ЛЕКЦИЯ 4-5

Тема: унифицированный язык моделирования UML. Особенности UML. Основные виды диаграмм.

Итак, системы *моделирования процессов разработки строились на качестве выполнения требований к разрабатываемой системе.*

При анализе каждой модели выявлена эффективность только тех, в которых предусмотрено постоянное общение с заказчиком, наличие, качество и полнота документации.

Моделирование может применяться во всех перечисленных моделях разработки(класическая, инкрементная, RAD, XP, спиральная)

Моделирование может производиться и на этапе постановки задачи, как заказчиком, так и разработчиком. Здесь спектр задач моделирования весьма велик.

Например, заказчик предъявил особые требования к дизайну сайта и сделал макеты страниц в системе фотошоп.

Если есть готовая модель web-страниц, то нет необходимости перечислять требования к дизайну страниц в техническом задании, достаточно просто сослаться на модель.

На этапе анализа требований моделирование – хороший способ обезопасить себя как заказчика, ведь некоторые детали можно уточнить у заказчика.

Примером может служить случай, когда дизайн сайта описан в словесной форме и, чтобы уточнить, что именно имел в виду заказчик, можно воспользоваться системой Axure RP Pro 7.0. Также необходимо поступать и с графическим интерфейсом программ, ведь графическая часть плохо описывается словесно. *Если предметом разработки является компьютерная игра, то, возможно необходимы 3D модели элементов игры.*

Моделирование активно применяется на этапе проектирования программного продукта. Простейшим примером моделирования является составление блок-схемы, которая является модель алгоритма решения задачи.

Примером системы для моделирования на этапе проектирования может служить Astah – программа, предназначенная для создания UML диаграмм.

UML (Unified Modeling Language) — унифицированный язык моделирования, активно применяется при объектно-ориентированном анализе. **Цель UML** наглядное проектирование архитектуры программного продукта.

Для чего нужны UML-диаграммы

1. **Для создания «чертежей» программы**, схем, которые показывают, как будет устроено программное обеспечение изнутри, — то есть для проектирования. Это может быть описание связей между компонентами, модулей или сервисов, программных процессов и многого другого.
2. **Для визуализации уже имеющейся программной структуры**. Ряд инструментов позволяет создать UML на основе существующего кода, в таком случае диаграмма сгенерируется автоматически. Это называется реверс-инжинирингом.
3. **Для автоматической генерации кода или технической документации к нему**, так как UML поддерживает возможность создания продукта на основе диаграмм. автоматически сгенерированный код позже нуждается в доработке. А вот созданная таким образом документация обычно наглядная и понятная.
4. **Для внутренней и внешней коммуникации между сотрудниками, заказчиками** и другими: картинки и диаграммы UML понятнее людям, чем текстовые описания.

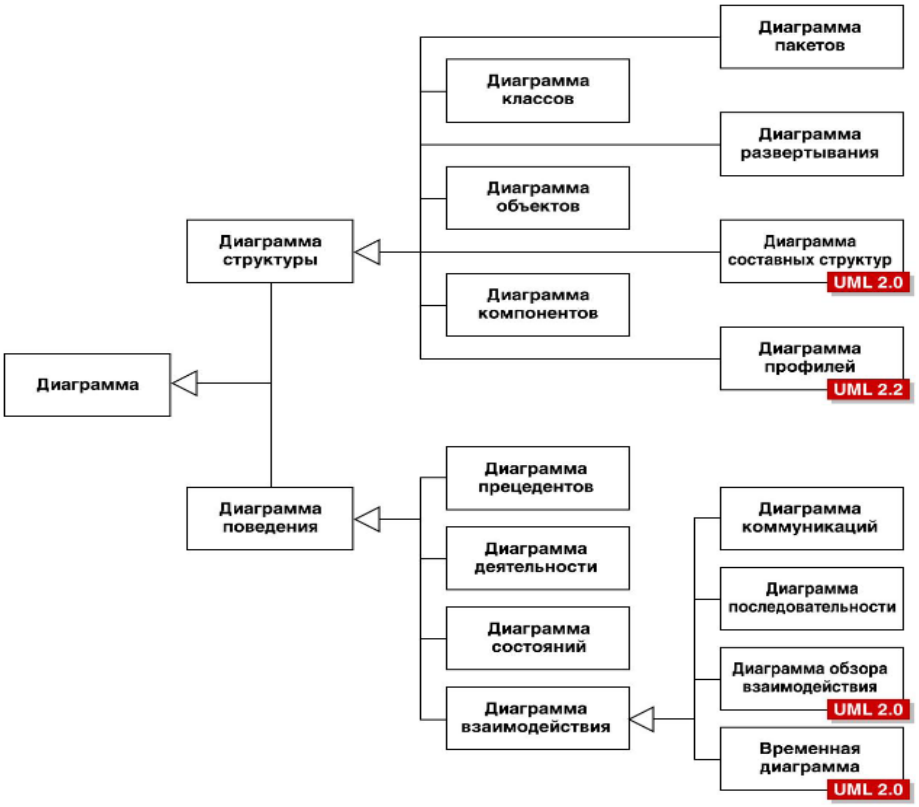
Важная особенность UML — этот язык поддерживает объектно-ориентированный подход, где все сущности представлены как объекты с определенными свойствами и методами. В диаграммах UML легко изобразить объекты, связи между ними, наследование и возможности передачи данных от одного объекта к другому.

Как устроена диаграмма UML

Схема UML — концептуальная: это значит, что она оперирует концепциями и связями между ними. Сама диаграмма состоит из фигур, значков, надписей, линий и контуров.

1. **Фигуры обычно обозначают ту или иную концепцию**: например, объект, класс, группу объектов или что-либо еще. Вариантов фигур в языке множество. Внутри одной фигуры могут находиться другие элементы, главное — чтобы они не пересекали границу.

2. **Значки тоже обозначают разные сущности, но отличаются от фигур**: внутрь них нельзя ничего поместить. Это могут быть более мелкие атомизированные структурные единицы, а могут быть служебные сущности, например для описаний.
3. **Надписи могут быть обычными, подчеркнутыми, курсивными**. Они именуют сущности, показывают, что есть что, и могут использоваться для описаний.
4. **Линии могут быть прямыми, ломаными, изогнутыми, направленными и ненаправленными, штриховыми и какими-либо** еще. Обычно они обозначают связи и зависимости сущностей друг от друга.
5. **Контуры — это контейнеры**, внутри которых помещаются концепции и связи между ними.



Все виды диаграмм в UML *Рис. 1. Виды диаграмм UML*

Среди них выделяют структурные диаграммы, отражающие ***статическую структуру системы:***

1. **диаграммы классов** описывают типы объектов системы (классы), их свойства и статические отношения, которые существуют между ними;
2. **диаграммы объектов** представляют снимок объектов системы в определенный момент времени;

3. **диаграммы компонентов** отображают компоненты системы, различные виды связей между ними и интерфейсы взаимодействия;

4. **диаграммы составных структур** отображают внутреннюю структуру компонентов или подсистем: части, взаимосвязи (коннекторы), интерфейсы и порты:

5. **диаграммы пакетов** позволяют структурировать модель или отобразить структуру системы, указывая разбиение на логические части, содержимое этих частей и взаимосвязи между ними;

6. **диаграммы развертывания** (размещения) представляют структуру аппаратных, коммуникационных средств, а также физическое расположение элементов программного обеспечения (элементов конфигурации) на оборудовании, информационные пути и протоколы взаимодействия между элементами;

7. **диаграммы профилей** описывают динамическую метамодель системы спомошью стереотипов, которые специфицируют изменение свойств объектов (классов, компонентов) в случае применения к системе различных профилей, модифицирующих ее поведение

и поведенческие диаграммы:

8. **диаграммы прецедентов** отображают функциональное назначение и границы системы или ее части в виде набора сценариев использования системы (прецедентов) акторами (заинтересованными лицами или внешними системами);

9. **диаграммы деятельности** описывают логику и процесс выполнения некоторой деятельности, включая основные шаги, переходы между ними и потоки управления;

10. **диаграммы состояний** (конечных автоматов) отображают все возможные состояния системы или отдельных объектов на протяжении их жизненного цикла, события и переходы между состояниями, а также структуру сложных и параллельных состояний;

Распространенные виды диаграмм:

1. **диаграмма вариантов использования** (use case diagram) состоит из эктеров и претендентов. Эктер отражает объект, а претендент – действие.

Диаграмма хорошо подходит для моделирования практически любых процессов.

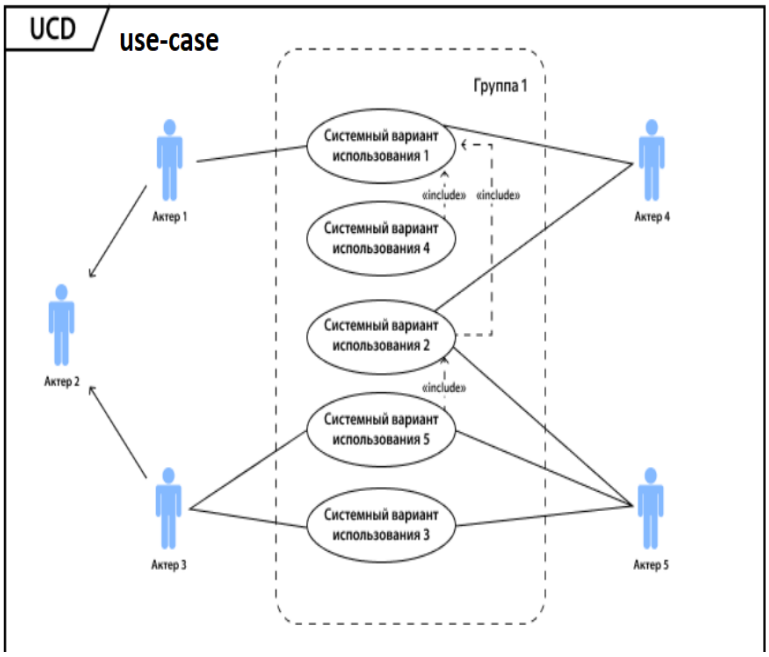
2. **Диаграмма последовательностей-** используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это средство документирования проекта с точки зрения сценариев использования!

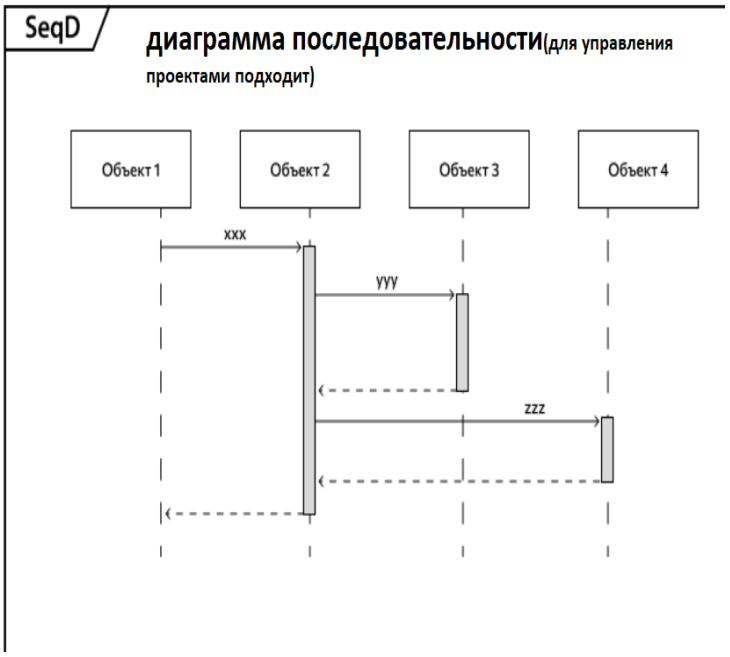
Диаграммы последовательностей обычно содержат *объекты*, которые *взаимодействуют в рамках сценария, сообщения*, которыми они обмениваются, и *возвращаемые результаты*, связанные с сообщениями.

3. **Диаграмма классов-** Диаграммы классов применяются только в объектно-ориентированном программировании. Они являются визуальным представлением структуры классов в программе.

Диаграммы классов позволяют разработчикам правильно спроектировать архитектуру программного продукта.

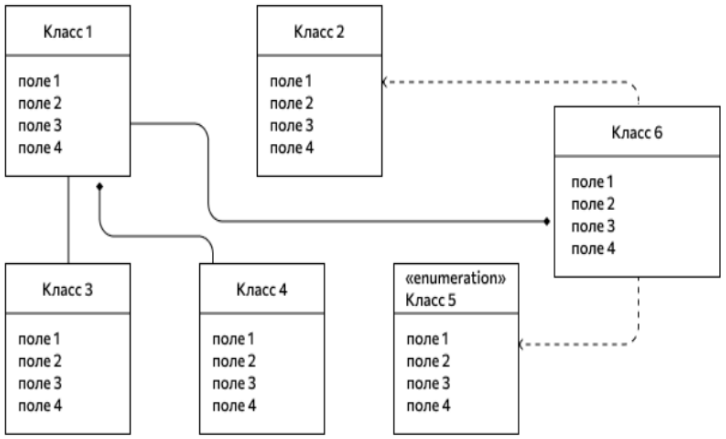
4.Диаграмма состояний-показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса





ClassD

диаграмма классов



StateD

диаграмма состояний