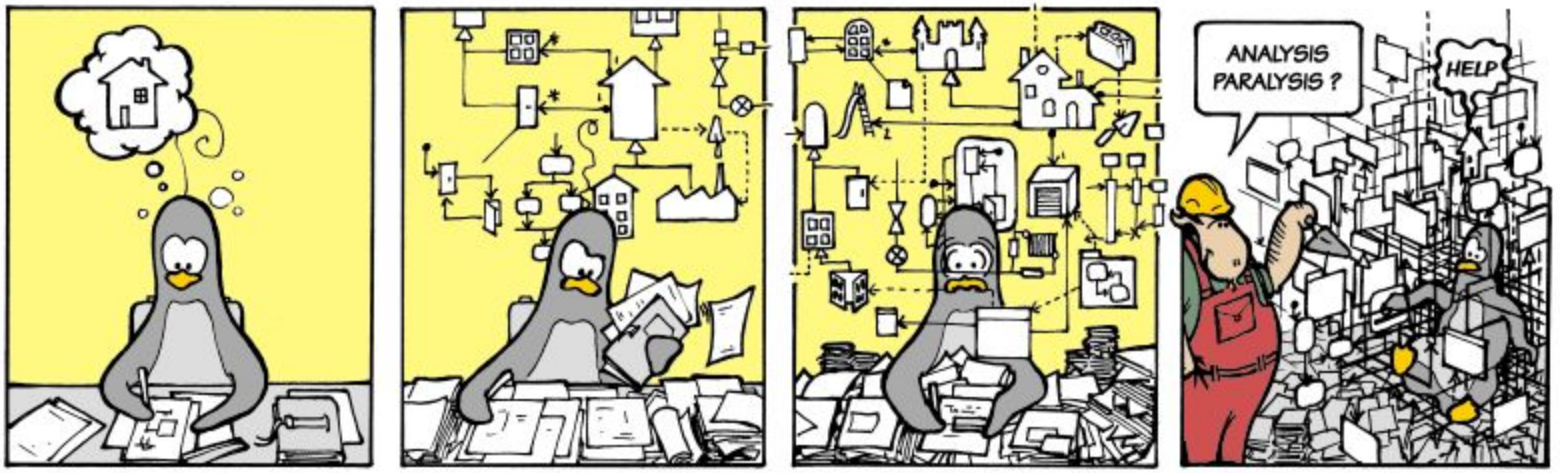
Diagrams



Diagrams

Почему нужно несколько видов диаграмм

Для начала определимся с терминологией. Автор уверен, что каждый из нас интуитивно понимает смысл этих понятий, но, чтобы внести полную ясность, снова заглянем в глоссарий и прочтем следующее:
Система - совокупность взаимосвязанных управляемых подсистем, объединенных общей целью функционирования. Да, не слишком информативно. А что же такое тогда подсистема?
Подсистема - это система, функционирование которой не зависит от сервисов других подсистем. Программная система структурируется в виде совокупности относительно независимых подсистем. Также определяются взаимодействия между подсистемами.
Тоже не слишком понятно, но уже лучше. Говоря "человеческим" языком, система представляется в виде набора более простых сущностей, которые относительно самодостаточны.
С понятием системы разобрались. В процессе проектирования система рассматривается **с разных точек зрения** с помощью моделей, различные представления которых предстают в форме диаграмм. Опять-таки у читателя могут возникнуть вопросы о смысле понятий модели и диаграммы.

Модель и диаграммы

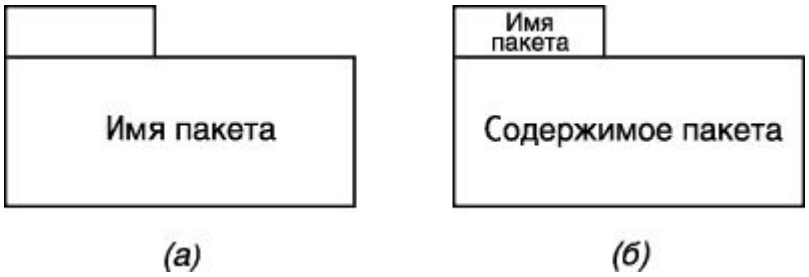
Модель - это некий (материальный или нет) объект, отображающий лишь наиболее значимые для данной задачи характеристики системы. Модели бывают разные - материальные и нематериальные, искусственные и естественные, декоративные и математические...
Осталось лишь сказать, что такое диаграмма. **Диаграмма** - это графическое представление множества элементов. Обычно изображается в виде графа с вершинами (сущностями) и ребрами (отношениями). Примеров диаграмм можно привести множество.
Но вернемся к проектированию ПО. В этой отрасли с **помощью диаграмм можно визуализировать систему с различных точек зрения**. Одна из диаграмм, например, может описывать взаимодействие пользователя с системой, другая - изменение состояний системы в процессе ее работы, третья - взаимодействие между собой элементов системы и т. д. Сложную систему можно и нужно представить в виде набора небольших и почти независимых моделей-диаграмм.
Ни одна отдельная диаграмма не является моделью. Диаграммы - лишь средство визуализации модели, и эти два понятия следует различать. Лишь **набор диаграмм составляет модель системы** и наиболее полно ее описывает, но не одна диаграмма, вырванная из контекста.

Уровни представления

Уровень представления — способ организации и рассмотрения модели на одном уровне абстракции, который представляет горизонтальный срез архитектуры модели, в то время как разбиение представляет ее вертикальный срез.
При этом исходная или первоначальная модель сложной системы имеет наиболее общее представление и относится к концептуальному уровню. Такая модель, получившая название концептуальной, строится на начальном этапе проектирования и может не содержать многих деталей и аспектов моделируемой системы. Последующие модели конкретизируют концептуальную модель, дополняя ее представлениями логического и физического уровня.

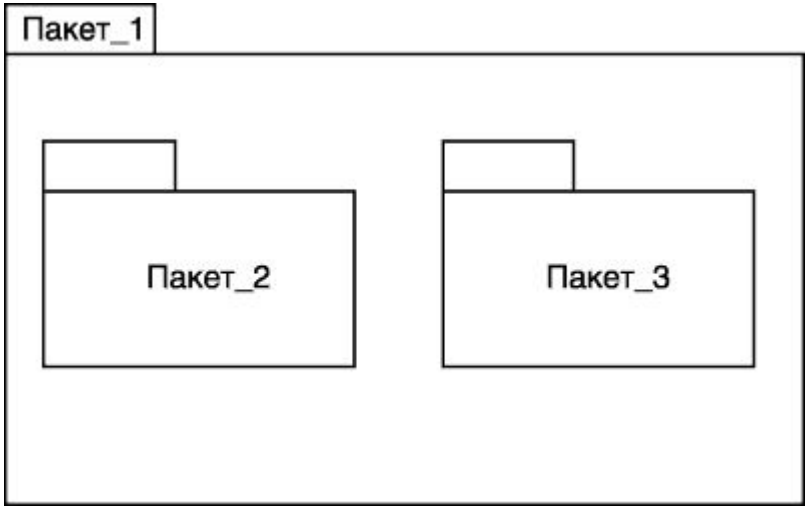
Пакеты

Пакет – основной способ организации элементов модели в языке UML. Каждый пакет владеет всеми своими элементами, т. е. теми элементами, которые включены в него. Про соответствующие элементы пакета говорят, что они принадлежат пакету или входят в него. При этом каждый элемент может принадлежать только одному пакету. В свою очередь, одни пакеты могут быть вложены в другие.
Подпакет (subpackage) — пакет, который является составной частью другого пакета. По определению все элементы подпакета принадлежат и более общему пакету. Тем самым для элементов модели задается отношение вложенности пакетов, которое представляет собой иерархию.



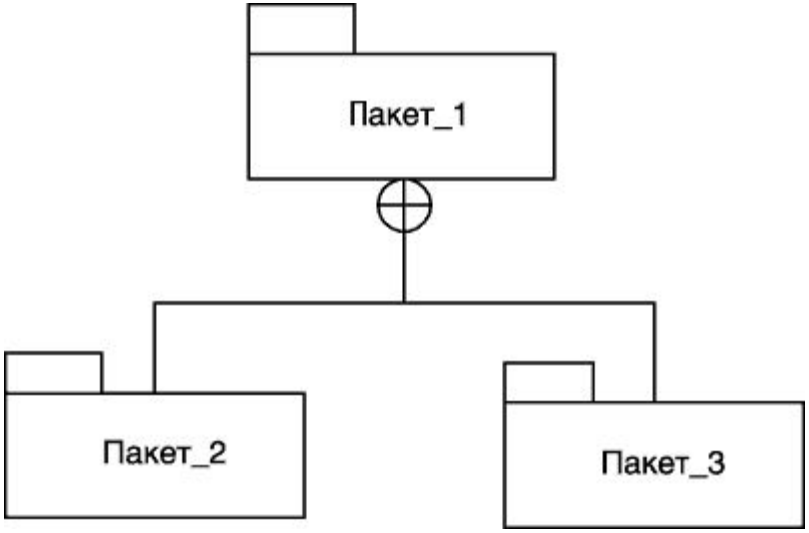
Пакеты

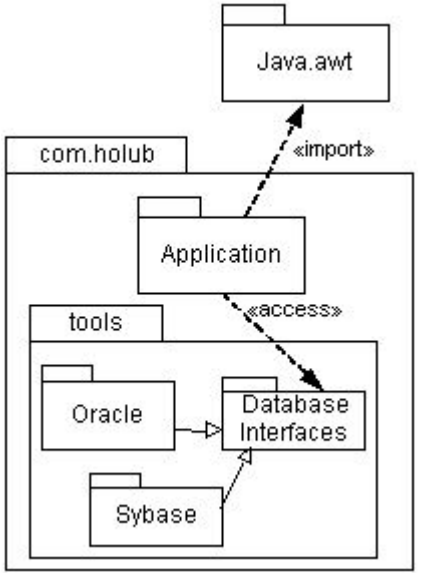
Одним из типов отношений между пакетами является отношение вложенности или включения пакетов друг в друга. В языке UML это отношение может быть изображено без использования линий простым размещением одного пакета-прямоугольника внутри другого пакета-прямоугольника. Так, в данном случае пакет с именем Пакет_1 содержит в себе два подпакета: Пакет_2 и Пакет_3.



Пакеты

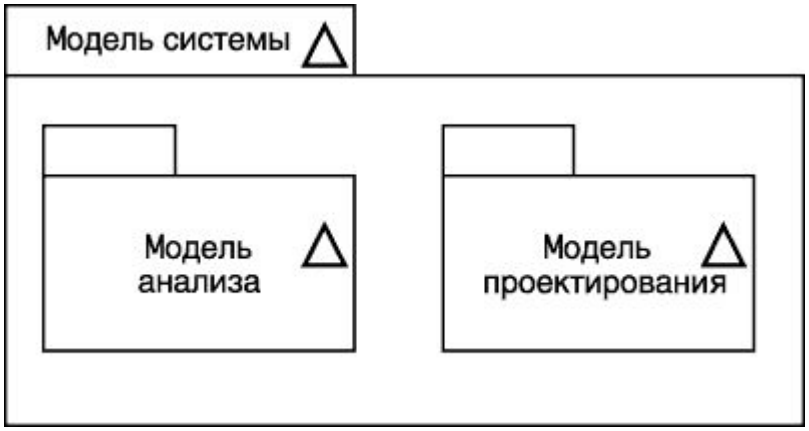
Кроме того в языке UML это же отношение может быть изображено с помощью отрезков линий аналогично графическому представлению дерева. В этом случае наиболее общий пакет или контейнер изображается в верхней части рисунка, а его подпакеты – уровнем ниже. Контейнер соединяется с подпакетами сплошной линией, на конце которой, примыкающей к контейнеру, изображается специальный символ. Он означает, что подпакеты "собственность" или часть контейнера, и, кроме этих подпакетов, контейнер не содержит никаких других.





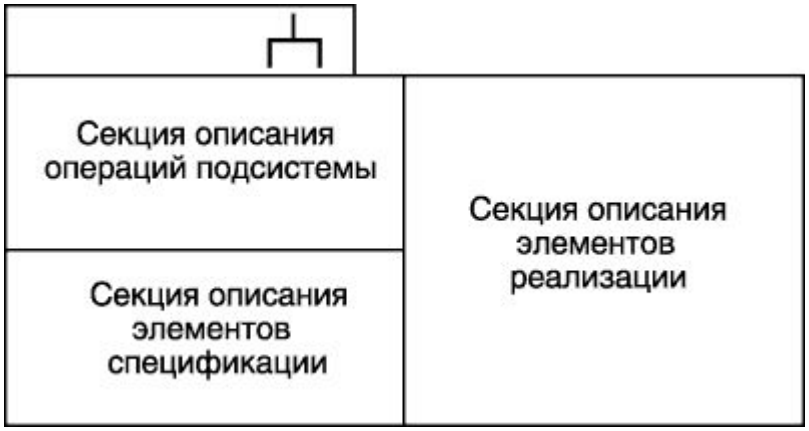
Модель

Модель является подклассом пакета и представляет собой абстракцию физической системы, которая предназначена для вполне определенной цели. Именно эта цель предопределяет те компоненты, которые должны быть включены в модель и те, рассмотрение которых не является обязательным. Другими словами, модель отражает релевантные аспекты физической системы, оказывающие непосредственное влияние на достижение поставленной цели. В прикладных задачах цель обычно задается в форме исходных требований к системе, которые, в свою очередь, в языке UML записываются в виде вариантов использования системы.



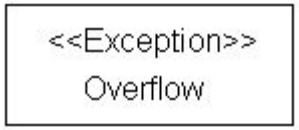
Подсистема

Подсистема есть просто группировка элементов модели, которые специфицируют простейшее поведение физической системы. Для графического представления подсистемы применяется специальное обозначение – прямоугольник, как в случае пакета, но дополнительно разделенный на три секции. При этом в верхнем маленьком прямоугольнике изображается символ, по своей форме напоминающий "вилку" и указывающий на подсистему. Имя подсистемы вместе с необязательным ключевым словом или стереотипом записывается внутри большого прямоугольника. Однако при наличии строк текста внутри большого прямоугольника имя подсистемы может быть записано рядом с обозначением "вилки".



Механизмы расширения

Стереотип (stereotype) — новый тип элемента модели, который расширяет семантику метамодели. Стереотипы должны основываться на уже существующих и описанных в метамодели языка UML типах или классах. Стереотипы предназначены для расширения именно семантики, но не структуры уже описанных типов или классов. Некоторые стереотипы предопределены в языке UML, другие могут быть указаны разработчиком. На диаграммах изображаются в форме текста, заключенного в угловые кавычки. Предопределенные стереотипы являются ключевыми словами языка UML, которые используются на канонических диаграммах на языке оригинала без их перевода.



Механизмы расширения

Помеченное значение — явное определение свойства как пары "имя – значение". В помеченном значении само имя называют тегом (tag).

Помеченные значения на диаграммах изображаются в форме строки текста специального формата, заключенного в фигурные скобки. При этом используется следующий формат записи: тег = значение. Теги встречаются в нотации языка UML, но их определение не является строгим, поэтому теги могут быть указаны самим разработчиком.

Ограничение (constraint) — некоторое логическое условие, ограничивающее семантику выбранного элемента модели.

Как правило, все ограничения специфицируются разработчиком. Ограничения на диаграммах изображаются в форме строки текста, заключенного в фигурные скобки. Для формальной записи ограничений предназначен специальный язык объектных ограничений (Object Constraint Language, OCL), который является составной частью языка UML.

Канонические диаграммы языка UML

Диаграмма — графическое представление совокупности элементов модели в форме связного графа, вершинам и ребрам (дугам) которого приписывается определенная семантика. Нотация канонических диаграмм - основное средство разработки моделей на языке UML. В нотации языка UML определены следующие виды канонических диаграмм:

- вариантов использования (use case diagra
- классов (class diagram)
- кооперации (collaboration diagram)
- последовательности (sequence diagram)
- состояний (statechart diagram)
- деятельности (activity diagram)
- компонентов (component diagram)
- развертывания (deployment diagram)

Диаграмма вариантов использования

Вариант использования - описание отдельного аспекта поведения системы с точки зрения пользователя (Буч).

Диаграмма классов

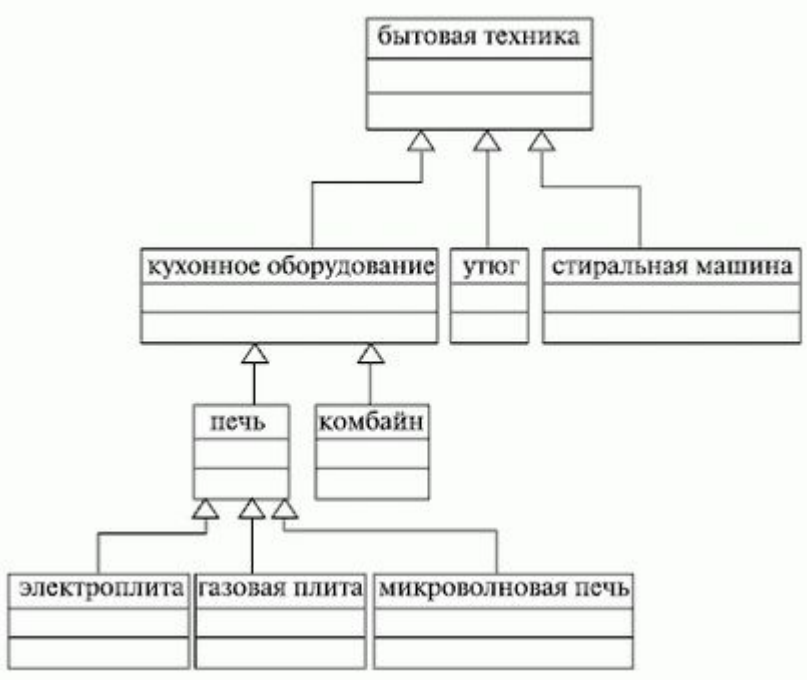


Диаграмма объектов

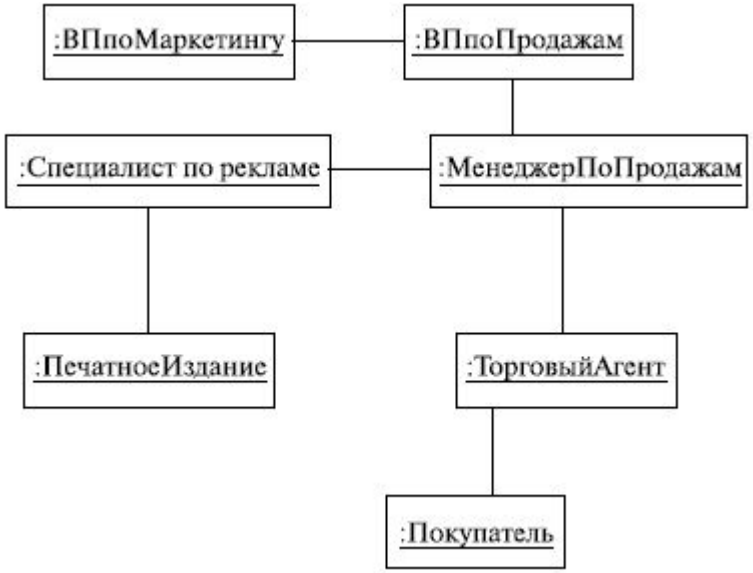


Диаграмма последовательностей

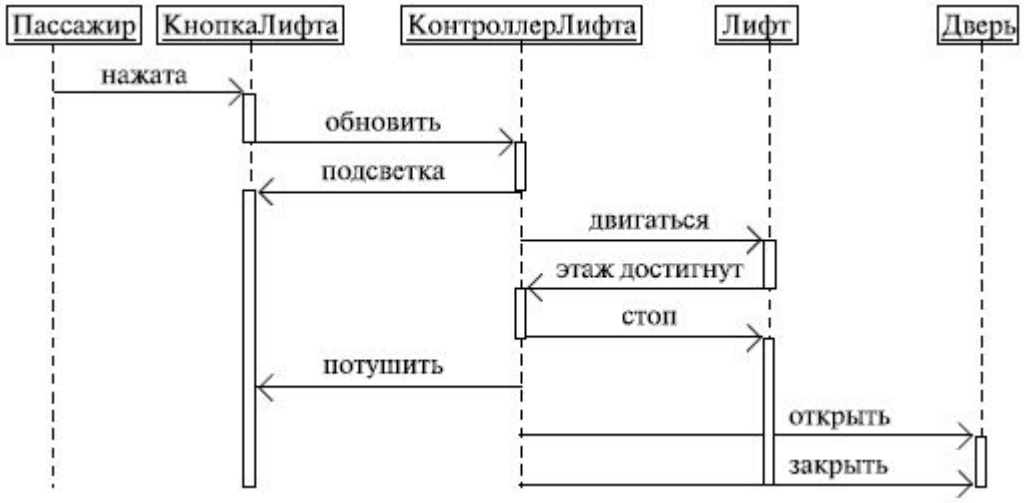


Диаграмма взаимодействия (кооперации)

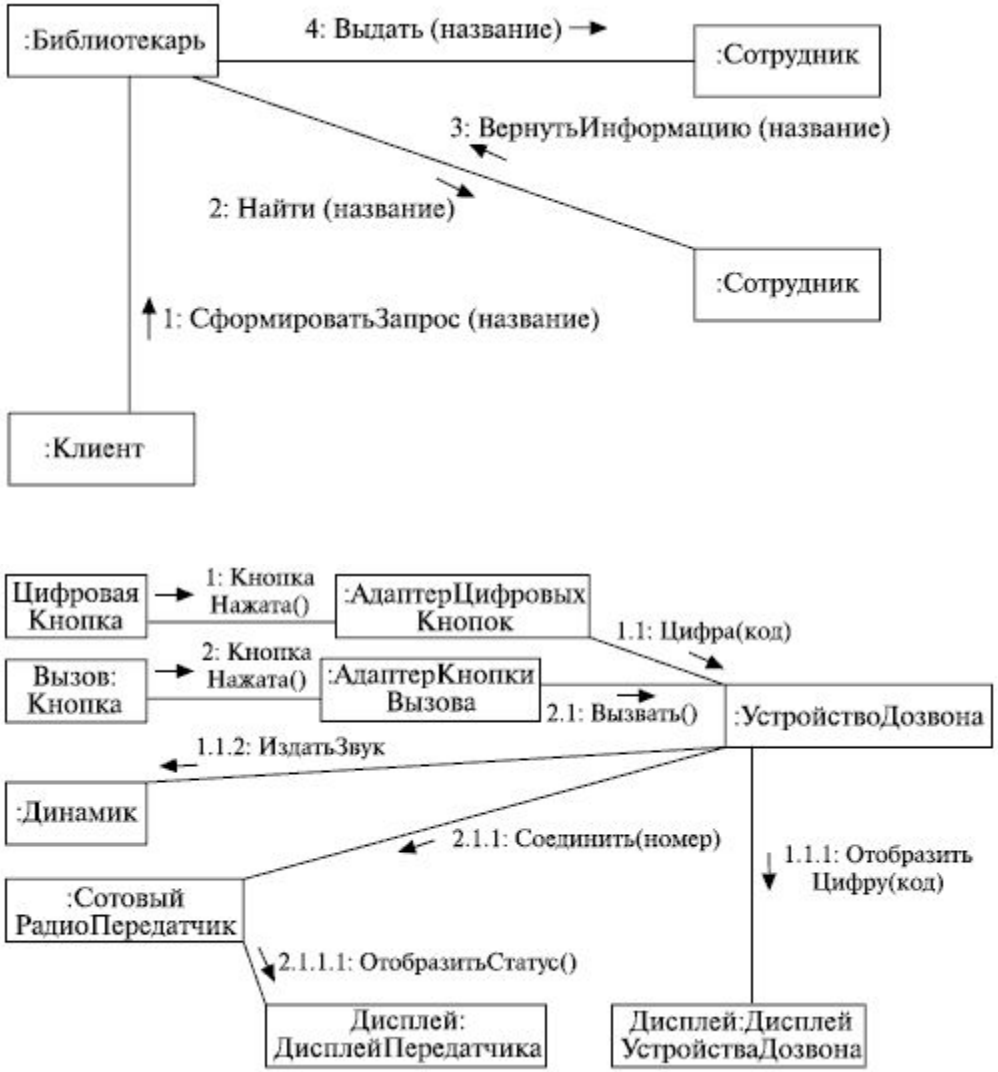
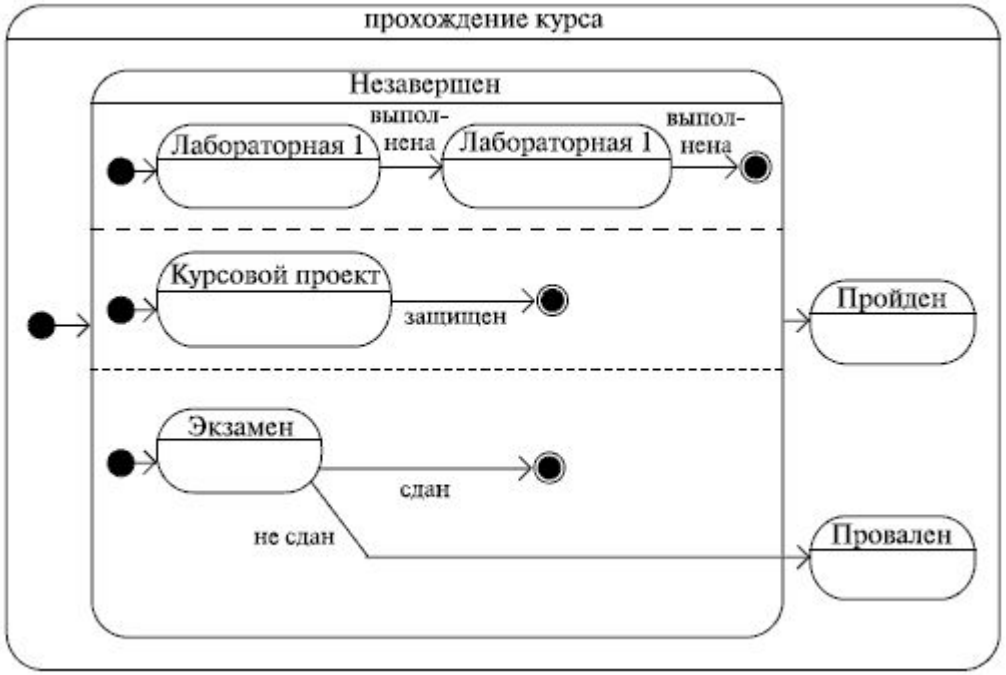


Диаграмма состояний



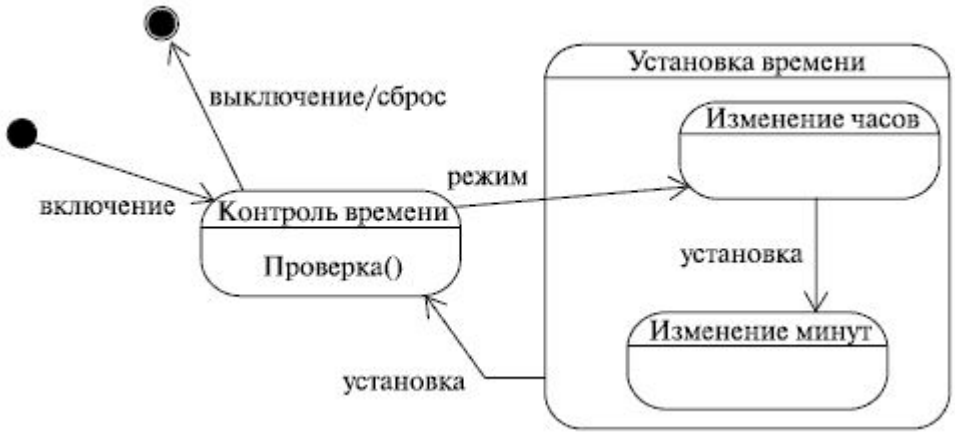


Диаграмма активности (деятельности)

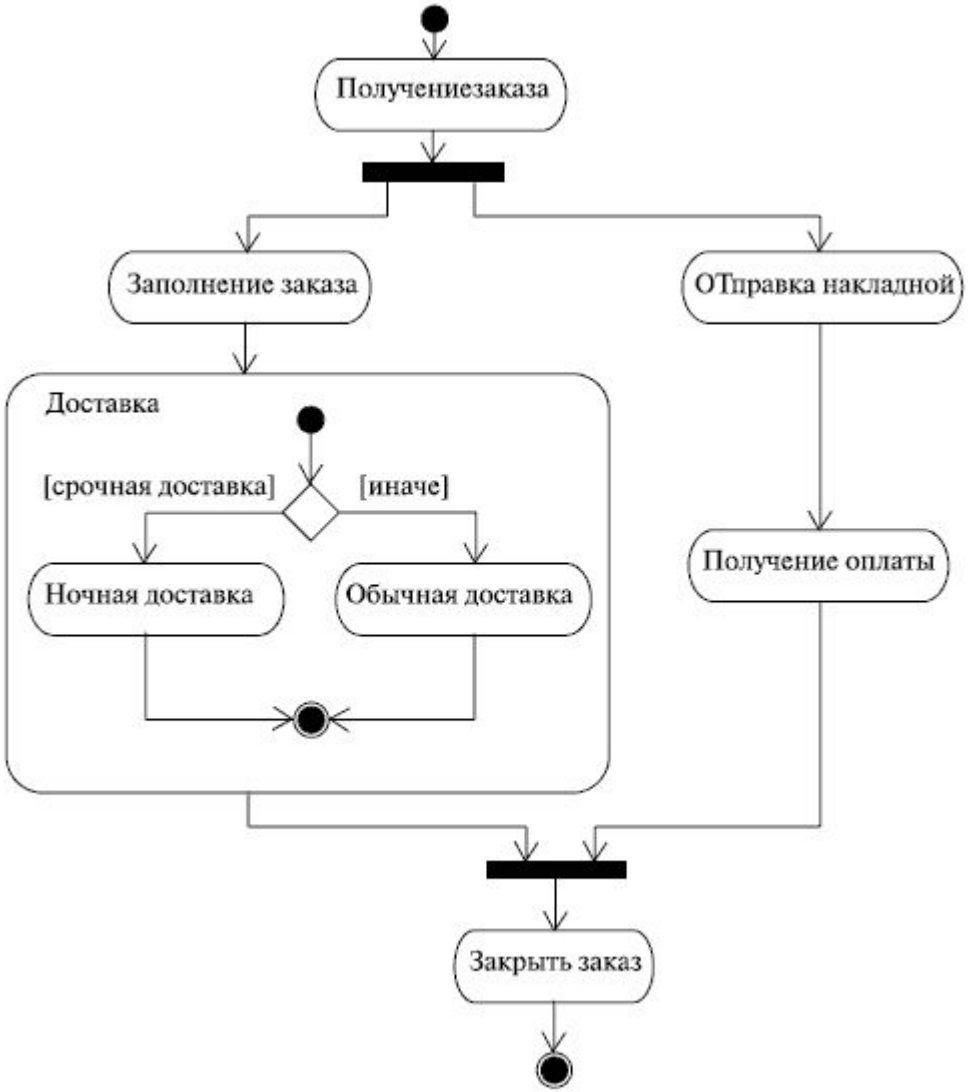
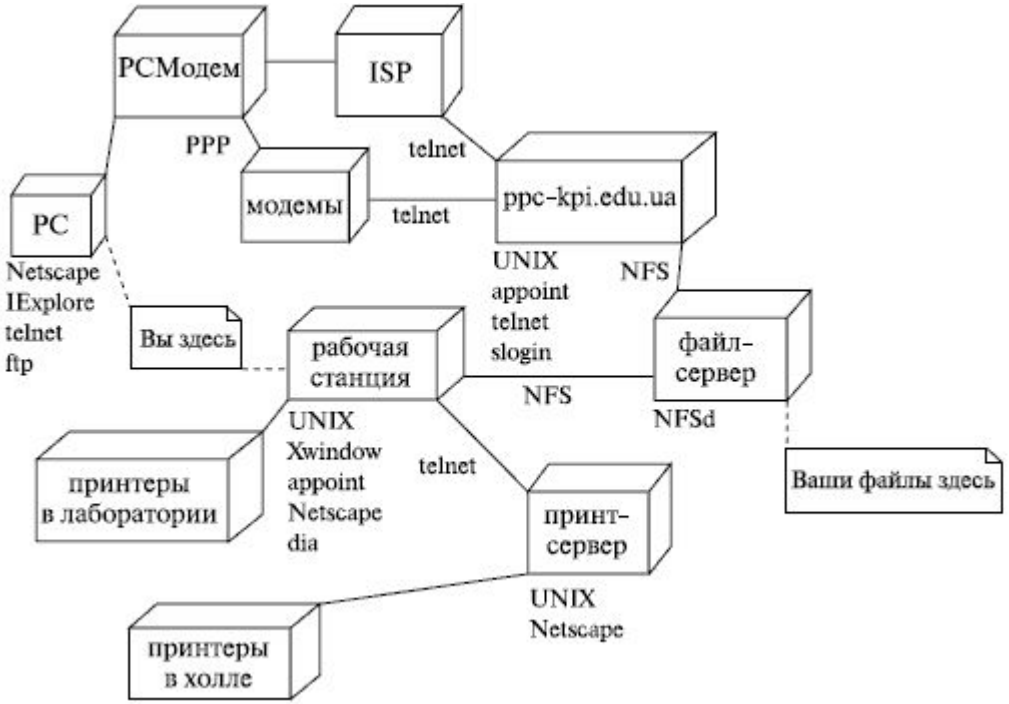
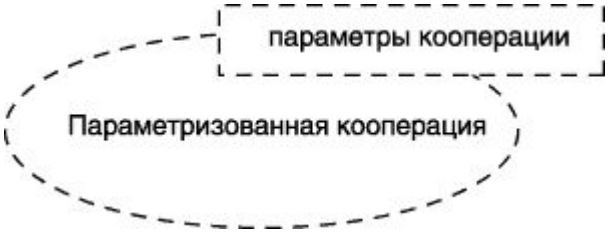


Диаграмма развертывания



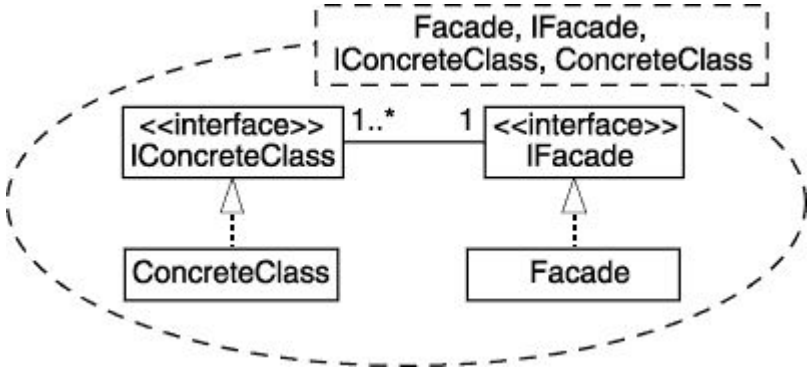
Паттерны проектирования и UML

В сфере разработки программных систем наибольшее применение получили паттерны проектирования GoF, некоторые из них реализованы в популярных средах программирования. При этом паттерны проектирования могут быть представлены в наглядной форме с помощью рассмотренных обозначений языка UML.
Паттерн проектирования в контексте языка UML представляет собой параметризованную кооперацию вместе с описанием базовых принципов ее использования. При изображении паттерна используется обозначение параметризованной кооперации языка UML, которая обозначается пунктирным эллипсом. В правый верхний угол эллипса встроен пунктирный прямоугольник, в котором перечислены параметры кооперации, которая представляет тот или иной паттерн.



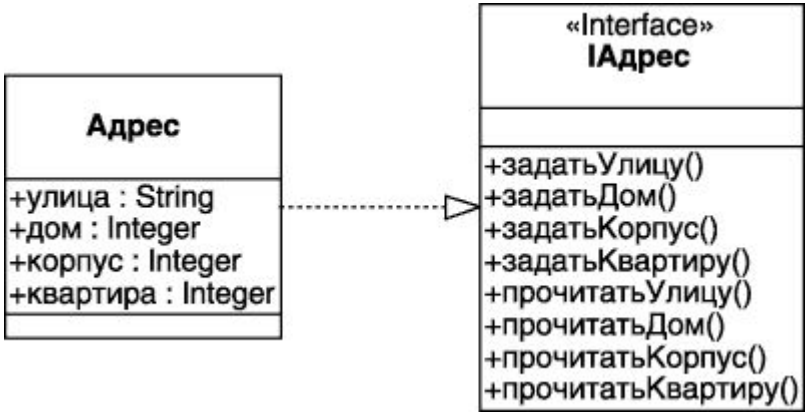
Паттерн Фасад и его обозначение в нотации языка UML

Паттерн Фасад предназначен для замены нескольких разнотипных интерфейсов доступа к определенной подсистеме некоторым унифицированным интерфейсом, что существенно упрощает использование рассматриваемой подсистемы. Общее представление *паттерна проектирования* Фасад может быть изображено с помощью следующей диаграммы параметризованной кооперации

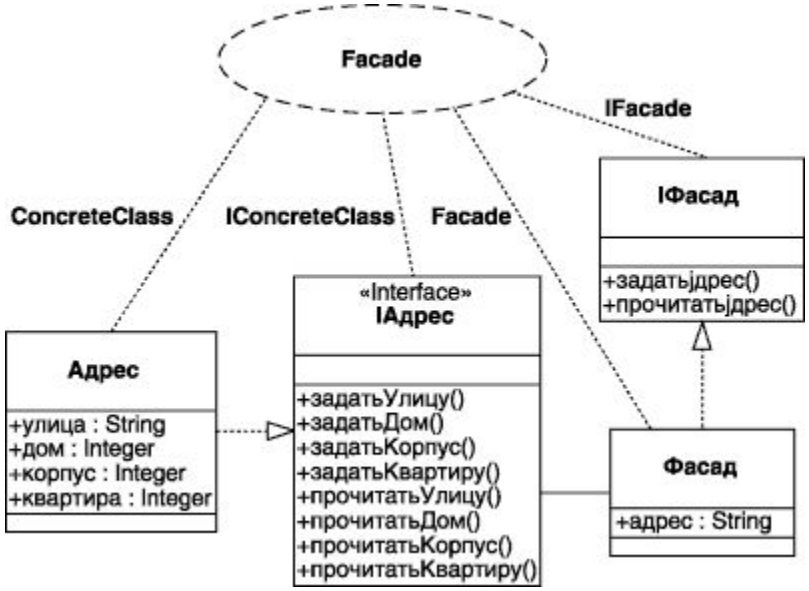


Изображенная параметризованная кооперация содержит 4 параметра: класс Facade (Фасад), интерфейс IFacade, интерфейсы IConcreteClass и конкретные классы ConcreteClass, в которых реализованы интерфейсы IConcreteClass. Пунктирная линия со стрелкой в форме треугольника служит для обозначения отношения реализации (не путать с отношением обобщения классов).

Паттерн Фасад и его обозначение в нотации языка UML



Фрагмент диаграммы классов до применения паттерна Фасад



Конкретная реализация паттерна проектирования Фасад

Паттерн Фасад и его обозначение в нотации языка UML

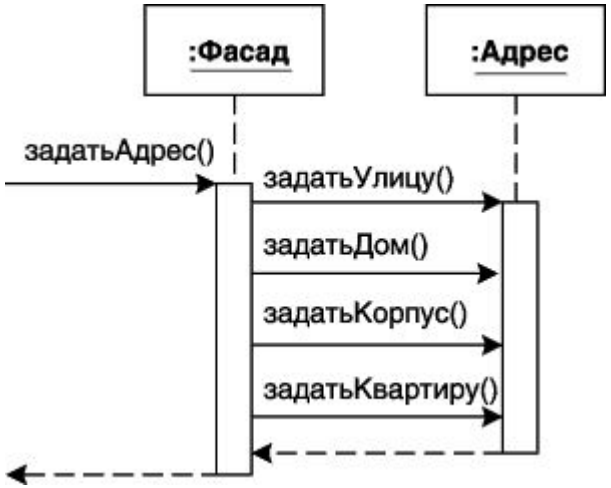


Диаграмма последовательности для выполнения операции задания адреса