Введение

Авторы UML:
Грэди Буч, Джеймс Рамбо, Айвар Джекобсон (Якобсон)
Grady Booch, James Rumbaugh, Ivar Jacobson

Unified Modeling Language — Унифицированный язык моделирования

Стандартный язык моделирования для программного обеспечения — язык для визуализации, описания, проектирования и документирования артефактов программных систем. Язык, используемый в унифицированном процессе. Язык, позволяющий разработчикам визуализировать их рабочие продуты (артефакты) в стандартизированных диаграммах.

Rational Unified Process — Унифицированный процесс

Процесс разработки программного обеспечения, основанный на Унифицированном языке моделирования, итеративный, архитектурно-ориентированный, управляемый вариантами использования и рисками. Процесс, организованный в четыре фазы, — анализа и планирования требований, проектирования, построения и внедрения, и в пять основных рабочих процессов — определение требование, анализ, проектирование, разработка и тестирование.

В языке UML определены следующие виды диаграмм:

- Диаграмма Вариантов Использования (Use Case Diagram) (Прецедентов)
- Диаграмма классов (Class Diagram)

- *Диаграммы поведения (Behavior Diagrams)*
  - Диаграмма состояний (State Chart Diagram)
  - Диаграмма деятельности (Activity Diagram)
- *Диаграммы взаимодействия (Interaction Diagrams)*
  - Диаграмма последовательности (Sequence Diagram)
  - Диаграмма кооперации (Collaboration Diagram)
- *Диаграммы реализации (Implementation Diagrams)*
  - Диаграмма компонентов (Component Diagram)
  - Диаграмма развертывания (Deployment Diagram)

На диаграммах языка UML существуют три типа визуальных обозначений:

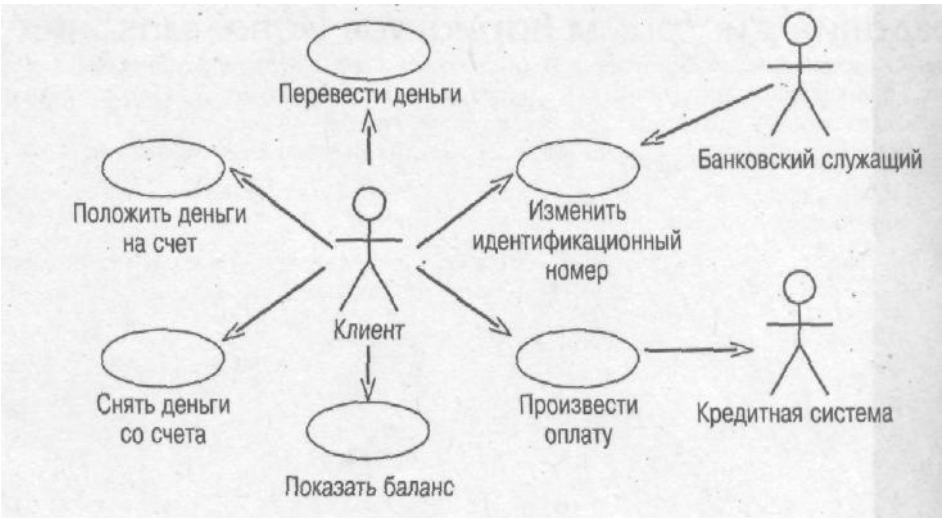
- *Связи*, которые представляются различными линиями на плоскости. Связи в языке UML обобщают понятие дуг и ребер из теории графов; но имеют менее формальный характер.
- *Текст*, который содержится внутри границ отдельных геометрических фигур на плоскости. При этом форма этих фигур (прямоугольник, эллипс) соответствует некоторым элементам языка UML (класс, вариант использования) и имеет фиксированную семантику.
- *Графические символы*, изображаемые вблизи от тех или иных визуальных элементов диаграмм.

Диаграмма Вариантов Использования

Диаграмма Вариантов Использования может содержать:

- Варианты использования
- Действующих лиц
- Связи коммуникации между вариантами использования и действующими лицами
- Связи использования и расширения между вариантами использования

- Связи обобщения действующих лиц
- Диаграммы Вариантов Использования
- Диаграммы Последовательности и Кооперативные диаграммы



Цель диаграмм Вариантов Использования — документирование вариантов использования (все входящее в сферу применения системы), действующих лиц (все вне этой сферы) и связей между ними. При построении диаграммы Вариантов Использования, следует придерживаться следующих правил:

Между *компонентами* диаграммы вариантов использования могут существовать различные *отношения* (связи), которые описывают *взаимодействие* экземпляров одних *актеров* и *вариантов использования* с экземплярами других актеров и вариантов. В языке UML имеется несколько стандартных видов отношений между актерами и вариантами использования:

- Отношение ассоциации (association relationship)
- Отношение расширения (extend relationship)
- Отношение включения (include relationship)
- Отношение обобщения (generalization relationship)

Варианты Использования начинают описывать, *что должна будет делать ваша система*. Более *конкретные детали* определяются в документе, называемом "потоком событий" (**flow of events**).

Цель потока событий — документирование процесса обработки данных, реализуемого в рамках варианта использования.

Этот документ подробно описывает, что будут *делать пользователи системы* и что — *сама система*.

Цель потока событий— описать, **что** будет делать система, **а не как** она будет это делать. Обычно поток событий содержит:

- Краткое описание
- Предусловия (pre-conditions)
- Основной поток событий
- Альтернативный поток событий
- Постусловия (post-conditions)

Описание

Каждый Вариант Использования должен иметь связанное с ним короткое описание того, что он будет делать.

Предусловия

Предусловия варианта использования — это такие условия, которые должны быть выполнены, прежде чем вариант использования начнет свою работу.

Основной и альтернативный потоки событий

Конкретные детали вариантов использования отражаются в *основном* в *альтернативном* потоках событий. Поток событий поэтапно описывает, что должно происходить во время выполнения заложенной в варианты использования функциональности. Поток событий уделяет внимание тому, что (а не как) будет делать система, причем описывает это с точки зрения пользователя. Первичный и альтернативный потоки событий содержат:

- Описание того, каким образом запускается вариант использования
- Различные пути выполнения варианта использования
- Нормальный, или основной, поток событий варианта использования
- Отклонения от основного потока событий (так называемые альтернативные потоки)
- Потоки ошибок
- Описание того, каким образом завершается вариант использования

Постусловия

Постусловиями называются такие условия, которые должны быть выполнены после завершения варианта использования..

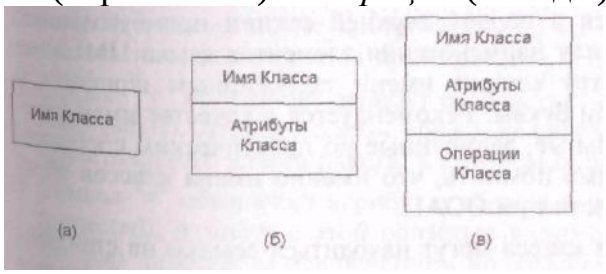
Архитектура

Понятие *архитектуры* программы включает в себя наиболее важные статические и динамические аспекты системы. Архитектура вырастает из требований к результату, в том виде, как их понимают пользователи и другие заинтересованные лица. Эти требования отражаются в вариантах использования.

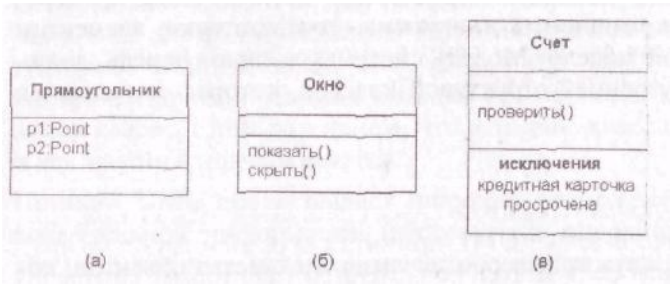
Диаграмма классов

Диаграмма классов (**class diagram**) служит для представления *статической структуры модели системы* в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений

Класс (class) в языке UML служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. В разделах могут указываться имя класса, *атрибуты* (переменные) и *операции* (методы).



Обязательным элементом обозначения класса является его имя. На начальных этапах разработки диаграммы отдельные классы могут обозначаться простым прямоугольником с указанием только имени соответствующего класса.



Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Должно начинаться с заглавной буквы. Рекомендуется в качестве имен классов использовать *существительные*, записанные по *практическим соображениям* без пробелов.

Примеры имен классов: "Сотрудник", "Компания", "Руководитель", "Клиент", "Продавец", "Менеджер", "Офис". Имена имеют непосредственное отношение к моделируемой предметной области и функциональному назначению проектируемой системы.

Класс может не иметь экземпляров или объектов. В этом случае он называется *абстрактным* классом, а для обозначения его имени используется наклонный шрифт (курсив).

Атрибуты класса

Во второй секции прямоугольника класса записываются его *атрибуты* (attributes) или свойства. Каждому атрибуту класса соответствует отдельная строка текста:

<квантор видимости><имя атрибута>[кратность]:
<тип атрибута> = <исходное значение>{строка-свойство}

Операция

Операция (**operation**) представляет собой некоторый сервис, предоставляющий каждый экземпляр класса по определенному требованию. Каждой операции класса соответствует отдельная строка:

<квантор видимости><имя операции>(список параметров):
<выражение типа возвращаемого значения>{строка-свойство}