Высокоуровневые методы информатики и программирования

Лекция 3

Концептуальная модель UML

Содержание

- Структура языка UML
- Обзор UML
- Концептуальная модель UML

Использование UML

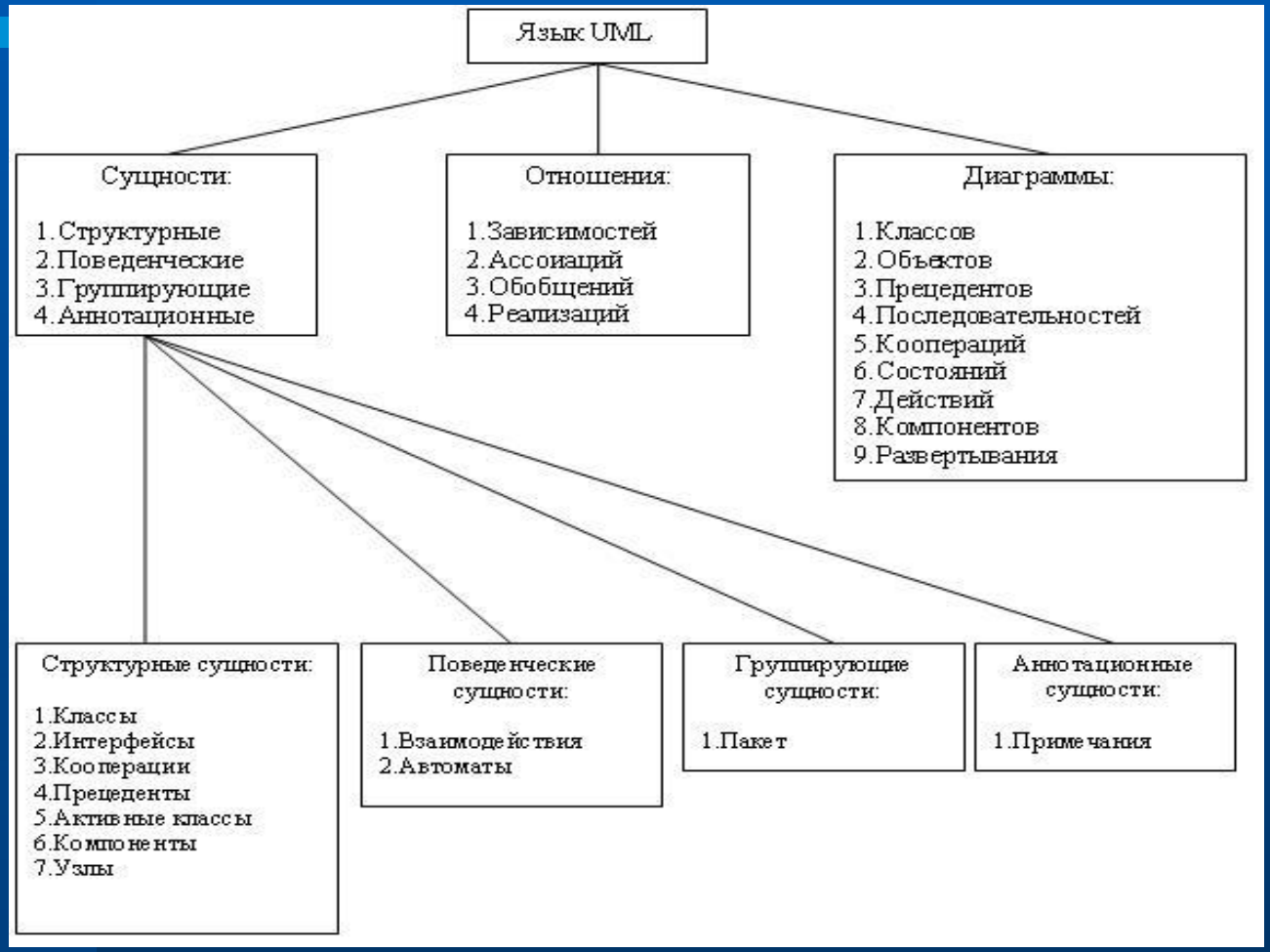
Язык UML предназначен прежде всего для разработки программных систем. Его использование особенно эффективно в следующих областях:

- информационные системы масштаба предприятия;
- банковские и финансовые услуги;
- телекоммуникации;
- транспорт;
- оборонная промышленность, авиация и космонавтика;
- розничная торговля;
- медицинская электроника;
- наука;
- распределенные Web-системы.

Использование UML

Сфера применения UML не ограничивается моделированием программного обеспечения. Его выразительность позволяет моделировать, скажем, документооборот в юридических системах, структуру и функционирование системы обслуживания пациентов в больницах, осуществлять проектирование аппаратных средств.

Структура языка UML



Обзор UML

Унифицированный язык моделирования (UML) является стандартным инструментом для создания "чертежей" программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем.

Обзор UML

UML пригоден для моделирования любых систем: от ИС масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования.

Изучение UML удобнее всего начать с его концептуальной модели, которая включает в себя три основных элемента: базовые строительные блоки, правила, определяющие, как эти блоки могут сочетаться между собой, и некоторые общие механизмы языка.

Обзор UML

Несмотря на свои достоинства, UML - это всего лишь язык; он является одной из составляющих процесса разработки программного обеспечения, и не более того. Хотя UML не зависит от моделируемой реальности, лучше всего применять его, когда процесс моделирования основан на рассмотрении прецедентов использования, является итеративным и пошаговым, а сама система имеет четко выраженную архитектуру.

Модели UML

- Модель использования *(описание функциональности ПО с т.зр. пользователя)*
- Логическая модель (*описывает ключевые абстракции ПО – классы, интерфейсы и т.п.,*)
- Модель реализации (*определяет реальную организацию программных модулей в среде разработки*)
- Модель процессов (*отображает организацию вычислений и оперирует понятиями «процессы», «нити»*)
- Модель развертывания (*показывает особенности размещения программных компонентов на конкретном оборудовании)*

Построение концептуальной модели UML

Для понимания UML необходимо усвоить его концептуальную модель, которая включает в себя три составные части:

- основные строительные блоки языка,
- правила их сочетания и
- некоторые общие для всего языка механизмы.

Усвоив эти элементы, вы сумеете читать модели на UML и самостоятельно создавать их - вначале, конечно, не очень сложные. По мере приобретения опыта в работе с языком вы научитесь пользоваться и более развитыми его возможностями.

Пример

Категория понятий-кандидатов	Примеры
Физические или материальные объекты	Самолет- как целое
Спецификации	Цвет
Место	Аэропорт
Роль человека	Покупатель
Контейнеры других объектов	Самолет как совокупность частей
Внешние системы	Система бронирования билетов
Абстрактные понятия Организация События	Летательный аппарат Фирма Встреча, покупка билета

Строительные блоки UML

Словарь языка UML включает три вида строительных блоков:

- сущности;
- отношения;
- диаграммы.

Сущности - это абстракции, являющиеся основными элементами модели. Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

Строительные блоки UML

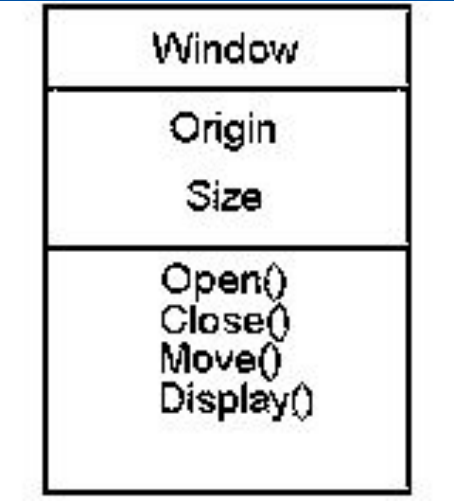
В UML имеется четыре типа сущностей:

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

Сущности являются основными объектно-ориентированными блоками языка. С их помощью можно создавать корректные модели.

Структурные сущности - это имена существительные в моделях на языке UML Как правило, они представляют собой статические части модели, соответствующие концептуальным или физическим элементам системы. Существует семь разновидностей структурных сущностей.

Строительные блоки UML



Класс (Class) - это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Класс реализует один или несколько интерфейсов. Графически класс изображается в виде прямоугольника, в котором обычно записаны его имя, атрибуты и операции.

Строительные блоки UML

Интерфейс (Interface) - это совокупность операций, которые определяют сервис (набор услуг), предоставляемый классом или компонентом. Т. о., интерфейс описывает видимое извне поведение элемента.

Интерфейс может представлять поведение класса или компонента полностью или частично; он определяет только спецификации операций (сигнатуры), но никогда - их реализации.

Графически интерфейс изображается в виде круга, под кот.пишется его имя.

Интерфейс редко существует сам по себе - обычно он присоединяется к реализующему его классу или компоненту.

Строительные блоки UML

Кооперация (Collaboration) определяет взаимодействие; она представляет собой совокупность ролей и других элементов, которые, работая совместно, производят некоторый кооперативный эффект, не сводящийся к простой сумме слагаемых.

Кооперация, следовательно, имеет как структурный, так и поведенческий аспект. Один и тот же класс может принимать участие в нескольких кооперациях; т.о., они являются реализацией образцов поведения, формирующих систему.

Графически кооперация изображается в виде эллипса, ограниченного пунктирной линией, в который обычно заключено только имя.

Строительные блоки UML

Прецедент (Use case) - это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного *актера* (Actor).

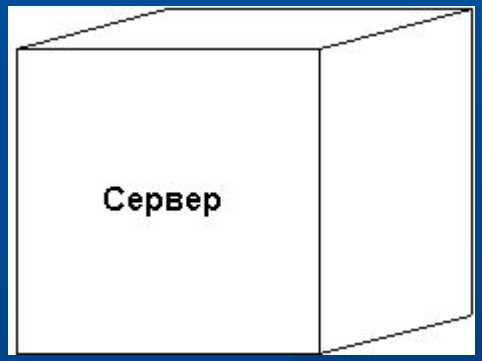
Прецедент применяется для структурирования поведенческих сущностей модели.

Прецеденты реализуются посредством кооперации. Графически прецедент изображается в виде ограниченного непрерывной линией эллипса, обычно содержащего только его имя.

Строительные блоки UML

Активным классом (Active class) называется класс, объекты которого вовлечены в один или несколько процессов, или нитей (Threads), и поэтому могут инициировать управляющее воздействие. Активный класс во всем подобен обычному классу, за исключением того, что его объекты представляют собой элементы, деятельность которых осуществляется одновременно с деятельностью других элементов. Графически активный класс изображается так же, как простой класс, но ограничивающий прямоугольник рисуется жирной линией и обычно включает имя, атрибуты и операции.

Строительные блоки UML



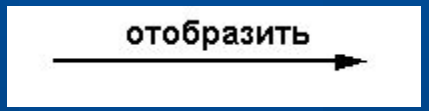
Узел (Node) - это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки. Совокупность компонентов может размещаться в узле, а также мигрировать с одного узла на другой. Графически узел изображается в виде куба, обычно содержащего только имя.

Строительные блоки UML

Поведенческие сущности (Behavioral things) являются динамическими составляющими модели UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей.

Строительные блоки UML

Взаимодействие (Interaction) - это поведение, суть которого заключается в обмене сообщениями (Messages) между объектами в рамках конкретного контекста для достижения определенной цели. С помощью взаимодействия южно описать как отдельную операцию, так и поведение совокупности объектов. взаимодействие предполагает ряд других элементов, таких как сообщения, последовательности действий (поведение, инициированное сообщением) и связи (между объектами). Графически сообщения изображаются в виде стрелки, над которой почти всегда пишется имя соответствующей операции.



Строительные блоки UML

Автомат (State machine) - это алгоритм поведения, определяющий последовательность состояний, через которые объект или взаимодействие проходят на протяжении своего жизненного цикла в ответ на различные события, а также реакции на эти события. С помощью автомата можно описать поведение сдельного класса или кооперации классов. С автоматом связан ряд других элементов: состояния, переходы (из одного состояния в другое), события (сущности, инициирующие переходы) и виды действий (реакция на переход). Графически состояние изображается в виде прямоугольника с закругленными углами, содержащего имя и, возможно, подсостояния.

Строительные блоки UML

В языке UML определены четыре типа *отношений*:

- зависимость;
- ассоциация;
- обобщение;
- реализация.

Строительные блоки UML

Зависимость (Dependency) - это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой. Графически зависимость изображается в виде прямой пунктирной линии, часто со стрелкой, которая может содержать метку.

Строительные блоки UML

Ассоциация (Association) - структурное отношение, описывающее совокупность связей; связь - это соединение между объектами. Разновидностью ассоциации является *агрегирование* (Aggregation) - так называют структурное отношение между целым и его частями. Графически ассоциация изображается в виде прямой линии (иногда завершающейся стрелкой или содержащей метку), рядом с которой могут присутствовать дополнительные обозначения, на пример кратность и имена ролей.

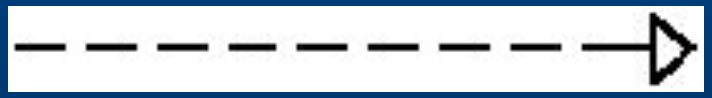
Строительные блоки UML

Обобщение (Generalization) - это отношение "специализация/обобщение", при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (родителя или предка).

Т.о., потомок (Child) наследует структуру и поведение своего родителя (Parent). Графически отношение обобщения изображается в виде линии с не закрашенной стрелкой, указывающей на родителя.

Строительные блоки UML

Реализация (Realization) - это семантическое отношение между классификаторами, при котором один классификатор определяет "контракт", а другой гарантирует его выполнение. Отношения реализации встречаются в двух случаях: во-первых, между интерфейсами и реализующими их классами или компонентами, а во-вторых, между прецедентами и реализующими их кооперациями. Отношение реализации изображается в виде пунктирной линии с не закрашенной стрелкой, как нечто среднее между отношениями обобщения и зависимости



Строительные блоки UML

Четыре описанных элемента являются основными типами отношений, которые можно включать в модели UML. Существуют также их вариации, например уточнение (Refinement), трассировка (Trace), включение и расширение (для зависимостей).

Строительные блоки UML

Диаграмма в UML - это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Диаграмма - в некотором смысле одна из проекций системы. Как правило, за исключением наиболее тривиальных случаев, диаграммы дают свернутое представление элементов, из которых составлена система. Один и тот же элемент может присутствовать во всех диаграммах, или только в нескольких (самый распространенный вариант), или не присутствовать ни в одной (очень редко).

Строительные блоки UML

Теоретически диаграммы могут содержать любые комбинации сущностей и отношений. На практике, однако, применяется сравнительно небольшое количество типовых комбинаций, соответствующих пяти наиболее употребительным видам, которые составляют архитектуру программной системы.

Строительные блоки UML

Таким образом, в UML выделяют девять типов диаграмм:

- диаграммы вариантов использования;
- диаграммы классов;
- диаграммы пакетов;
- диаграммы последовательностей действий;
- диаграммы кооперации;
- диаграммы деятельностей;
- диаграммы состояний объектов;
- диаграммы компонентов;
- диаграммы размещения.

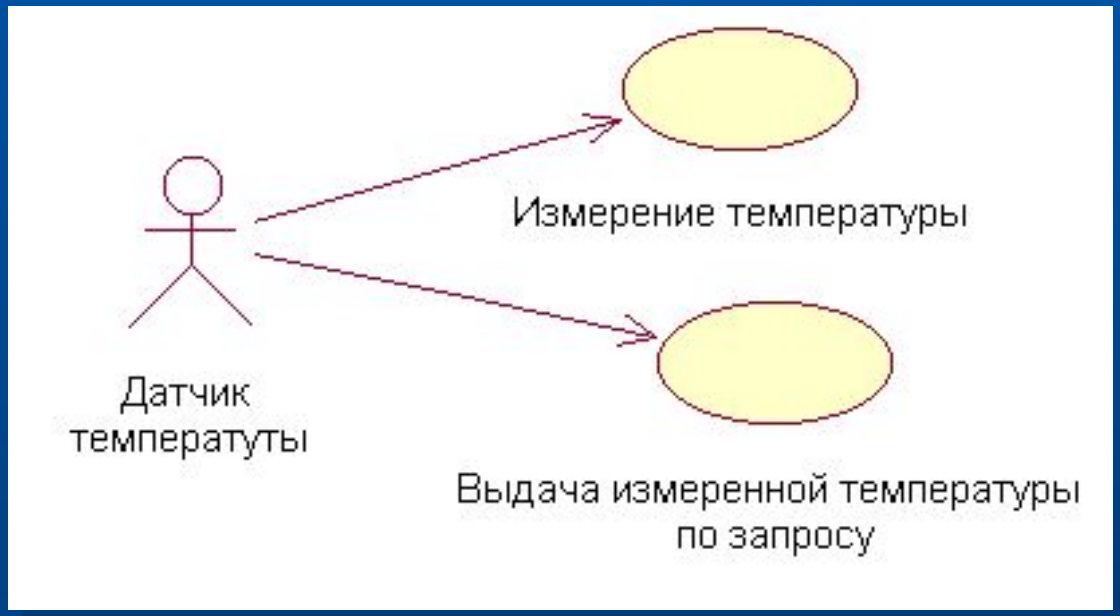
Use case diagram

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Каждая такая диаграмма или, как ее обычно называют, каждый Use case – это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

Use case diagram



Deployment diagram

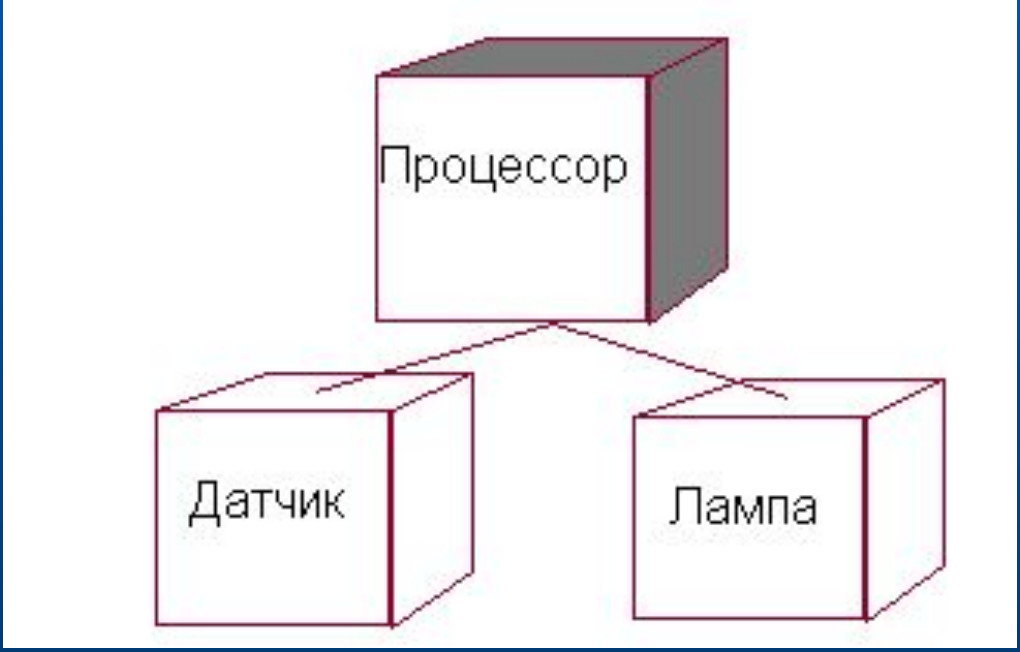
Диаграммы топологии

Этот вид диаграмм предназначен для анализа аппаратной части системы, то есть «железа», а не программ. В прямом переводе с английского Deployment означает «развертывание», но термин «топология» точнее отражает сущность этого типа диаграмм.

Для каждой модели создается только одна такая диаграмма, отображающая процессоры (Processor), устройства (Device) и их соединения.

Обычно этот тип диаграмм используется в самом начале проектирования системы для анализа аппаратных средств, на которых она будет эксплуатироваться.

Deployment diagram



Диаграммы состояний

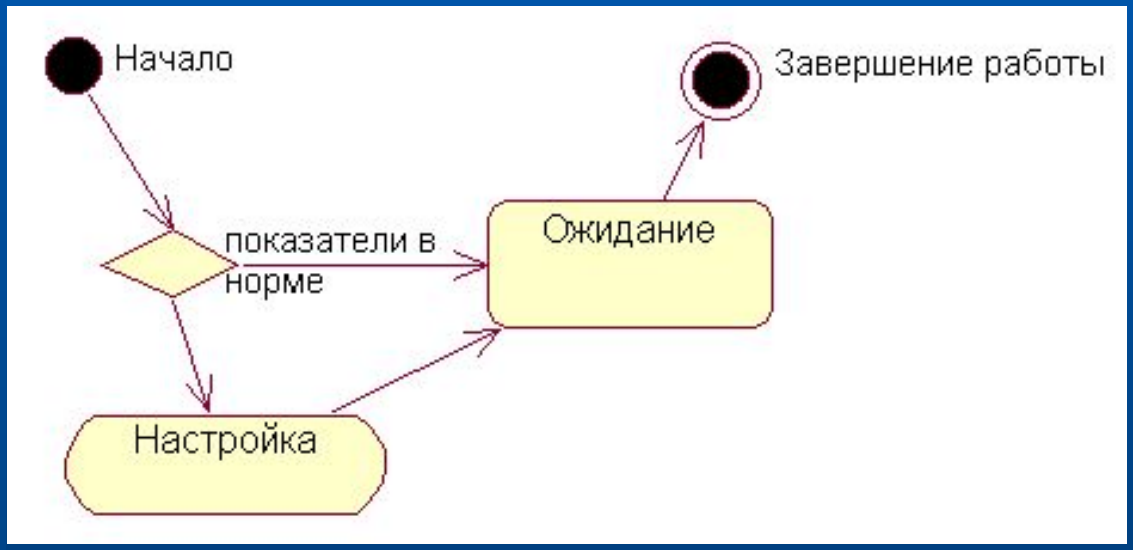
State Machine diagram (диаграммы состояний)

Каждый объект системы, обладающий определенным поведением, может находится в определенных состояниях, переходить из состояния в состояние, совершая определенные действия в процессе реализации сценария поведения объекта. Поведение большинства объектов реальных систем можно представить с точки зрения теории конечных автоматов, то есть поведение объекта отражается в его состояниях, и данный тип диаграмм позволяет отразить это графически. Для этого используется два вида диаграмм: Statechart diagram (диаграмма состояний) и Activity diagram (диаграмма активности)

Диаграмма состояний

Диаграмма состояний (Statechart) предназначена для отображения состояний объектов системы, имеющих сложную модель поведения. Это одна из двух диаграмм State Machine, доступ к которой осуществляется из одного пункта меню.

Диаграмма активности



Диаграммы активности

Фактически данный тип диаграмм может использоваться и для отражения состояний моделируемого объекта, однако, основное назначение Activity diagram в том, чтобы отражать бизнес-процессы объекта. Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов.

Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов любой сложности, в том числе может использоваться для составления блок-схем.

Диаграммы взаимодействия

Interaction diagram (диаграммы взаимодействия)

Этот тип диаграмм включает в себя диаграммы Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Диаграммы взаимодействия

Sequence diagram (диаграммы последовательностей действий)

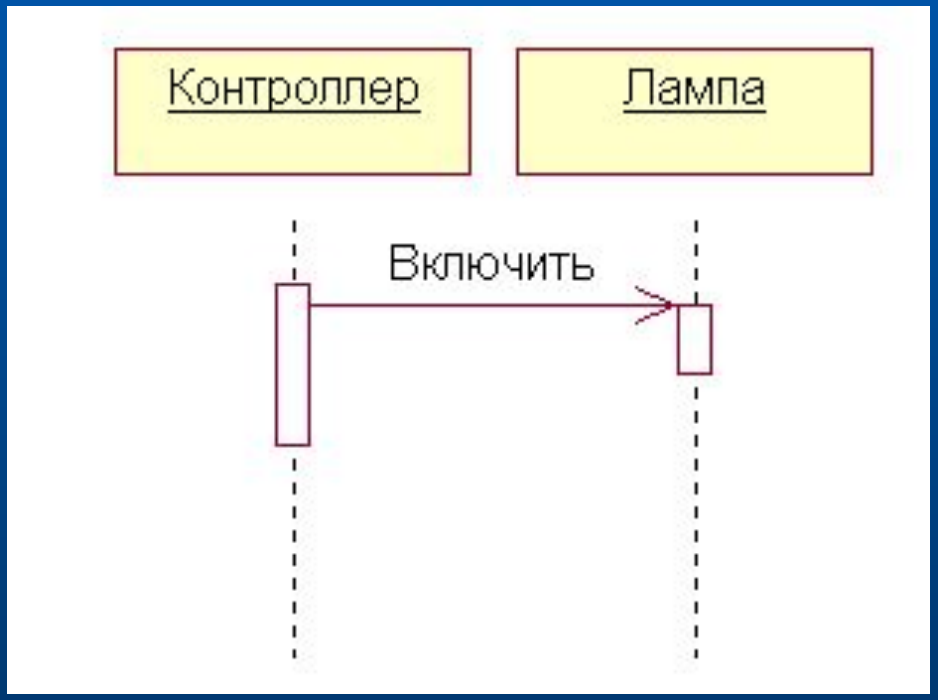
Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами.

Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений. Для того чтобы окинуть взглядом все взаимосвязи объектов, служит Collaboration diagram.

Диаграммы взаимодействия

Sequence diagram (диаграммы последовательностей действий)



Диаграммы взаимодействия

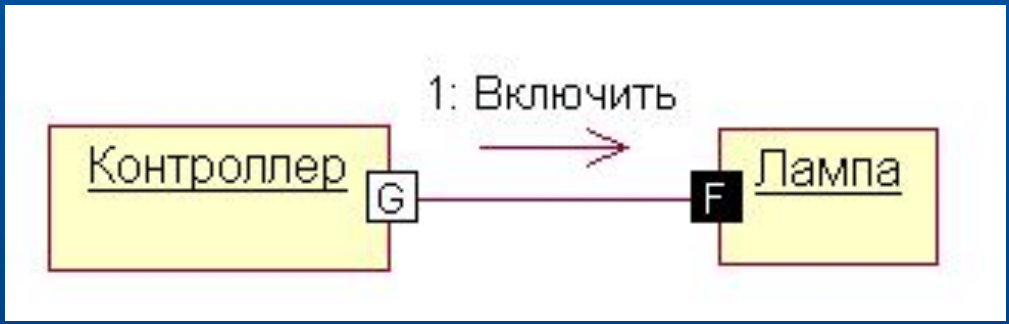
Collaboration diagram (диаграммы сотрудничества)

Этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

По причине того, что диаграммы Sequence и Collaboration являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из Sequence диаграммы диаграмму Collaboration и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

Диаграммы взаимодействия

Collaboration diagram (диаграммы сотрудничества)



Диаграммы классов

Class diagram (диаграммы классов)

Этот тип диаграмм позволяет создавать логическое представление системы, на основе которого создается исходный код описанных классов.

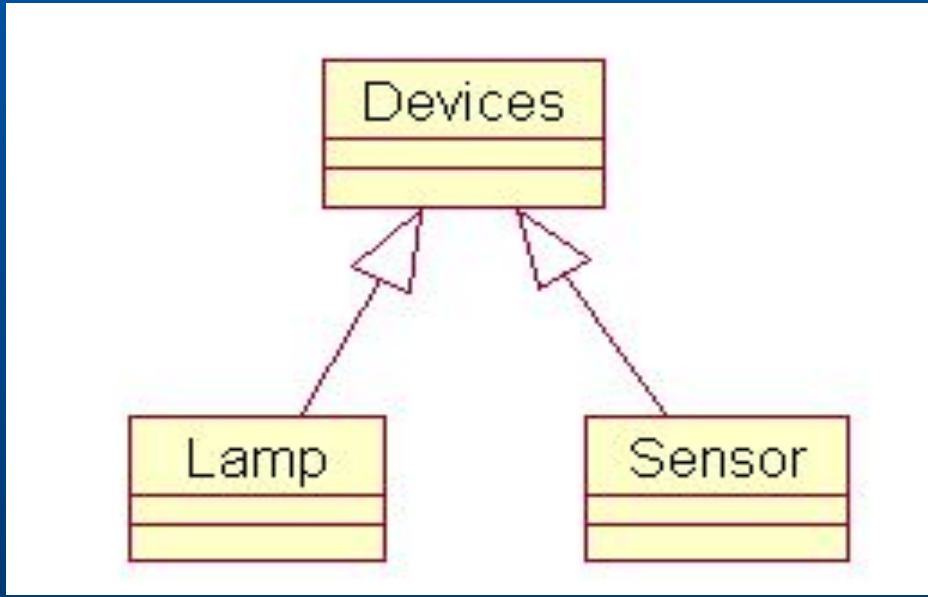
Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (Classes) и интерфейсов (Interfaces). Данный тип диаграмм противоположен по содержанию диаграмме Collaboration, на котором отображаются объекты системы.

Диаграммы классов

Rational Rose позволяет создавать классы при помощи данного типа диаграмм в различных нотациях. В нотации, предложенной Г. Бучем, которая так и называется Booch, классы изображаются в виде чего-то нечеткого, похожего на облако. Таким образом Г.Буч пытается показать, что класс – это лишь шаблон, по которому в дальнейшем будет создан конкретный объект.

Диаграммы классов

Class diagram (диаграммы классов)



Диаграммы компонентов

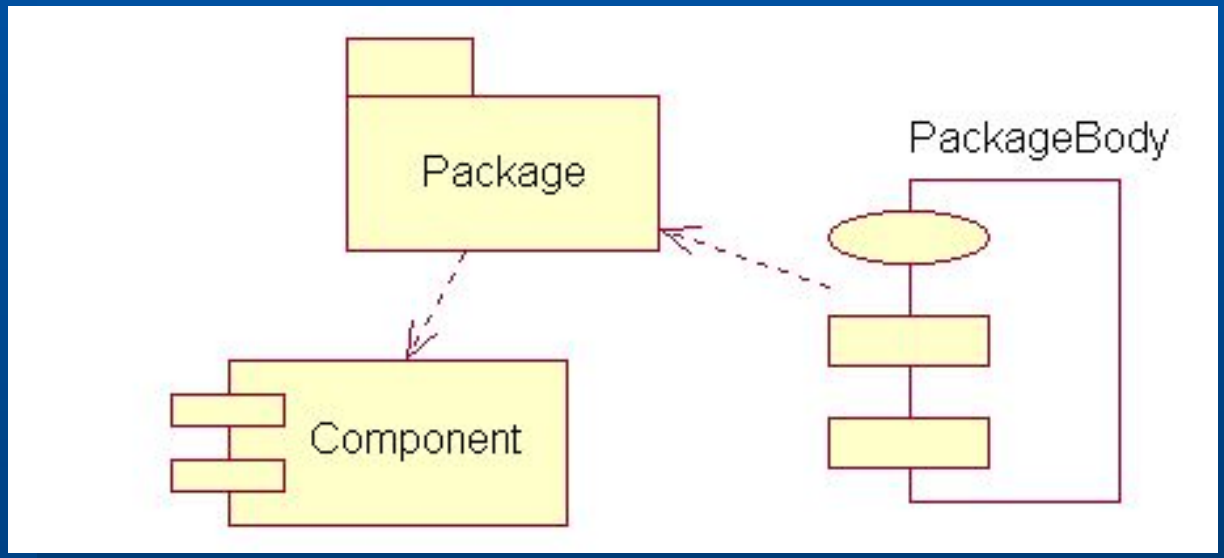
Component diagram (диаграммы компонентов)

Этот тип диаграмм предназначен для распределения классов и объектов по компонентам при физическом проектировании системы. Часто данный тип диаграмм называют диаграммами модулей.

При проектировании больших систем может оказаться, что система должна быть разложена на несколько сотен или даже тысяч компонентов, и этот тип диаграмм позволяет не потеряться в обилии модулей и их связей.

Диаграммы компонентов

Component diagram (диаграммы компонентов)



Вопросы?