Лекция – UML-диаграммы. Отношения между классами.

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного ***анализа*** и ***проектирования***.

Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Словарь UML включает три вида строительных блоков:

1. **Диаграммы.**
2. **Сущности.**
3. **Связи.**

Диаграммы

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Язык UML включает **13** видов диаграмм, среди которых на первом месте в списке – **диаграмма классов**.

Диаграммы классов показывают

1. набор классов,

2. интерфейсов,

3. а также их связи.

Диаграммы этого вида чаще всего используются для ***моделирования*** объектно-ориентированных систем.

Сущности

1. Структурные

2. Поведенческие

3. Аннотирующие

Структурные сущности – это «имена существительные» в модели UML. Основным видом структурной сущности в диаграммах классов является **класс.**

Поведенческие сущности

Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели, которое заключается в обмене ***сообщениями между наборами объектов***.

Всего существует **три** основных вида поведенческих сущностей.

1. **взаимодействие** (**interaction**) – представляет собой поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели.

Сообщение изображается в виде линии со стрелкой, почти всегда сопровождаемой именем операции.

Отобразить

Поведенческие сущности

Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели, которое заключается в обмене ***сообщениями между наборами объектов***.

Всего существует **три** основных вида поведенческих сущностей.

2. **автомат** (**state machine**) – представляет собой поведение, характеризуемое последовательностью состояний объекта, в которых он оказывается на протяжении своего жизненного цикла в ответ на события, вместе с его реакцией на эти события.

Графически состояние представлено прямоугольником с закругленными углами.

Поведенческие сущности

Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели, которое заключается в обмене ***сообщениями между наборами объектов***.

Всего существует **три** основных вида поведенческих сущностей.

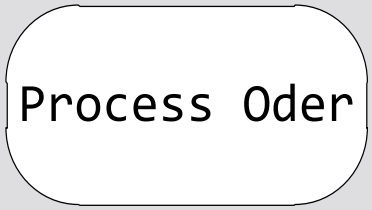
3. **деятельность** (**activity**) – определяет последовательность шагов процесса вычислений.

Например

Во ***взаимодействии*** внимание сосредоточено на наборе взаимодействующих объектов,

в ***автомате*** – на жизненном цикле одного объекта;

для деятельности же в центре внимания – последовательность шагов безотносительно к объектам, выполняющим каждый шаг. Отдельный шаг деятельности называется **действием** (**action**).

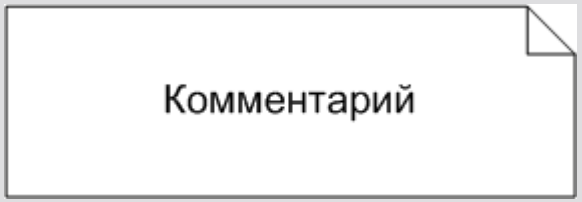


Сущности

Аннотирующие сущности – это поясняющие части UML-моделей, иными словами, *комментарии*, которые можно применить для описания, выделения и пояснения любого элемента модели.

Главная из аннотирующих сущностей **– *примечание*.**

Графически представлен прямоугольником с загнутым углом; внутри помещается текстовый или графический комментарий.



Структурные сущности - Класс

Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

1. Имя класса
2. Атрибуты (свойства) класса
3. Операции (методы) класса.

Для атрибутов и операций может быть указан один из трех типов видимости:
— — private (частный)
— protected (защищенный)
+ — public (общий)

Видимость для полей и методов указывается в виде левого символа (**-, #, +**) в строке с именем соответствующего элемента.

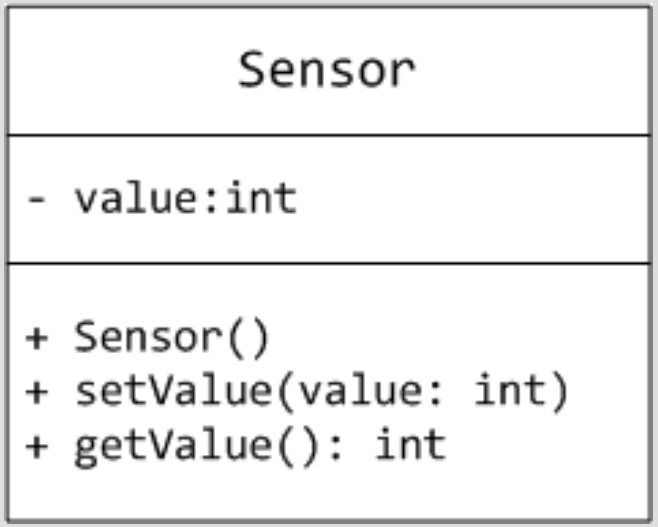
Структурные сущности - Класс

Каждый класс должен обладать ***именем***, отличающим его от других классов.

Для ***абстрактного класса*** имя класса записывается ***курсивом***.

Абстрактные методы класса обозначаются ***курсивным шрифтом***.

Статические методы класса ***обозначаются подчеркиванием***.



Отношения между классами

Существует четыре типа связей в UML:

1. Зависимость;

2. Ассоциация;

3. Обобщение;

4. Реализация;

Отношения между классами - Зависимость

Представляет собой связь между двумя элементами модели, в которой изменение одного элемента (независимого) может привести к изменению ***семантики*** другого элемента (зависимого).

Графически представлена пунктирной линией, иногда со стрелкой.
Со стороны стрелки указывается независимая сущность.
Может быть снабжена меткой.

Например, рассмотрим два простых класса **Y**, **X**.

Здесь, класс **Y** обладает методом **void Foo()**. Класс **X** содержит методы **f1,f2 f3,f4**, каждый из которых, определенным образом зависит от класса **Y**. Зависимость выражается в следующем, т.е. если в независимом классе **Y** изменить метод **Foo**, в частности либо название, либо добавить входные аргументы и т.д., то это повлечет за собой серьезные изменения в классе **X**, т.е. потребуется вносить изменения в каждый метод(изменять семантику вызовов).

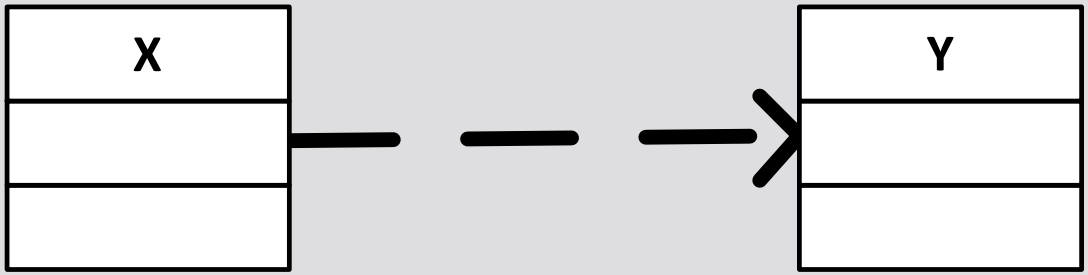
```
class Y{
public:
    void Foo(){        }
};

class X {
      void f1(Y y)  { y.Foo();        }
      void f3(Y &y) { y.Foo();        }
      void f2(Y *y) { y->Foo();       }
      void f4()     { Y y; y.Foo();   }};
```

```
class Y{
public:
    void Foo(){        }
};

class X {
        void f1(Y y)  { y.Foo();        }
        void f3(Y &y) { y.Foo();        }
        void f2(Y *y) { y->Foo();       }
        void f4()     { Y y; y.Foo();   }};
```

UML диаграмма для отображения такого отношения между классами, будет выглядеть следующим образом. Линия начинается у зависимого класса, а стрелка идет к независимому классу.



Отношения между классами - Ассоциация

Ассоциация – это структурная связь между элементами модели, которая описывает набор связей, существующих между ***объектами***.

Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно ***перемещаться*** от объектов одного класса к другому.

Двойные ассоциации представляются линией без стрелок на концах, соединяющей два классовых блока.

Отношения между классами - Ассоциация

Один конец ассоциации называется полюсом.
Полюс обладает *кратностью*, ограничивающей количество связанных между собой объектов.

Кратность (*multiplicity*) – это количество объектов одного класса, которые могут быть связаны с одним объектом класса противоположного конца ассоциации.
Кратность указывается рядом с полюсом под линией ассоциации и может принимать одно из следующих значений:

1 – точно один объект;
0..1 – ноль или один объект;
0..* – ноль или больше объектов;
1..* – один или больше объектов;
* – много объектов.

Отношения между классами - Ассоциация

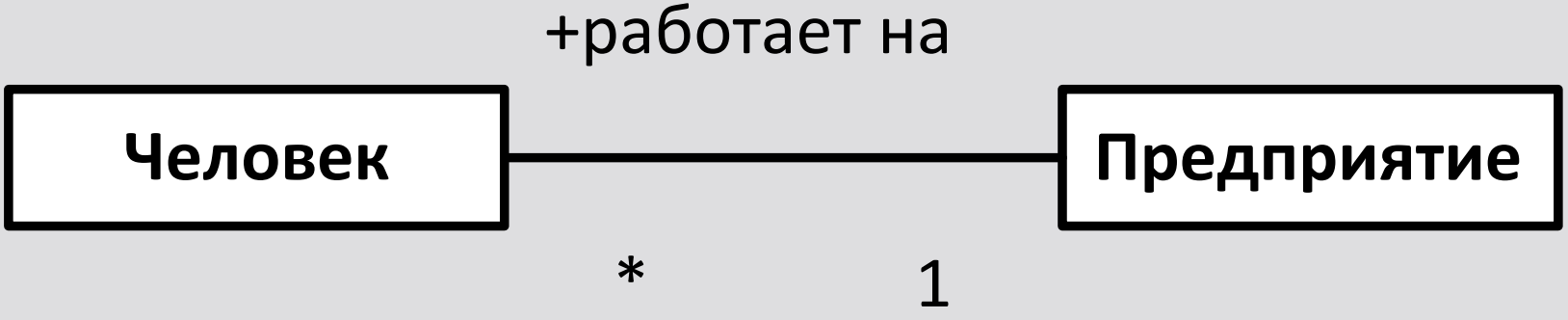
Например.

Здесь показана двунаправленная ассоциация с кратностью на одном полюсе *много*, а на другом – *один*.

Эта ассоциация двунаправленная, поэтому справедливы оба утверждения:

много людей работает на одном предприятии,

и на одном предприятии работает много людей.



```
class Y{
public:
    Foo(){        }
};

class X {
public:
        Y *y_ptr; // pointer
        X(Y *y) : y_ptr(y) {}
        void setY(Y *y) { y_ptr = y; }
        void f()        { y_ptr->Foo(); }
};
```

Ассоциация сводится к тому, что один объект (класса X) должен содержать **указатель** или **ссылку** на объект другого (класс Y). Обладая указателем или ссылкой можем изменять независимый объект, т.е. в зависимом классе X можем изменять независимый объект Y.

Отношения между классами - Агрегация

Агрегация – это разновидность *ассоциации*, обозначающая отношение *часть целого* (*part of*), где каждая пара классов определяется **отдельно.**

Агрегация имеет два важных свойства:

транзитивность (если A является частью B, а B – частью C, то A является частью C) и

асимметричность (если A является частью B, то B не является частью A).

Для отношения *часть целого* в UML существует две формы:
общая (**агрегация**) и частная (**композиция**).

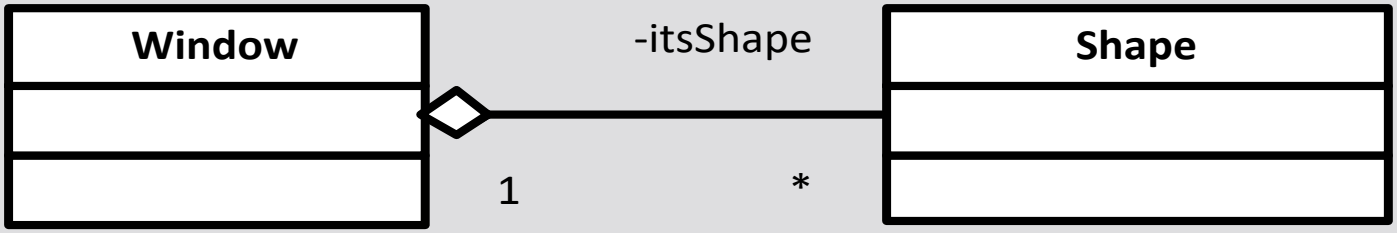
Агрегация встречается, когда один класс является ***коллекцией*** или ***контейнером*** других.

Отношения между классами - Агрегация

По умолчанию агрегацией называют агрегацию ***по ссылке***, то есть когда время существования содержащихся классов не зависит от времени существования содержащего их класса.

Если контейнер будет уничтожен, то его содержимое – нет.

Графически агрегация представляется пустым ромбом на блоке класса **«целое»**, и линией, идущей от этого ромба к классу **«часть»**.




```
class Window
{
private:
vector <itsShapes*> obj; //окно содержит список фигур
};
```

Отношения между классами - Композиция

Композиция (*composition*) – особый случай агрегации с дополнительными ограничениями:

1. часть может принадлежать не более чем одному целому;

2. приписанная к некоторому целому часть автоматически получает срок жизни, совпадающий со сроком жизни целого.

Программирование ***агрегации*** допускает ***независимую*** обработку объектов-частей и объекта-целого.

При программировании ***композиции*** ***удаление*** объекта-целого автоматически ***должно вызывать удаление*** всех составляющих его объектов-частей, поскольку они принадлежат целому.

Отношения между классами - Композиция

- **Композиция** — более строгий вариант агрегации. Т.е Если контейнер будет ***уничтожен***, то всё его содержимое будет ***также уничтожено***.
- 


```
class X {
        Y a; // 1; Composition
        Y b[10]; // 0..10; Composition
};

class X {
        Y *a; // 0..10; Composition
        X() { a = new Y[10]; }
        ~X(){ delete [] a; }
};
```

Отношения между классами -Обобщение

Обобщение (*generalization*) является частным случаем *ассоциации* и обозначает отношение типа *общее – частное* (*is – a*).

Обобщение указывает на наличие общих характеристик (атрибутов и операций) между классами.

Класс, имеющий наиболее общие для некоторой группы классов характеристики, называется ***суперклассом***, а его специализированная версия – ***подклассом***.

Открытые и защищенные атрибуты и операции суперкласса наследуются всеми подклассами.

Любой подкласс может добавлять уточняющие характеристики к унаследованным.

Дополнительные атрибуты и/или операции в подклассах могут отсутствовать.

Отношение обобщения **не является** набором связей между объектами, поэтому направление и кратность для обобщения **не указываются** подклассов.

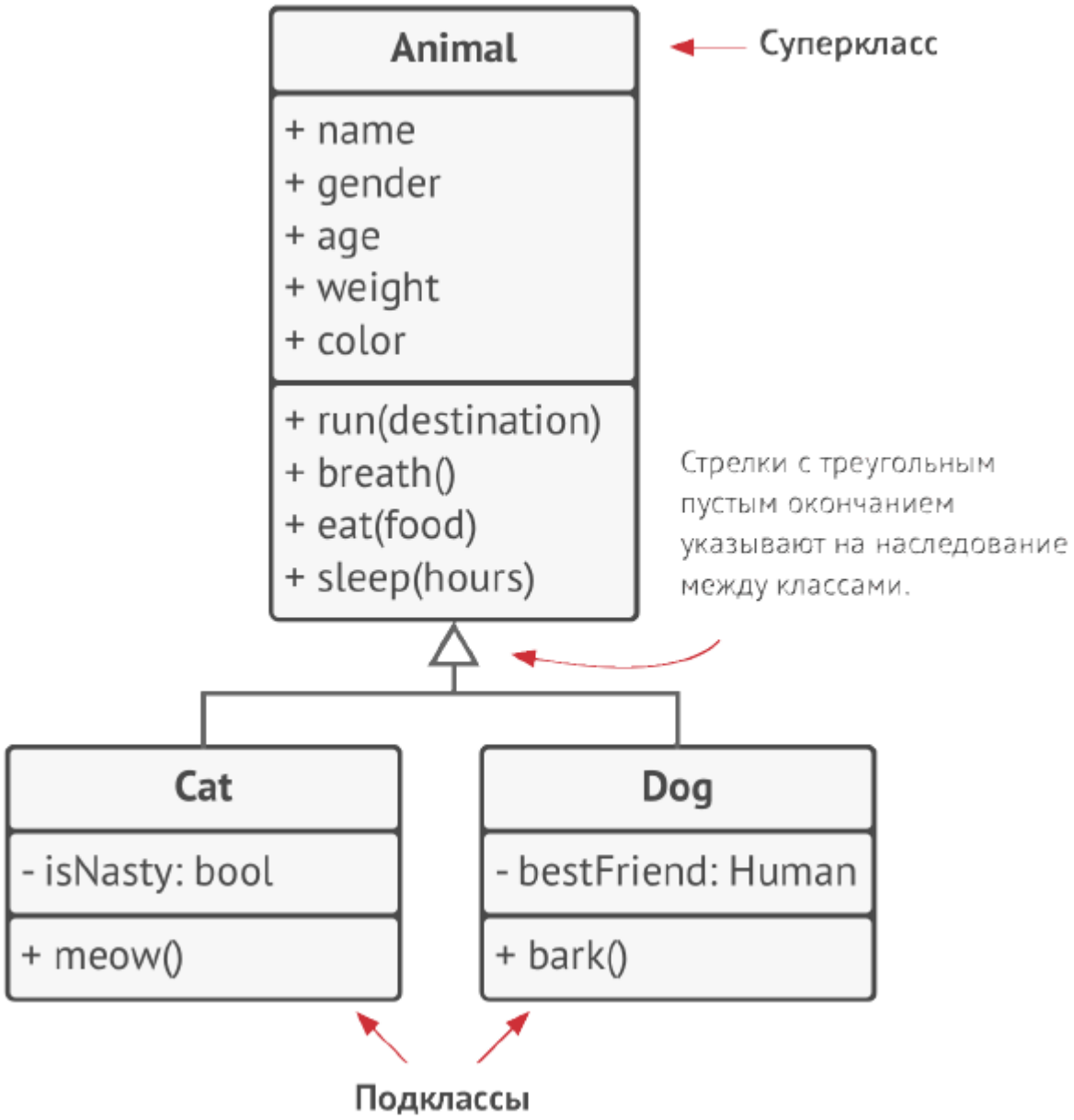
Отношения между классами -Обобщение

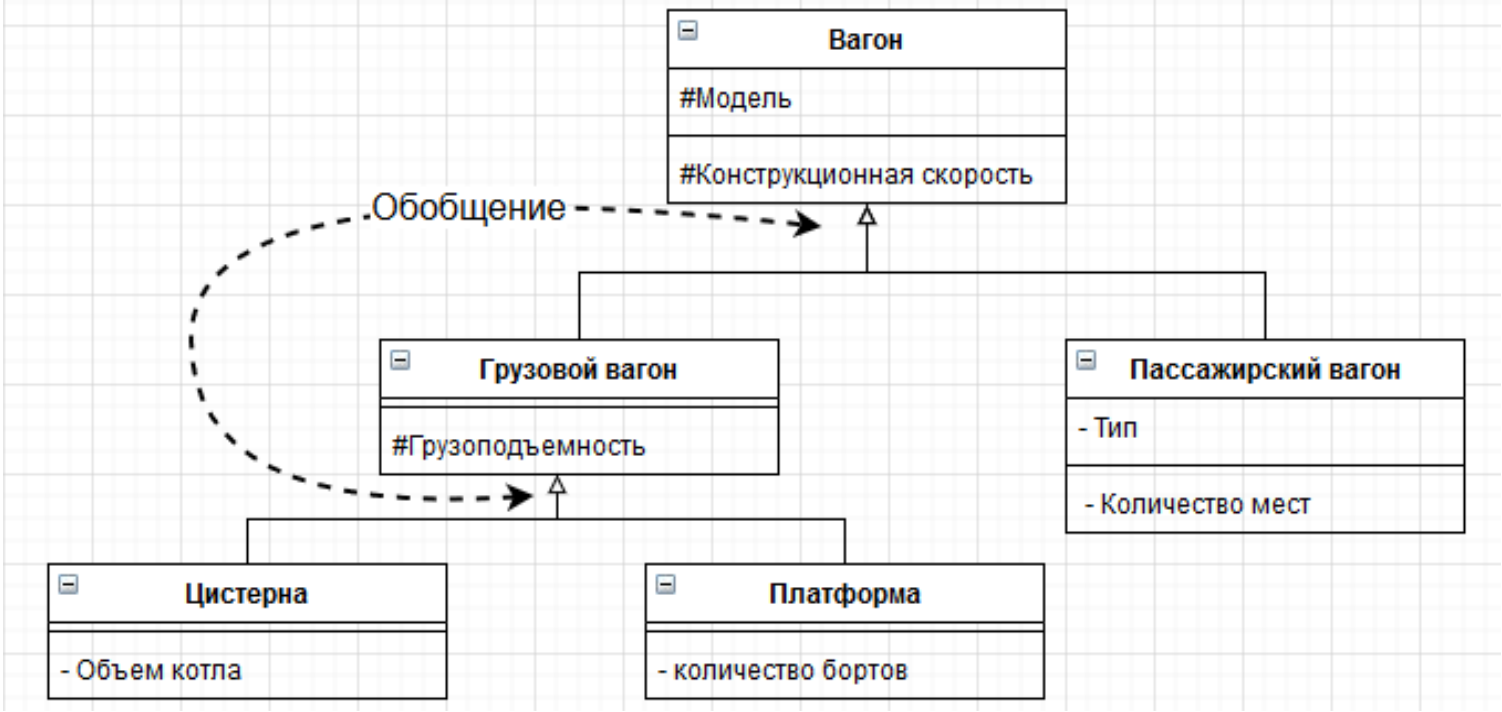
Обобщение допускает многоуровневую иерархию классов, но лучше ограничиться тремя уровнями, чтобы не усложнять модификацию программ, так как изменение суперкласса требует корректировки

Программируется обобщение при помощи механизма наследования. Суперкласс становится *базовым* классом (как правило, *абстрактным*), подклассы реализуются как *производные* классы.

Следует заметить, что абстрактный класс может иметь чистые виртуальные функции, следовательно, во всех его производных классах должны быть определены методы для реализации таких функций. В UML обобщение обозначается линией с треугольником на конце.

Вершина треугольника направлена в сторону суперкласса.


Animal
+ name
+ gender
+ age
+ weight
+ color
+ run(destination)
+ breath()
+ eat(food)
+ sleep(hours)
Суперкласс
Стрелки с треугольным пустым окончанием указывают на наследование между классами.
Cat
- isNasty: bool
+ meow()
Dog
- bestFriend: Human
+ bark()
Подклассы



Первый уровень иерархии на диаграмме содержит класс ***Вагон,*** являющийся ***суперклассом*** для двух классов второго уровня: ***Грузовой вагон*** и ***Пассажирский вагон.***

Оба класса наследуют от суперкласса ***Вагон*** защищенные атрибуты (*модель и конструкционная скорость*), а также имеют дополнительные.

Грузовой вагон служит суперклассом для классов третьего уровня иерархии: ***Цистерна*** и ***Платформа.*** К их закрытым атрибутам добавляются атрибуты классов ***Грузовой вагон*** и ***Вагон.***

Литература

Гради Буч, Джеймс Рамбо, Ивар Якобсон "Язык UML. Руководство пользователя"