

Введение в UML

**UML -
Uniform (Унифицированный)
Modeling (Язык)
Language (Моделирования)**

UML является визуальным языком моделирования, который позволяет системным архитекторам представлять своё видение системы в стандартной и лёгкой для понимания форме. UML предоставляет эффективный механизм совместного использования проектных решений и взаимодействия разработчиков друг с другом.

Терминология:

- Диаграмма - графическое изображение элементов и связи между ними.
- Система - комбинация программных и аппаратных средств, которые обеспечивают выполнение поставленной задачи.
- Разработка системы представляет собой процесс её создания для клиента, т.е. человека которому необходимо решить какую-то проблему.

Польза UML

Аналитик разрабатывает документы с описанием этой проблемы и передаёт их разработчикам - программистам, которые создают программное обеспечение для решения требуемой задачи и гарантируют его развёртывание на аппаратных средствах.

Сформулировать видение системы - чрезвычайно важный момент. Раньше анализ проводился "на пальцах". В настоящее время ключевым моментом процесса разработки является хорошо продуманный план. План, в свою очередь, должен составляться лишь после тщательного анализа требований клиента.

Ключевым аспектом процесса проектирования является его правильная организация, когда аналитики, клиенты, программисты и другие специалисты, участвующие в разработке системы, способны понять друг друга и прийти к общему мнению.

Ещё одной отличительной чертой процесса разработки современных систем является дефицит времени для выполнения работ. Если предельные сроки сдачи

подсистем нагромождаются друг на друга, то обеспечение непрерывности процесса разработки становится жизненно важной необходимостью.

Другой аспект современной жизни - слияние корпораций - также предъявляет свои требования к процессу разработки. Когда одна компания приобретает другую, новая организация должна ввести изменения в используемый процесс разработки.

История.

Авторами UML являются Grady Booch, James Rumbaugh & Ivar Jacobson. В 80-х, в начале 90-х они независимо друг от друга придумывали методологии объектно-ориентированного анализа и проектирования. Потом в 94-95 годах они втроем оказались в Rational Software Corporation.

Остальное - история. Предварительные версии UML начали использоваться в области создания программного обеспечения, а на основании отзывов потребителей производились существенные доработки. Это привело к возникновению консорциума UML(DEC, Hewlett-Packard, Microsoft, Oracle, Ration,...). В 1997 году консорциум выбрал превью версию UML и представил её на рассмотрение OMG.

В конце 1997 года вышла версия 1.0. После этого группа OMG приступила к сопровождению и выпустила в конце 1988 года его две новые версии. Язык UML стал стандартом де-факто в области разработки программного обеспечения. В настоящее время язык продолжает активно развиваться. В 2002 году вышла версия 2.0, которая считается текущей.

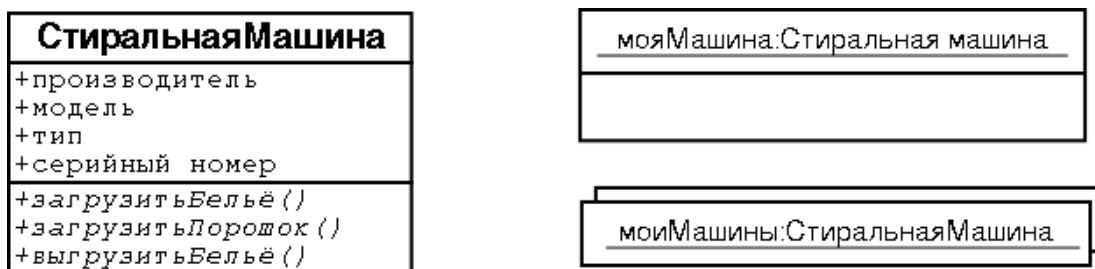
Диаграммы.

Язык UML включает набор графических элементов, используемых на диаграммах. Будучи языком, UML содержит правила для объединения этих элементов. Диаграммы используются отображения различных представлений системы. Этот набор различных представлений называется моделью. Модель UML системы можно сравнить с художественно оформленной моделью здания. Важно отметить, что модель UML описывает, что должна будет делать система. В то же время, ничего не сообщается, как она будет реализована.

Вообще, при создании модели используется что-то хорошо известное, для того, чтобы понять что-то менее известное.

Диаграмма классов.

Легко можно увидеть, что все окружающие нас вещи различаются по категориям(автомобиди, мебель, стир. машины). Мы обращаемся к этим категориям, как к классам. Класс - это категория или группа вещей, которая имеет сходные атрибуты и общие свойства.



Диаграммы классов представляют собой отправную точку процесса разработки.

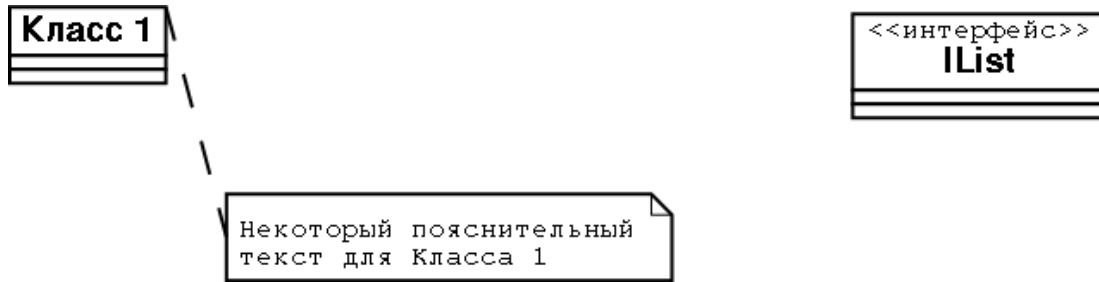
Диаграмма объектов.

Объект представляет собой экземпляр класса - особую сущность, которая имеет заданные значения атрибутов и операций. Если на примере стиральной машины, то её атрибуты могут иметь вид: компания-производитель - "Сантехника", наименование модели - "Мойдодыр", серийный номер - "13-666-13" и ёмкость - 16 фунтов.

Расширения языка:

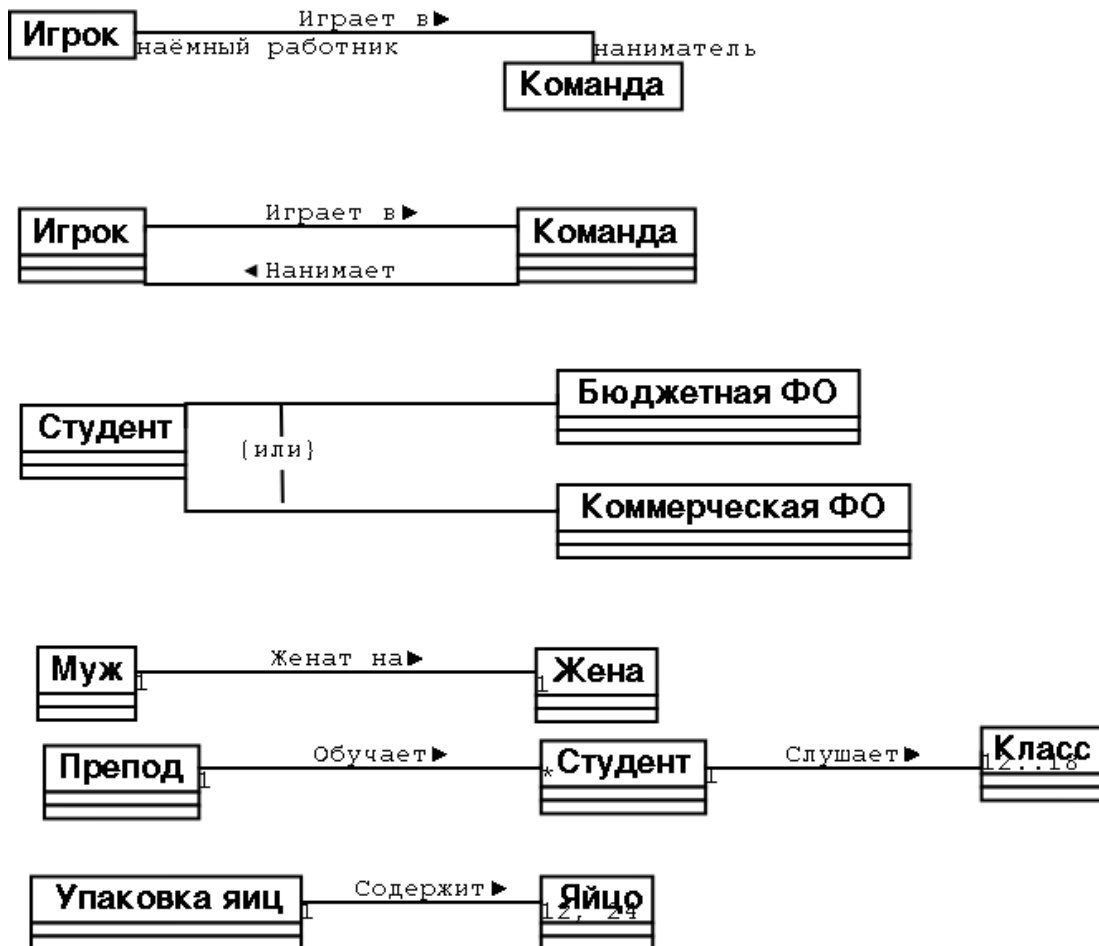
Примечания служат для пояснения, почему эта часть диаграммы расположена именно здесь и как с ней работать. Стереотипы позволяют использовать существующие элементы UML и преобразовывать. Хороший пример - концепция

интерфейса, т.е. класса не имеющего атрибутов.

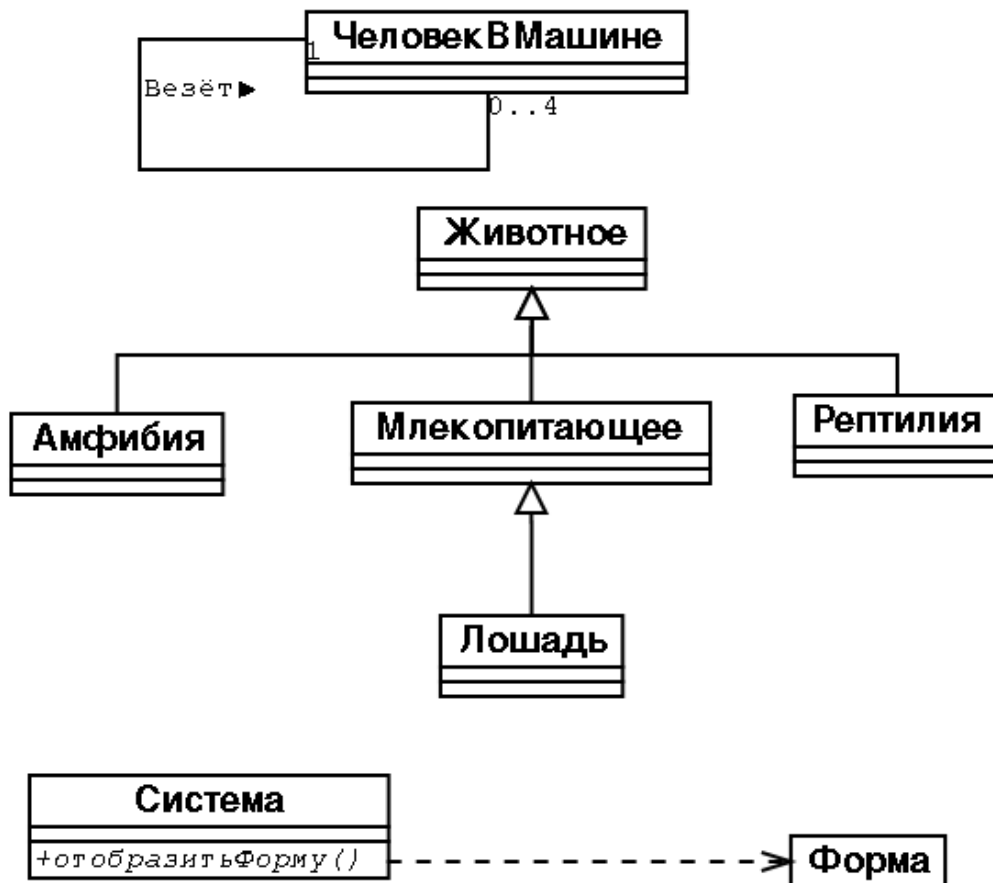
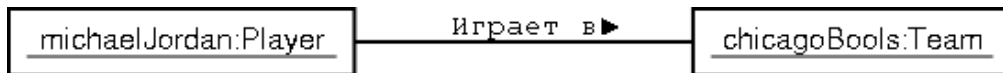


Ассоциации.

Если классы концептуально взаимодействуют друг с другом, то это взаимодействие называется ассоциацией. Ассоциации имеют: ограничения(например {по очереди}, {или}), квалификатор(доп. информация в отношении "один ко многим"), классы, кратность, может быть рефлексивной.

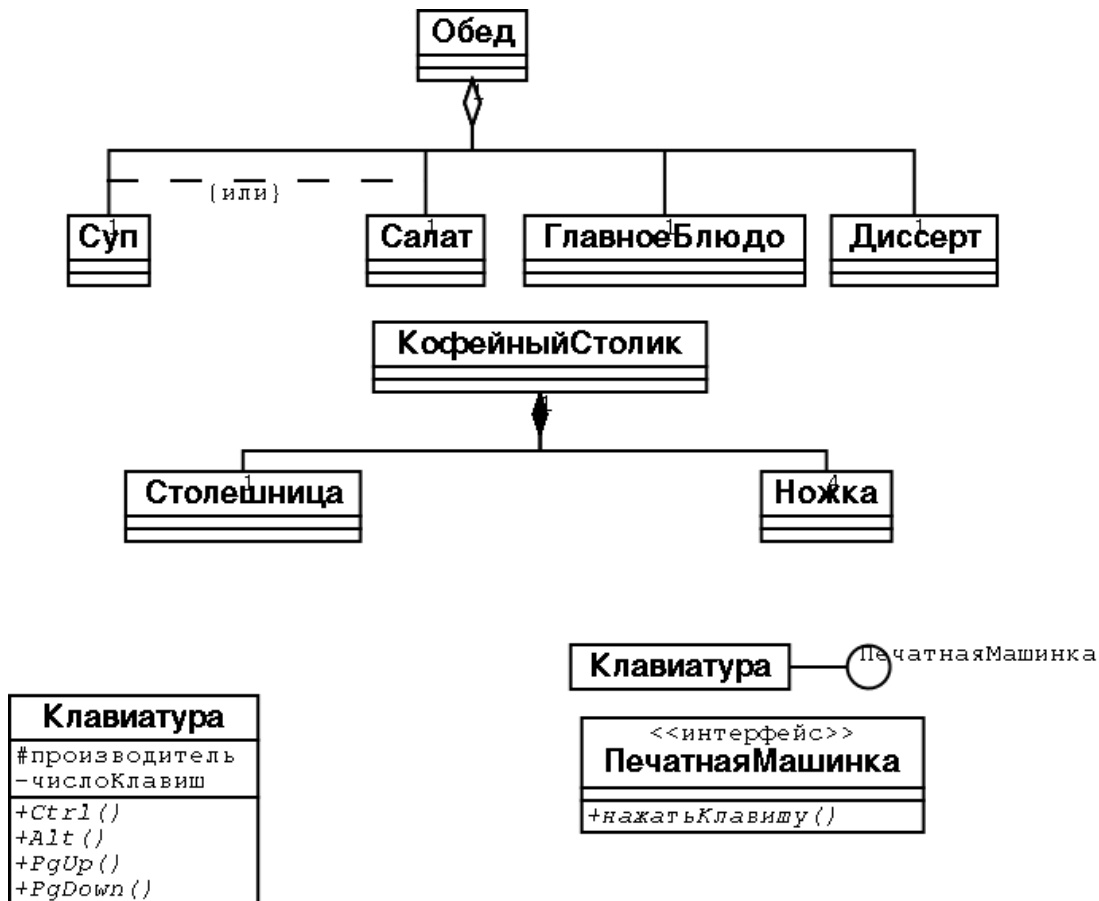


Может быть наследованием, зависимостью.



Агрегация, композитные объекты, интерфейсы и реализации.

Иногда класс состоит из некоторого количества классов-компонентов. Это особый тип взаимосвязи, называемый агрегацией. Обозначается незакрашенным ромбом. Можно использовать {или} для того, чтобы показать выбор. Агрегация задаёт отношение "часть-целое".



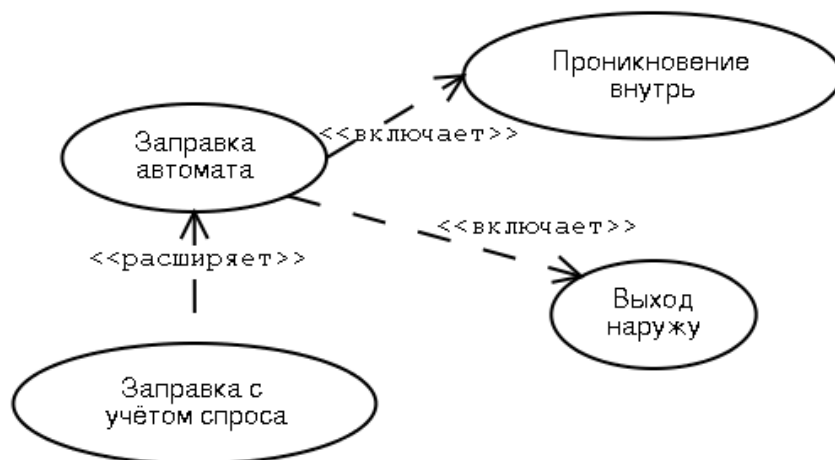
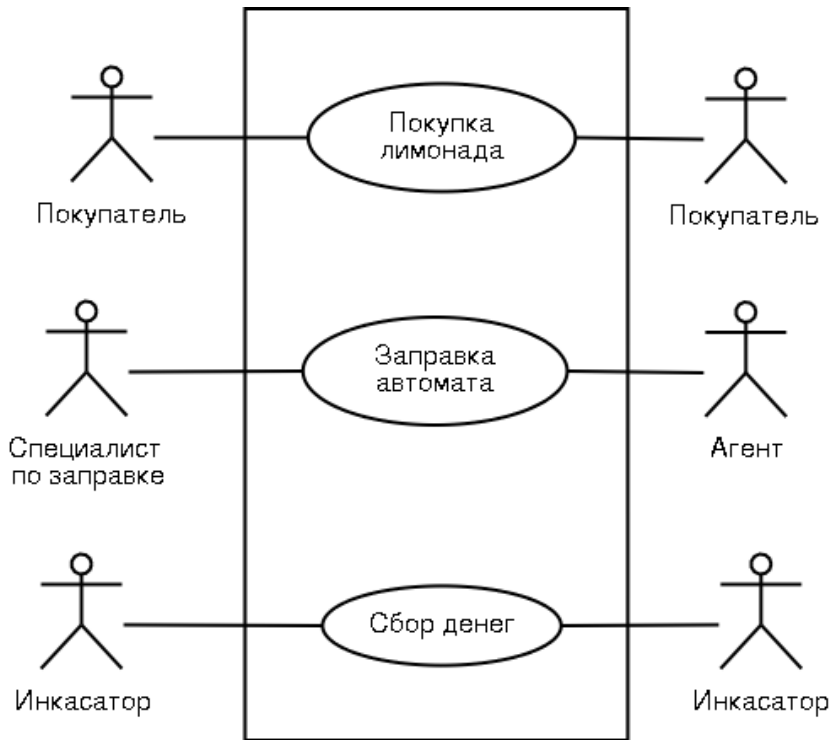
Композит - это строгий тип агрегации, характеризующийся тем, что каждый элемент может принадлежать только одному целому. Можно выделить набор повторно используемых операций и объединить их в группу. Существует два вида обозначения интерфейса: кружок и стереотип. Термин видимость применяется по отношению к атрибутам и операциям и задаёт типы других классов, которые могут пользоваться ими.

- Открытая область "+" - все могут использовать.
- Защищённая область "#" - используется только наследниками.
- Закрытая область "-" - только класс имеет доступ.
- Реализация предполагает открытость операций интерфейса.
- Статические атрибуты и операции (присущие классу, единые для всех объектов данного класса) подчёркиваются.

Диаграмма прецедентов(Use case)

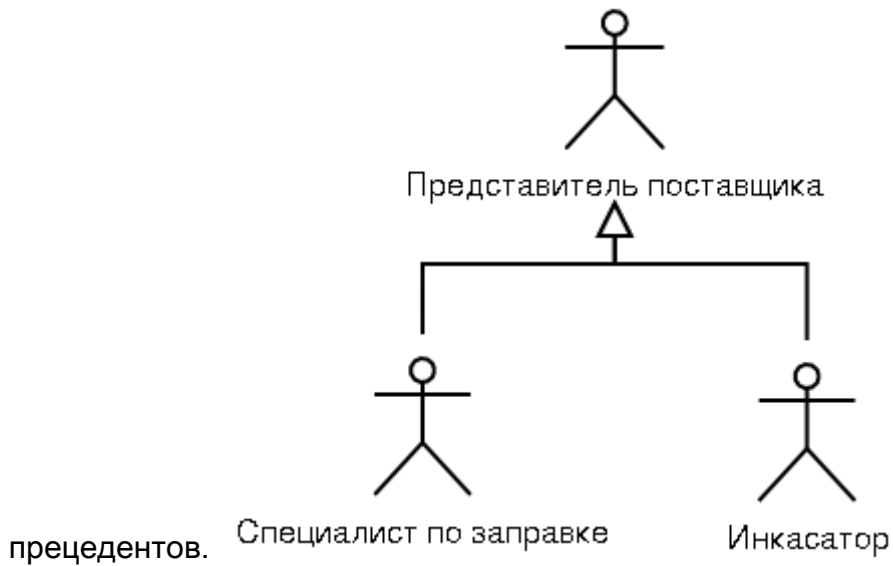
Прецедент - коллекция сценариев использования системы. Каждая последовательность действий инициируется другой системой, пользователем и т.д. в какой-то момент времени. Сущности, инициирующие сценарии называются

исполнителями. Прецеденты можно использовать повторно. Один способ включение, другой расширение.



Прецеденты можно обобщать(наследовать) подобно классам. При наследовании дочерний прецедент прибавляет к родительскому свои шаги. Исполнители тоже

могут наследоваться. В начале, очень важно создать высокоуровневую диаграмму

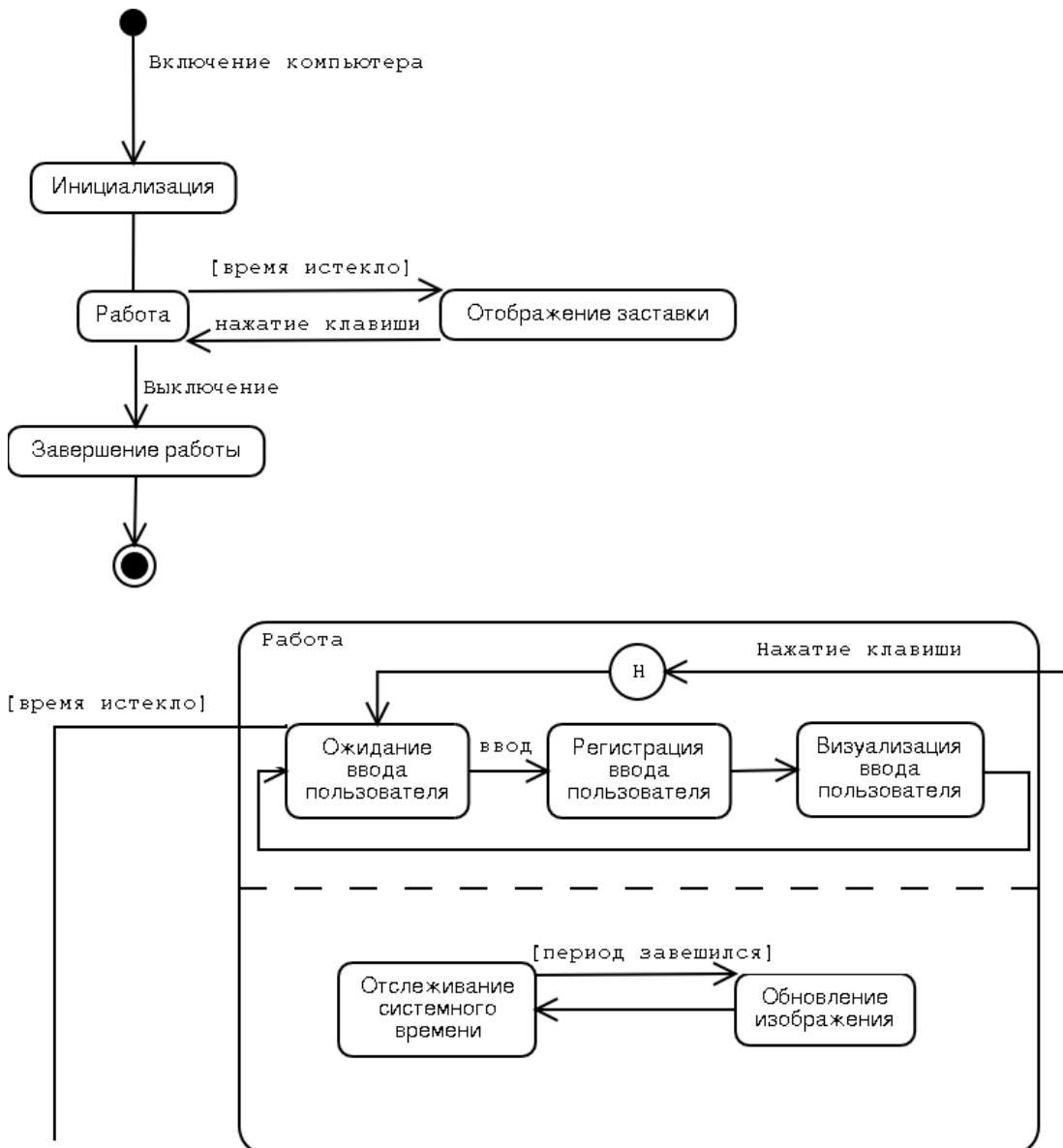


Диаграммы состояний.

Объекты меняют своё состояние в ответ на происходящие события и течением времени. Диаграмма состояний представляет состояния объекта и переходы между ними, а также начальное и конечное состояние объекта.

Переменные состояния такие, как таймеры и счётчики иногда бывают полезны.

Виды деятельности включают события и действия. К стандартным видам деятельности относятся вход (что происходит, когда система заходит в состояние), выход (что происходит, когда система выходит из состояния), выполнение (что происходит, когда система находится в состоянии).



К линиям перехода можно добавить дополнительные детали. Указывают событие, вызвавшее переход. Состояния могут быть сложными. Они могут внутри содержать подсостояния(и даже группы, работа в которых происходит параллельно). Такие состояния называют композитными. Кроме того, композитное состояние может запомнить его состояние перед выходом из него. Это обозначается буквой "H"(History) в кружке. Для глубокой истории, где запоминаются все подчинённые состояния(во вложенных в данное) есть обозначение "H**".

Событие, обеспечивающее переход на диаграмме состояний объекта получателя называется сигналом. Сигналы, как и любые классы, можно наследовать. Переход может быть условным(по событию) или безусловным(по выполнению всех

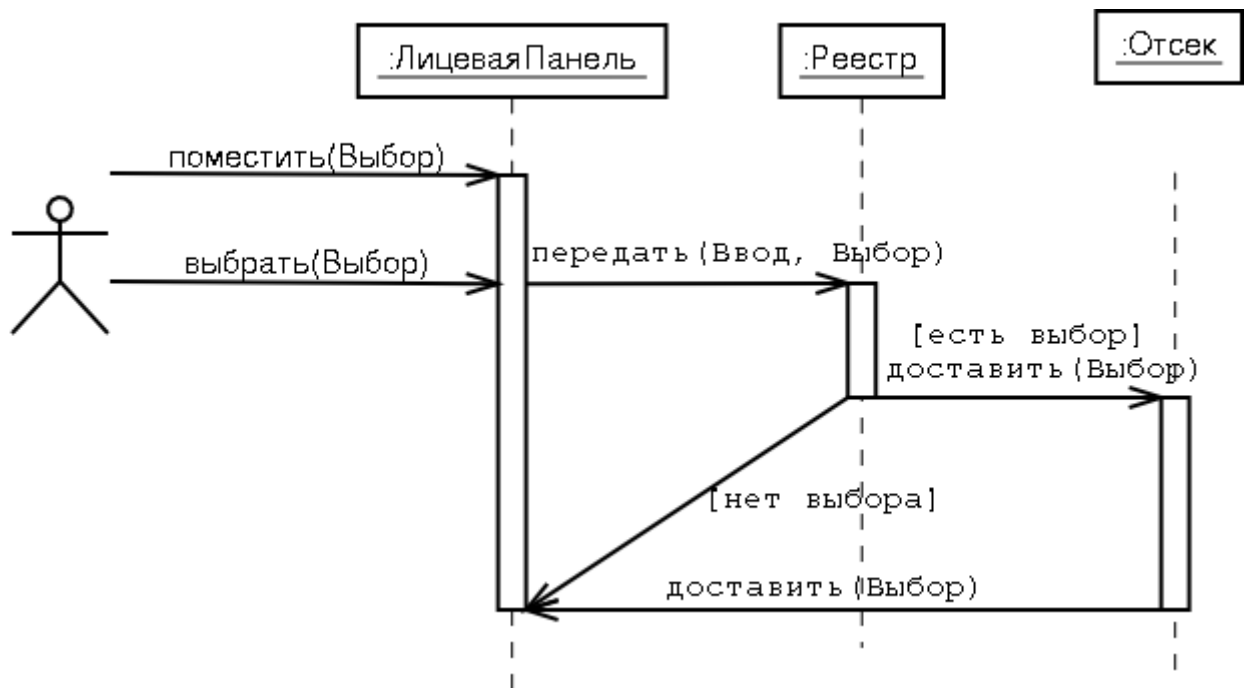
действий).



На диаграмме состояний отображаются все переходы между состояниями одного объекта системы. Диаграммы дают полную информацию аналитикам и разработчикам о желаемом поведении.

Диаграммы последовательностей.

Диаграмма последовательности состоит из обычных объектов, сообщений (в виде стрелок), а также вертикальной оси времени. Сообщение может быть простым (передача управления), синхронным (ожидает ответа), асинхронным (не ожидает ответа, продолжает действовать). Пример: 1. Покупатель помещает монету в щель на лицевой панели автомата. 2. Покупатель выбирает сорт лимонада. 3. Монета попадает в реестр. 4. Для рассматриваемого основного сценария считаем, что нужный сорт лимонада имеется и реестр даёт команду отсеку доставить лимонад к лицевой панели автомата.

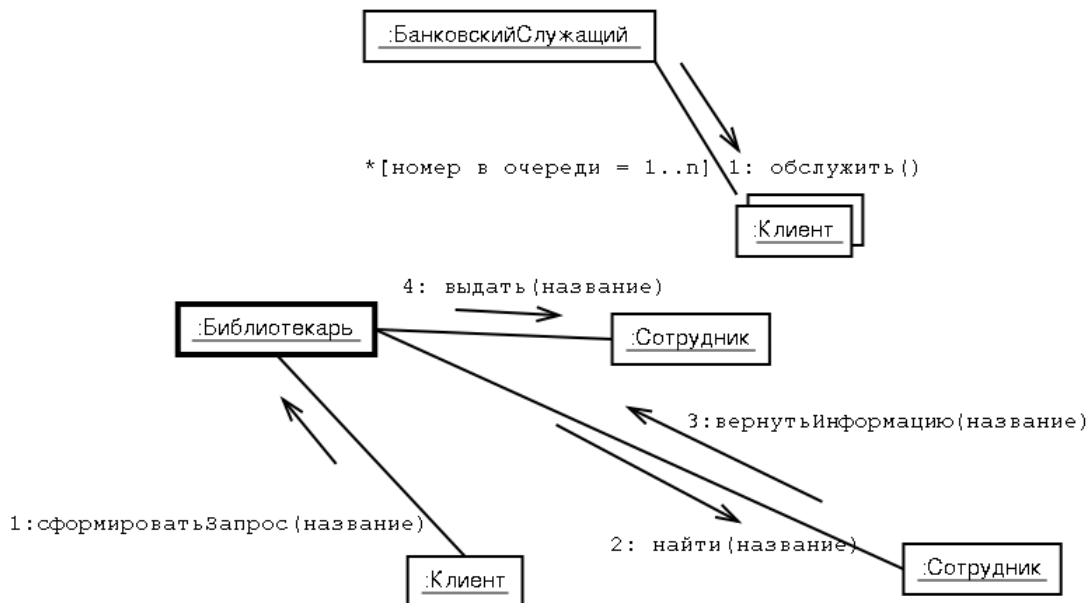
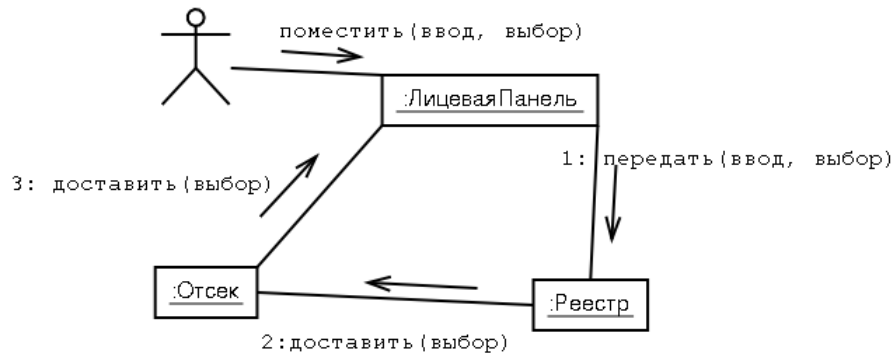


На диаграмме последовательностей возможны следующие навороты: [переход по условию], *[цикл пока], создание (создать(), '<<'создать'>>'), удаление объекта(X). Диаграмма последовательностей UML добавляет изменение времени ко взаимодействию объектов. Узкий прямоугольник - точка активации(выполнение одной из операций).

Диаграммы кооперации.

Диаграмма кооперации - это ещё один способ представления информации, которая раньше была изображена на диаграмме последовательности. Эти два типа диаграмм семантически эквивалентны. Диаграмма последовательностей упорядочена в соответствии со временем, диаграмма кооперации - в соответствии с пространственным расположением объектов.

На диаграмме кооперации ассоциации между объектами изображаются в виде сообщений, передаваемых одним объектом другому. Сообщение: стелка, порядк. номер, имя. Условия, как мы раньше отображаемся в кв. скобках. Чтобы описать цикл, надо поместить звёздочку перед условием.

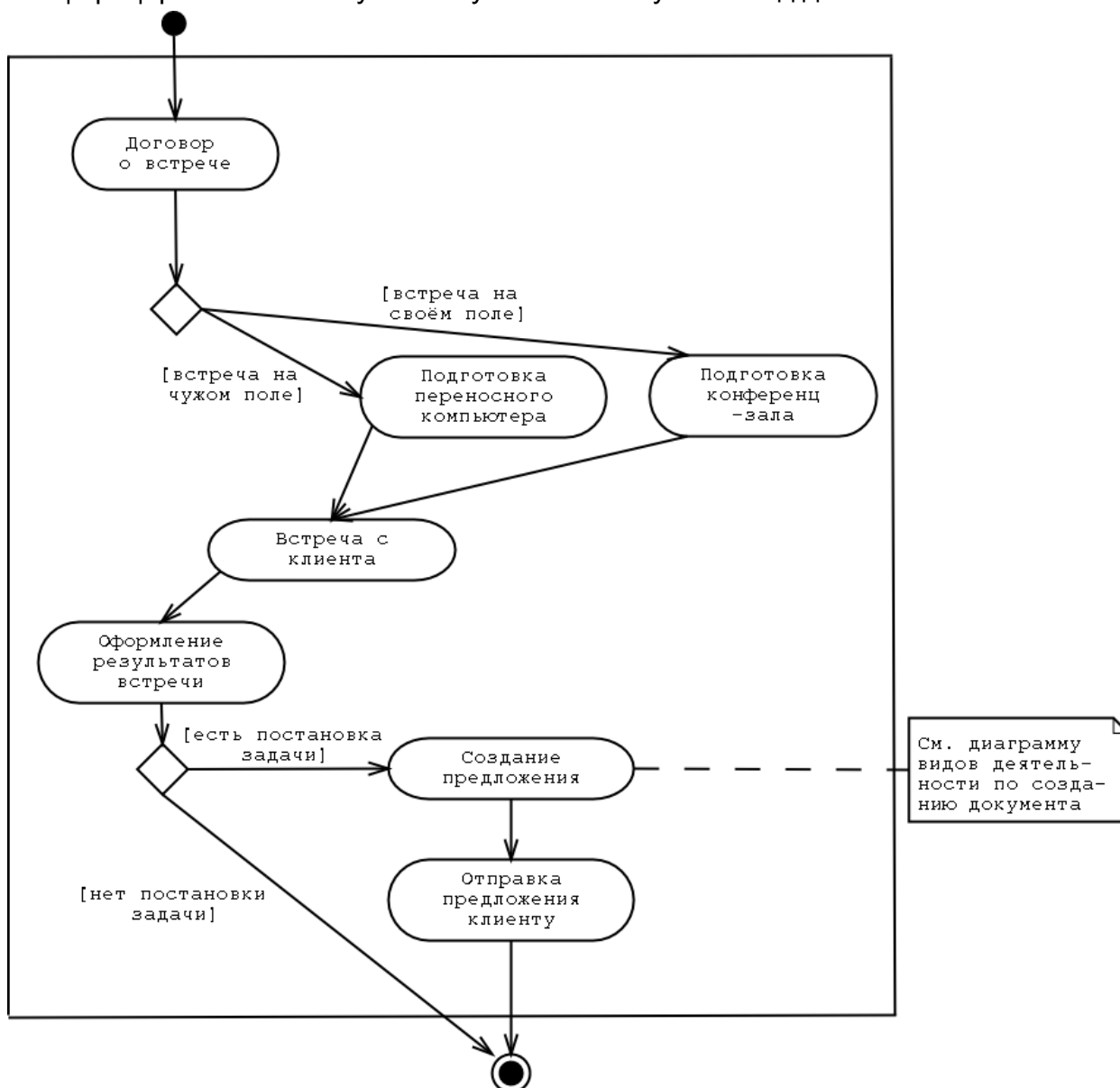


Некоторые операции являются дочерними по отношению к другим. В системе нумерации сообщений их номера пишутся после десятичной точки от имя предка. Диаграмма кооперации позволяет моделировать мн-во объектов получателей. Можно также изображать активные объекты, управляющие потоком сообщений, а также синхронизованные сообщения.

Диаграммы видов деятельности.

Эти диаграммы очень похожи на блок-схемы. Существуют точки принятия решения, барьеры и параллельные потоки, передача сигналов. Возможно

специфицировать к какому объекту относится нужный вид деятельности.



Диаграммы компонентов относятся к миру компьютеров.

Примеры: таблица, массив данных, исполняемый файл, динамически подключаемая библиотека, документ и т.д.

Типы компонент:

1. Компоненты развёртывания, которые формируют базис рабочих систем(DLL, исп.файлы).
2. Компоненты результатов деятельности, из которых создаются компоненты развёртывания(сорцы, файлы с данными)
3. Компоненты исполнения, создаваемые в результате работы системы.

Диаграммы развёртывания.

Диаграмма развёртывания UML описывает физич. систему в готовом виде.

Система состоит из узлов, каждый из которых изображается в виде куба. Линия между двумя кубами, символизирует соединение узлов.

Существует два типа узлов: процессор - это узел, который может выполнить команды компонента. Устройство - узел, которое это делать не может.

CASE-средства и редакторы диаграмм

- Rational Rose
- Real & Real IT
- Select Enterprise
- Visual UML - рекомендация Шмullера.
- Dia (Linux)

Литература.

1. ooad.asf.ru
2. Джозеф Шмullер "Освой самостоятельно UML за 24 часа".
3. Гради Буч "Объектно-ориентированный анализ и проектирование".

Источник статьи:

http://se.math.spbu.ru/seminars/se1/SE_16.htm