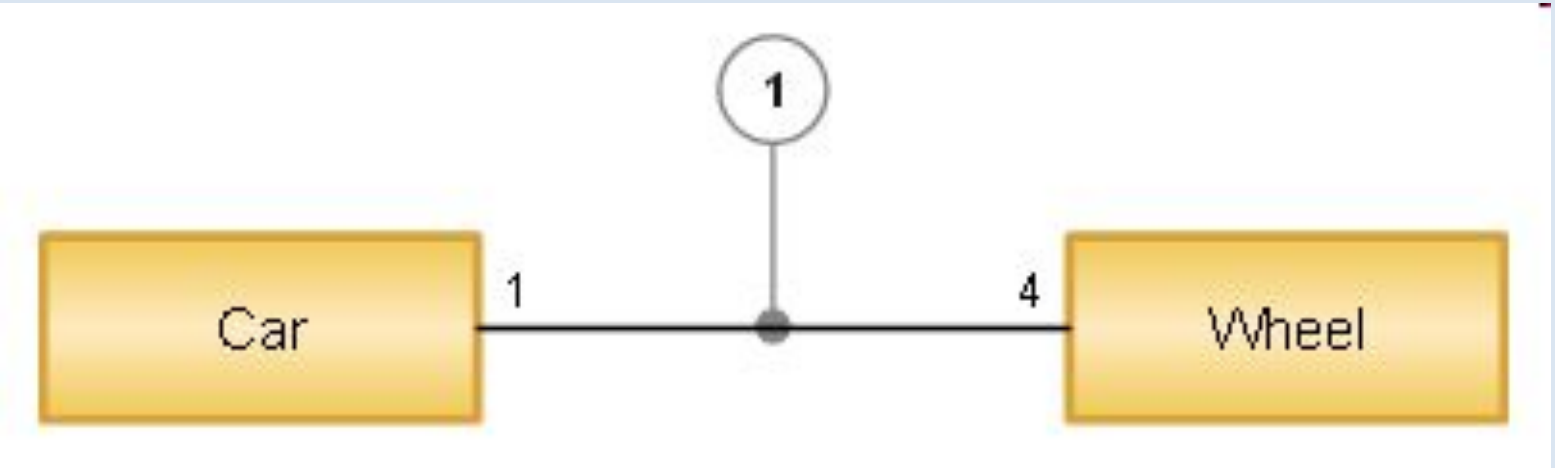


Рис. **Отношение ассоциации**

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).

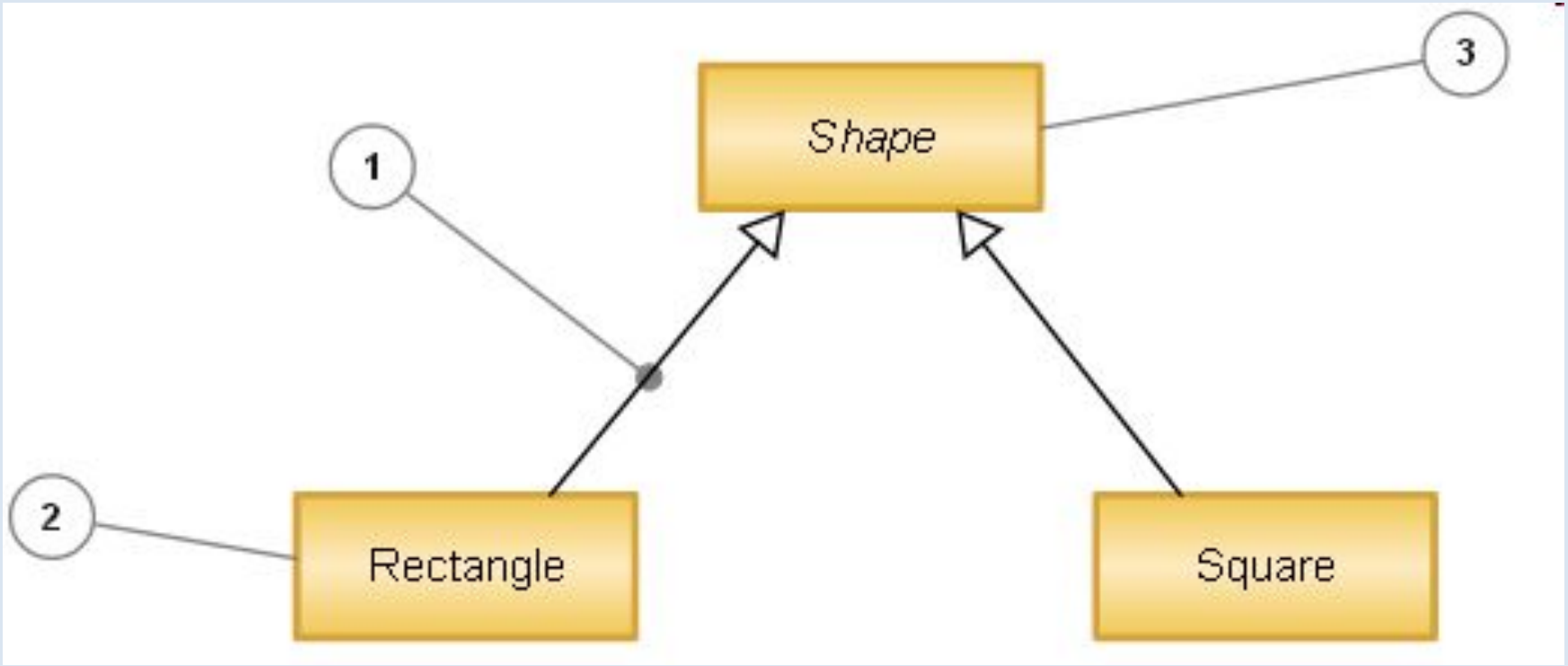


Рис. **Отношение обобщения**

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 (1), направленной от реализующей сущности (2) к реализуемой (3).

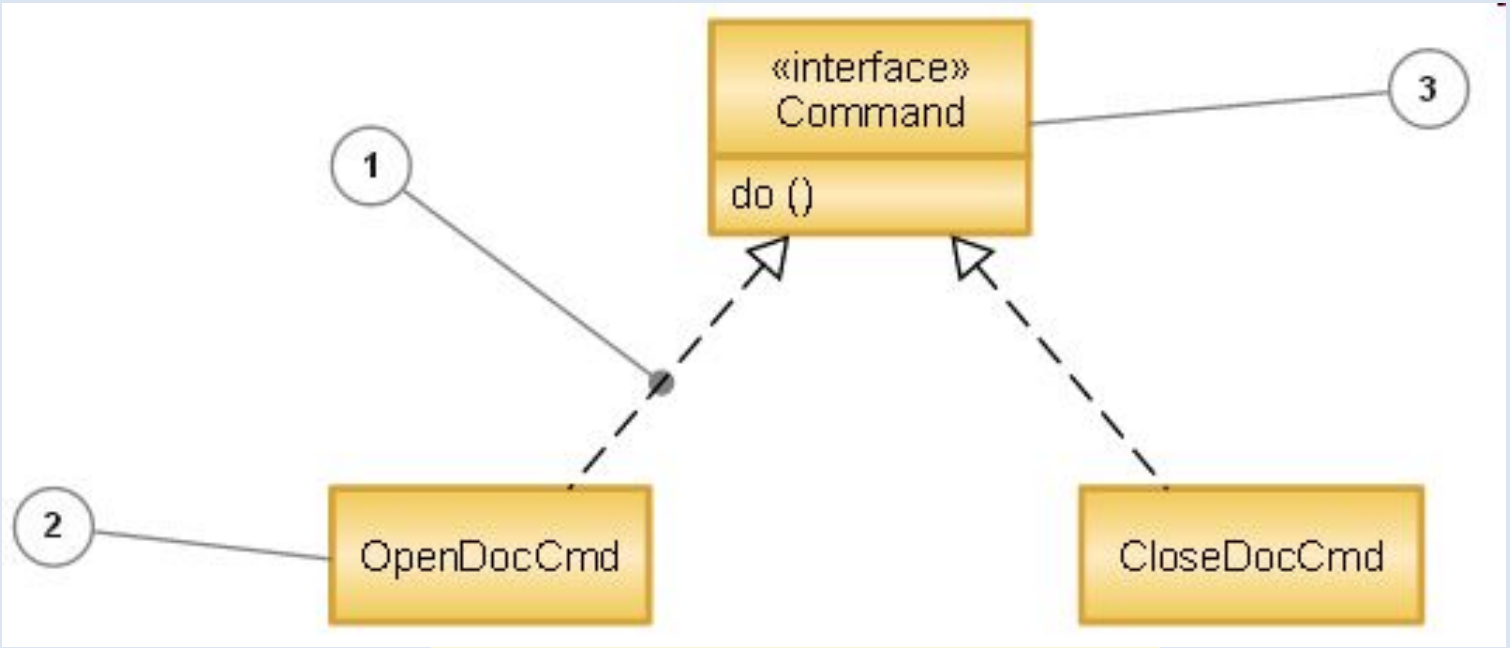


Рис. **Отношение реализации**