РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

Тема 1. Принципы объектно-ориентированного моделирования

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 (точнее, нагруженный мульти-псевдо-гипер-орграф), 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, как нетрудно догадаться, 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, находящаяся вне моделируемой системы и непосредственно взаимодействующая с ней.
- 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.

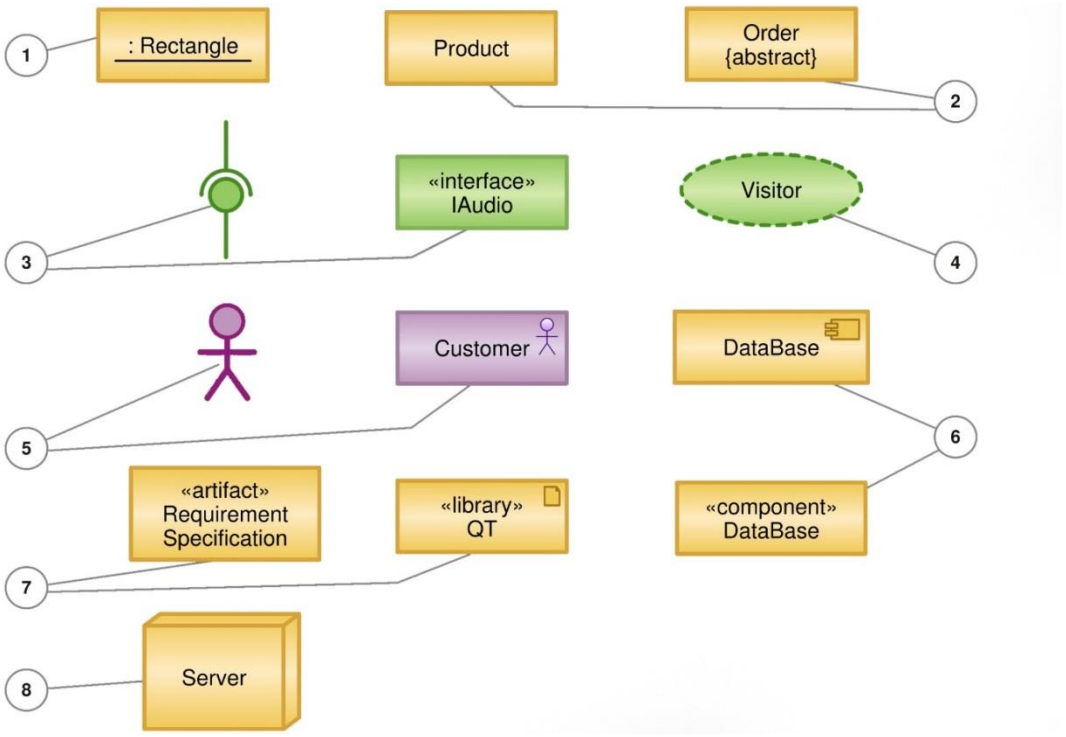

Рис. **Нотация структурных сущностей**

Поведенческие сущности предназначены для описания поведения. Основных поведенческих сущностей всего две: состояние и действие (точнее, две с половиной, потому что иногда употребляется еще и деятельность, которую можно рассматривать как особый случай состояния).

- Состояние (state) (1) – период в жизненном цикле объекта, находясь в котором объект удовлетворяет некоторому условию и осуществляет собственную деятельность или ожидает наступления некоторого события.
- Деятельность (activity) (2) можно считать частным случаем состояния, который характеризуется продолжительными (по времени) не атомарными вычислениями.
- Действие (action) (3)– примитивное атомарное вычисление.

Это только надводная часть айсберга поведенческих сущностей: 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.

- Вариант использования (use case) (4) – множество сценариев, объединенных по некоторому критерию и описывающих последовательности производимых системой действий, доставляющих значимый для некоторого действующего лица результат.

Ниже приведена стандартная нотация в минимальном варианте для поведенческих сущностей.

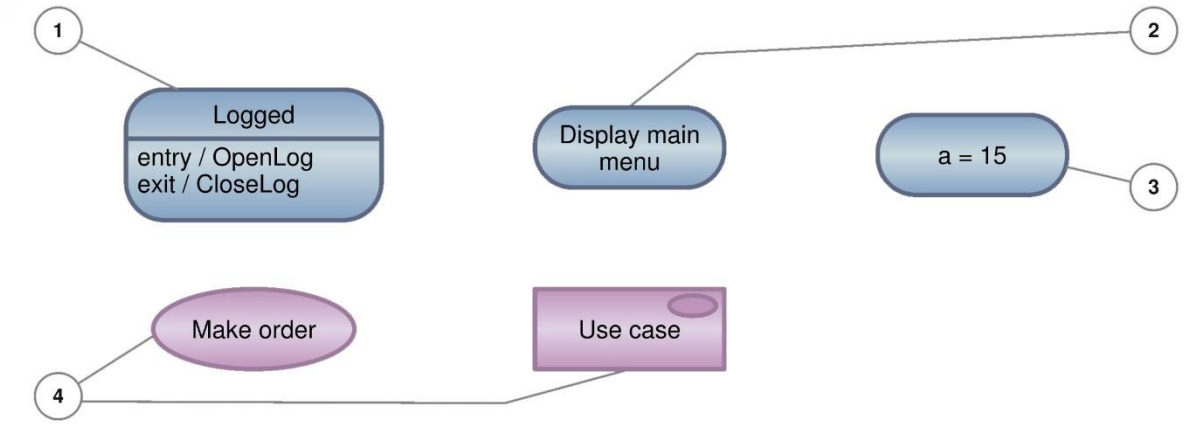

Рис. **Нотация поведенческих сущностей**

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**

Приведенная классификация не является исчерпывающей. У каждой из этих сущностей есть различные частные случаи и вариации.

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 (1), направленной от зависимой сущности (2) к независимой (3), как показано на следующем рисунке. 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.

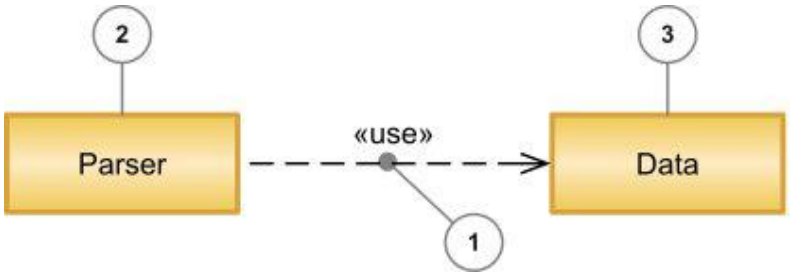

Рис. **Отношение зависимости**

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, как показано на следующем рисунке. 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.

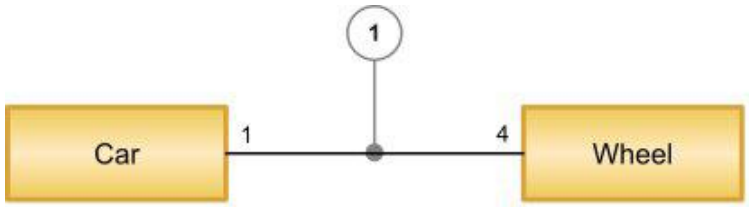

Рис. **Отношение ассоциации**

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), как показано на следующем рисунке.

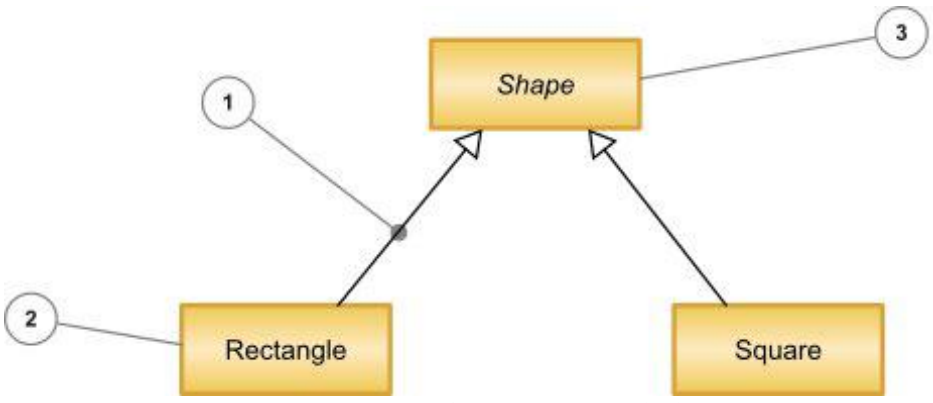

Рис. **Отношение обобщения**

Отношение реализациии 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), как показано на следующем рисунке.

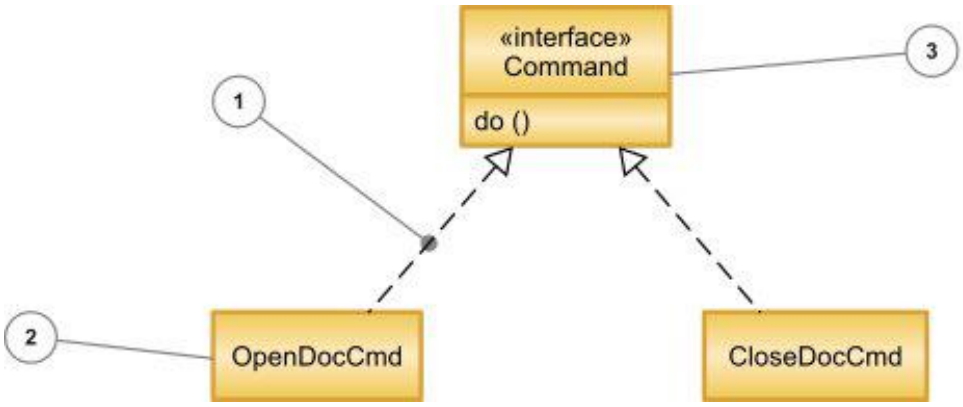

Рис. **Отношение реализации**

Перечисленные типы отношений являются основными.

Диаграммы

Предыдущий параграф преследует примерно те же цели, какие имеет описание лексики в обычном языке программирования. А именно, мы обрисовали (еще не полностью, но уже достаточно) множество слов UML (лексем, графических примитивов, элементов моделирования – называйте, как хотите – фиксированного термина нет). Пора переходить к синтаксису, а именно к описанию того, как из слов конструируются предложения.

На первый взгляд, все очень просто: берутся сущности и, если нужно, указываются отношения между ними. В результате получается модель, то есть граф (с разнородными вершинами и ребрами), нагруженный дополнительной информацией. Но при более внимательном рассмотрении обнаруживаются проблемы. Мы хотим затратить некоторые усилия на обсуждение этих проблем, иначе целый ряд особенностей UML может показаться висящим в воздухе, хотя на самом деле, эти особенности и есть продуманное решение замалчиваемых обычно проблем.

Рассмотрим следующую аналогию с естественным языком. Каждая тройка «сущность» — «отношение» – «сущность» в модели вполне может рассматриваться как простое утверждение: 2 < 5, ртуть тяжелее железа. Пока все хорошо, но вспомним, что в графе (в модели) никакой упорядочивающей структуры нет: нельзя сказать, что это вершина первая, а это – вторая. Продолжая нашу аналогию, получается, что модель – это множество несвязанных между собой предложений, никак не упорядоченное, некий "поток сознания", не в обиду современной литературе будь сказано. Если взять какой-нибудь "нормальный" текст", претендующий на внятное описание чего-либо, то можно заметить, что помимо структуры предложений, имеется масса дополнительных структур: предложения объединены в абзацы, абзацы собраны в параграфы и главы, у которых есть заголовки, помимо обычных абзацев и заголовков есть примечания и сноски. Текст, в котором нет этих структур, понять очень трудно[vv]. Отсюда вывод: помимо сущностей и

отношений, в модели должна быть какая-то структура, которая бы помогала ее составлению и пониманию.

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

- Диаграмма (diagram) – это графическое представление некоторой части графа модели.

Вообще говоря, в диаграмму можно было бы включить любые (допустимые) комбинации сущностей и отношений, но произвол в этом вопросе затруднил бы понимание моделей. Поэтому авторы UML определили набор рекомендуемых к использованию типов диаграмм, которые получили название *канонических* типов диаграмм.

Инструменты моделирования, как правило, обеспечивают работу со всеми каноническими диаграммами, но делают это довольно догматически, не позволяя отойти от канона ни на шаг, даже если это нужно по существу задачи. С другой стороны, некоторые инструменты, наряду с каноническими, поддерживают и апокрифические типы диаграмм. Было бы удобно, если бы набор канонических диаграмм предлагался по умолчанию, но пользователь мог бы настроить, изменить и переопределить этот набор в случае необходимости, примерно так, как это делается с шаблонами Microsoft Word. Некоторые инструменты, но далеко не все, поддерживают такие возможности.

Заметим, что помимо сущностей и отношений на диаграмме присутствует другие элементы модели, которые мы также будем называть *конструкциями* языка. Это тексты, которые могут быть написаны внутри фигур сущностей или рядом с линиями отношений, рамки диаграмм и их фрагментов, значки, присоединяемые к линиям или помещаемые внутрь фигур. Эти элементы не только помогают представить модель в более наглядной форме, но подчас несут значительную смысловую нагрузку.

Классификация диаграмм

В UML 1 всего определено 9 канонических типов диаграмм.

- Диаграмма 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)

Этот список является итогом многочисленных дискуссий и компромиссов, поэтому не следует воспринимать его как догму. В частности, расхожее утверждение "в UML определены девять типов диаграмм" является не совсем верным: в метамодели UML определены элементы модели (сущности и отношения) и способы их комбинирования, а девять типов диаграмм – это уже надстройка над языком, отражающая сложившуюся практику его использования.

Канонические диаграммы отнюдь не образуют полного ортогонального набора: они пересекаются как по включенным в них средствам, так и по области применения. Более того, некоторые из них являются частными случаями других, есть просто семантически

эквивалентные пары, можно привести примеры допустимых диаграмм, для которых затруднительно указать однозначно, к какому именно из канонических типов диаграмма относится.

Сказанное можно проиллюстрировать условной классификацией диаграмм, приведенной ниже.

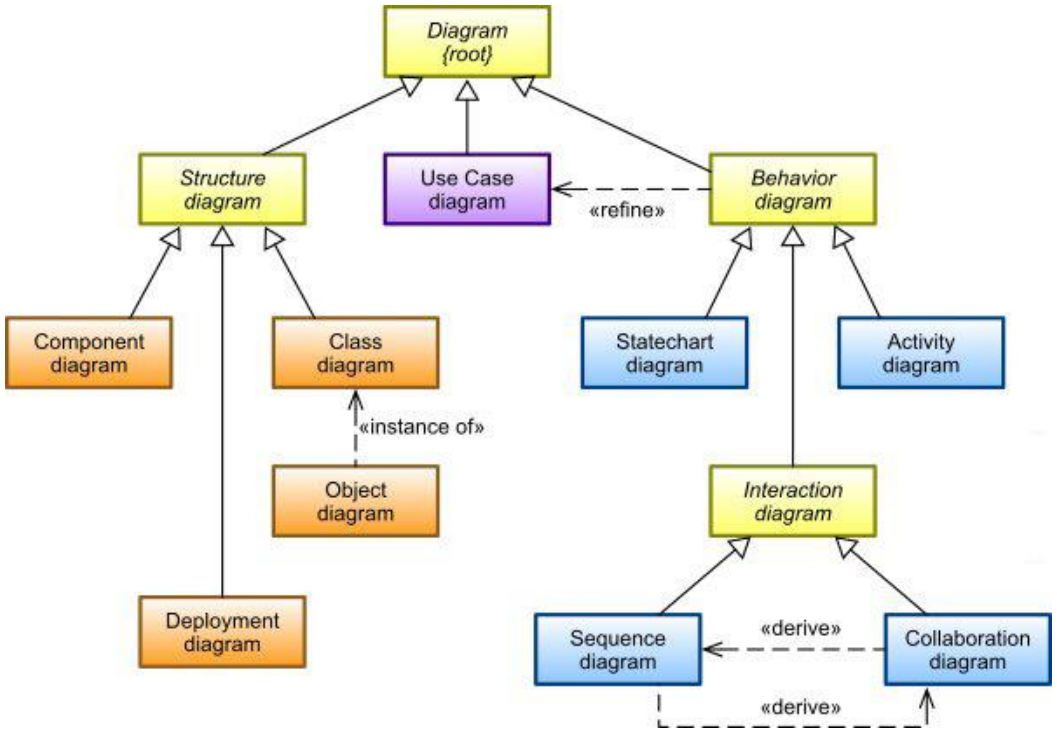

Рис. **Иерархия типов диаграмм для UML 1**

Мы поместили диаграмму использования отдельно, не относя ее ни к диаграммам описания структуры, ни к диаграммам описания поведения. В большинстве источников диаграммы использования относят к описанию поведения, что нам представляется некоторой натяжкой[v]. Кроме отношений обобщения, на диаграмме классов, а именно эта диаграмма изображена на рис. **Иерархия типов диаграмм для UML 1**, дополнительно показаны некоторые зависимости между отдельными UML диаграммами. Эти зависимости в каждом конкретном случае носят различный характер, что отражено посредством использования различных стереотипов. Подробнее о стандартных стереотипах зависимостей.

В UML 2[v] внесены значительные коррективы как в список канонических диаграмм, а именно их число увеличилось до 13, так и в список доступных конструкций языка, что значительно расширило область его применения. Кроме этого две диаграммы были переименованы: диаграмма кооперации была переименована в диаграмму коммуникации[vv], а диаграмма состояний в диаграмму автомата.

Список новых диаграмм и их названий, приведен ниже.

- Диаграмма внутренней структуры (Composite Structure diagram)
- Диаграмма пакетов (Package diagram)
- Диаграмма автомата (State machine diagram)
- Диаграмма коммуникации (Communication diagram)
- Обзорная диаграмма взаимодействия (Interaction Overview diagram)
- Диаграмма синхронизации (Timing diagram)

На рис. **Иерархия типов диаграмм для UML 2 (часть 1 и 2)** приведена диаграмма классов, отражающая взаимосвязь диаграмм в UML 2.

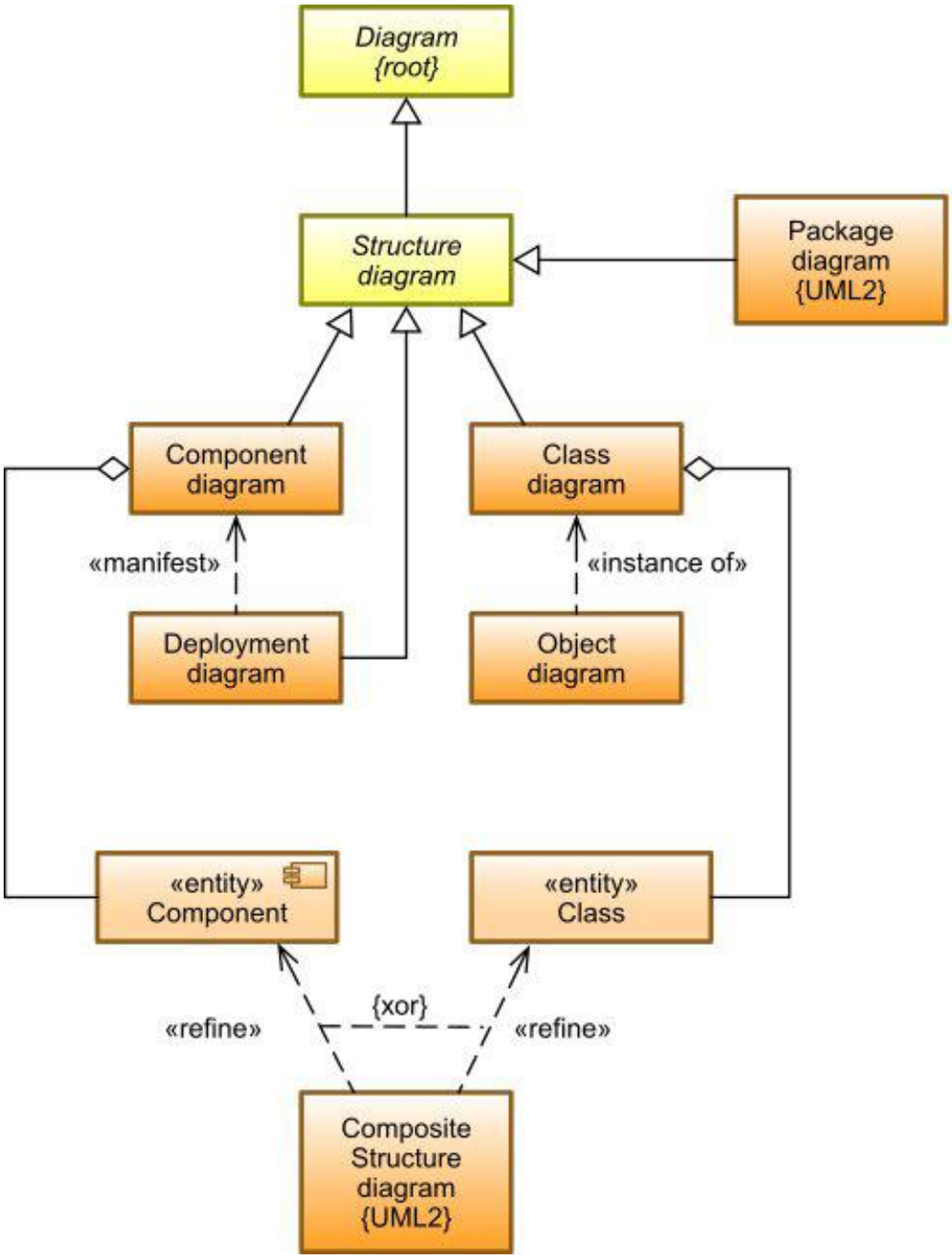

Рис. **Иерархия типов диаграмм для UML 2 (часть 1)**

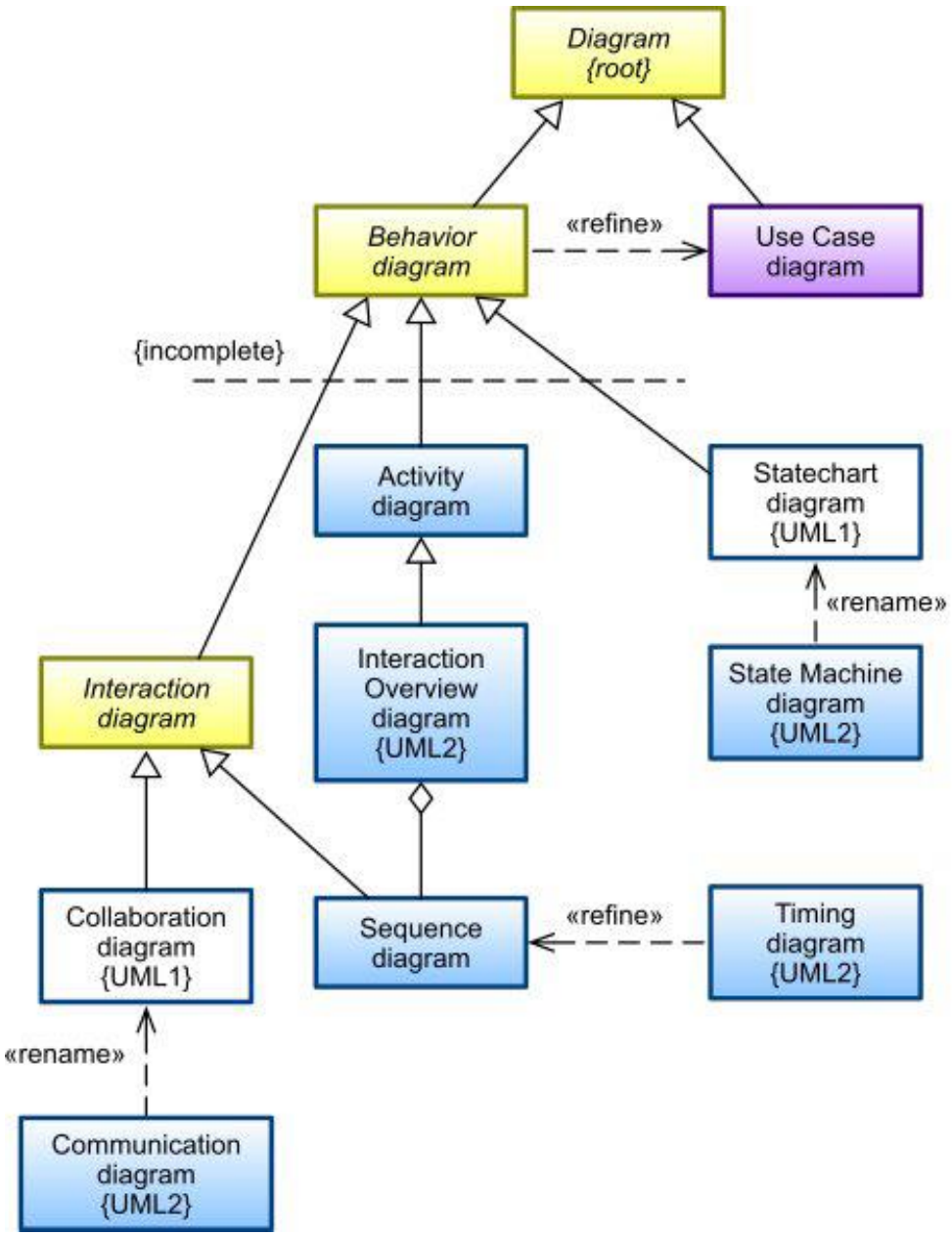

Рис. **Иерархия типов диаграмм для UML 2 (часть 2)**

Общий шаблон представления диаграммы приведен ниже.

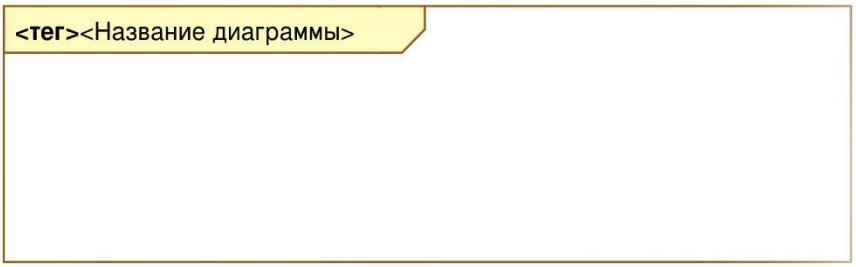

Рис. **Нотация для диаграмм**

Основных элементов оформления два: наружная рамка и ярлычок с названием диаграммы. Если с рамкой все просто – это прямоугольник, ограничивающий область в котором должны находиться элементы диаграммы, то название диаграммы записывается в специальном формате, приведенном на рис. **Нотация для диаграмм.**

Указанная сложная форма ярлычка поддерживается не всеми инструментами. Впрочем, это не обязательно, поскольку семантика первична, а нотация вторична. Далее мы везде используем в качестве ярлычка диаграммы прямоугольник, и это не должно вызывать недоразумений.

Возможные теги (типы) для диаграмм приведены в следующей таблице. Теги, предлагаемые стандартом, записаны во второй столбец. Однако, как показала практика, предлагаемые стандартом правила не всегда удобны и логически обоснованы, поэтому третий столбец таблицы содержит разумную на наш взгляд альтернативу.

Табл. **Типы и теги диаграмм**

Название диаграммы	Тег (стандартный)	Тег (предлагаемый)
Диаграмма использования	**use case** или **uc**	**use case**
Диаграмма классов	**class**	**class**
Диаграмма автомата	**state machine** или **stm**	**state machine**
Диаграмма деятельности	**activity** или **act**	**activity**
Диаграмма последовательности	**interaction** или **sd**	**sd**
Диаграмма коммуникации	**interaction** или **sd**	**comm**
Диаграмма компонентов	**component** или **cmp**	**component**
Диаграмма размещения	не определен	**deployment**
Диаграмма объектов	не определен	**object**
Диаграмма внутренней структуры	**class**	**class** или **component**
Обзорная диаграмма взаимодействия	**interaction** или **sd**	**interaction**
Диаграмма синхронизации	**interaction** или **sd**	**timing**
Диаграмма пакетов	**package** или **pkg**	**package**