**Лекция 6. Основы UNL - проектирования систем**

1. **Процесс проектирования ИС**
2. **Формирование архитектуры информационной системы**

3. Основные диаграммы языка UML используемые для проектирования программных систем с распределенными базами данных

1. **Этапы проектирования информационных систем**

**Процесс проектирования ИС** – это процесс построения такого описания этой системы, которое делает возможным ее изучение и изготовление. Эти описания называются представлениями системы.

Существуют следующие *виды представленный*:

1. Представление с точки зрения пользователя. Оно отвечает на вопрос, что делает система?
2. Представление с точки зрения проектировщика. Отвечает на вопрос, из чего состоит и как устроена система?
3. Представление с точки зрения процессов. Отвечает на вопрос, как ведет себя система?
4. Представление с точки зрения реализации. Отвечает на вопрос, из каких конечных частей состоит система (исполнимые файлы, файлы инициализации)
5. Представление с точки зрения размещения. Где устанавливаются конкретные части системы

Спроектировать систему – значит создать 5 этих представленный.

Все виды представленный в процессе проектирования представляются в виде конструкций некоторого языка (в определенной нотации).

Существует несколько языков проектирования. Эти языки, как правило, графические, то есть элементами языка являются изображения. Большинство языков проектирования имеют инструментальную поддержку, автоматизирующую процесс проектирования.

Например,

1. BPWIN, ERWIN –программные среды. Позволяют создать проект системы.
2. SilverRun:
   1. BMP – моделирует процессы;
   2. ERX – строит схему «сущность – связи»;
   3. RDM – строит концептуальную схему базы данных;
   4. WRM – разработка форм приложений.

Для проектирования информационных систем широко используется язык UML – унифицированный язык визуального моделирования. Это язык для визуализации, специфицирования, конструирования и документирования программных систем.

Язык UML при разработке объектно-ориентированных систем использует собственную стандартизованную нотацию. Для автоматизации визуального проектирования систем на языке UML используется :

* программный продукт Rational Rose ;
* программный продукт Visual UML, поставляемый совместно с системой DELPHI .

Визуальным моделированием системы называется процесс графического представления системы с помощью некоторого стандартного набора графических символов. Основной целью визуального моделирования является организация общения между пользователями, разработчиками, аналитиками, тестировщиками, менеджерами. Создавая диаграмму, мы показываем взаимодействие между пользователем и системой, взаимодействие объектов внутри системы. Каждая заинтересованная сторона извлекает из модели ценную для себя информацию:

* Аналитики – состав объектов и взаимодействие между ними;
* Разработчики – какие объекты надо создавать и что вон должны делать;
* Тестировщики – анализ взаимодействия между объектами и что нужно тестировать;
* Менеджеры – как система устроена, как организовать службы и их взаимодействие;
* Руководители – как необходим организовать работу организации в целом.

На языке UML система представляется в виде набора диаграмм на крторых изображены сущности ,связанные отношениями Эти диаграммы имеют в разных авторов разные названия.

Сущности – абстракции представляющие объекты предметной области и системы.

## Сущности UML

Структурные

Поведенчес-кие

Группирую-щие

Аннотаци-онные

Классы

Інтерфейсы

Узлы

Кооперации

Прецеденты

Активные класы

Компоненты

Взаємодействия

Автоматы

Пакеты

Примечания

Подсистемы

Модели

Каркасы

Требова-ния

#### Отношение – отражение зависимостей между обектами,

В языке UML определено четырех типа отношений между элементами:

**Отношение зависимости -**  такое отношение между сущностями, при которой изменение состояния первой (независимой) влияет на состояние другой (зависимой)

Независимая

(Кассир)

Зависимая

(Сотрудник)

**Отношение ассоциации –** структурное отношение, которое описывает связь, то есть соединение между объектами (с разными типами связей). На основе ассоциаций выполняется навигация по элементам

Детальная

(Студент)

Главная

(Вуз)

Головній

(Группа)

Имя

роли

Имя

роли

(входить)

1

\*

Возможна ассоциация с двойным направлением навигации

Ім’я ролі

(входить)

Детальная

(Спонсор)

Главная

(Вуз)

Ім’я ролі

**Отношение обобщение (агрегирование)–** отношение показывает что один элемент (дочерний, часть) есть структурной частью другого (родительского, целого)

Целое

(Вуз)

Часть

(Студент)

Возможно композитное агрегирование. Сущность есть часть целого и имеет единое время жизни с ним

Часть

(Отдел)

Целое

(Вуз)

**Отношение реализации –** отношение показывает что один элемент определяет контракт(обязательства) а другой выполняет этот контракт

Исполнитель

(Класс)

# Определитель

( интерфейс)

**Диаграммы UML.**

Язык UML предусматривает использование совокупности диаграмм. Приведем общей описание этой совокупности. Диаграммы строятся в виде связанного графах в котором сущности являются вершины, а дуги – отношение.

###### ДІАГРАМИ UML

Стуктурные

# Поведінкові

Розгортання

Компонентів

Обєктів

Класів

Станів

Кооперації

Послідовностей

Прецедентів

Взаємодії

Діяльності

**2 Формирование архитектуры информационной системы** Архитектура информационной системы описывается с помощью 5 видов или представлений системы, любой из которых есть одна из возможных проекций организации и структуры системы и отвечает отдельному аспекту ее функционирования. Приведем эти представления:

* Вид с точки зрения ***прецедентов использование****.* Этот вид предназначен для описания системы как она конечным пользователем, аналитиком то лицом что проводит тестирование системы. В языке UML статические аспекты этого вида представленные диаграммами прецедентов, а динамические – диаграммами взаимодействий, состояний и действий.
* Вид с точки зрения ***проектирования****.* Этот вид предназначен для описания функциональных требований к системе, то есть функций, которые она должна предоставлять пользователю. В языке UML статические аспекты этого вида представленные диаграммами классов и объектов, а динамические – диаграммами взаимодействий, состояний и действий.
* Вид с точки зрения ***процессов****.* Этот вид предназначен для описания механизмов что выполняются параллельно и проблем их синхронизации В языке UML статические аспекты этого вида представленные диаграммами активных классов и объектов, а динамические – диаграммами взаимодействий, состояний и действий.
* Вид с точки зрения ***реализации****.* Этот вид предназначен для описания компонентов и файлов что используются для сборки и конечной конфигурации системы. В языке UML статические аспекты этого вида представленные диаграммами компонентов, а динамические – диаграммами взаимодействий, состояний и действий.
* Вид с точки зрения ***развертывание системы****.* Этот вид предназначен для описания топологии аппаратных средств на которые выполняется система. В языке UML статические аспекты этого вида представленные диаграммами развертывания, а динамические – диаграммами взаимодействий, состояний и действий.

3. Основные диаграммы языка UML используемые для проектирования программных систем с распределенными базами данных

**3.1 Диаграмма прецедентов**

На диаграмме прецедентов представлены пользователи (АКТЕРЫ) системы и варианты использования системы (ПРЕЦЕДЕНТЫ), а также связи между ними

Такое представление – это общий взгляд на систему независимый от пути ее реализации . Главная цель представления – отобразить что именно система может выполнять а не то как она это будет делать. Представление вариантов использования содержит:

* актеров (ACTORS)- действующие лица, которые находятся в состоянии взаимодействия с системой.
* прецеденты – ( USE CASE) варианты использования – функции верхнего уровня которые система предоставляет для пользователя.
* связи между актерами и вариантам использования
* документацию по вариантам использования, которая уточняет описание процессов, необходимых для выполнения функции с включением процедур обработки ошибок

Приведем пример диаграммы на рисунке.

Актер

(Клиент банка)

Связь

коментар

Прецедент

(Получить деньги)

Для элементов диаграммы прецедентов определяются соответствующие спецификации.

Спецификации актеров: имя актера, количество экземпляров, стереотип

Спецификации прецедентов: имя прецедента, описание , стереотип

Язык UML предусматривает использование нескольких типов связей между элементами диаграммы прецедентов, а именно

* связи коммуникации, то есть связь между актером и прецедентом направление стрелки определяет элемент что есть инициатором коммуникации;
* связь использования , связь которой показывает как один прецедент всегда использует другой;
* связь расширения связь которой показывает как один периодически прецедент использует другой;
* связь обобщения актера

3.2 Диаграммы взаимодействий и последовательностей. Диаграммы взаимодействия используются для моделирования динамических аспектов поведения системы отображают множество объектов, отношений между ними и сообщение которыми они обмениваются существует два вида таких диаграмм – диаграммы последовательностей и диаграммы кооперации.

Клієнт

На диаграммы последовательностей внимание акцентируется на последовательности благоустроенности сообщений во времени. Графически такая диаграмма представляет собой таблицу, в которую объекты располагаются вдоль осы ОХ, а сообщение в порядке возрастания времени вдоль осы ОУ.

На диаграммы кооперации внимание отводится структурной организации объектов, которые принимают или отправляют сообщение. Графически изображается в виде графа.

Пример диаграмма последовательностей сообщений

С : Клієнт

:Транзакція

р:ODBCProxy

“Create”

setaction(a,b,c)

Set value(a)

Set value(b)

commit

“destroy”

Пример диаграмма кооперации

С : Клієнт

:Транзакція

P:ODBCProxy

“local”

1. “create”
2. setaction
3. “destroy”

2.1Setvalue(a)

2.2 Setvalue(b)

“global”

**3.3 Диаграммы классов**

Рассматривая диаграмму взаимодействия, мы обнаружим на них объекты предметной области. Рассматривая объекты более обобщенно можно выделить множество объектов, обладающих одинаковой структурой и одинаковым поведением. Такое выделение называют *типизацией*, в результате которой формируются *классы* – это абстракция, представляющая такой набор объектов.

В языке UML для каждого проекта строится диаграмма классов. Их много и классы на них, а также связи между классами могут показываться с различной степенью детализации.

|  |  |
| --- | --- |
|  | ris2 |

При создании **класса** для него определяют **спецификации**:

1. Имя (name) класса
2. Область видимости (visible) :рublic , рrivate ,рackage – видимый внутри пакета.

Множественность (cardinality) – указание количества экземпляров класса.

1. Объем памяти, выделяемый под экземпляр класса (space)
2. Устойчивость класса: устойчивый , временный

Из статических классов формируется база данных программной системы. И в будущем множество статических классов порождает концептуальную схему базы данных.Параллелизм :sequential (последовательный) , synchronize (ситнхронный).

Стереотип класса:

# <Boundary> – граничный класс для общенияется с другими системами;

# <Entity> – класс сущности – статический класс, из таких классов создается база данных;

# <Control class> – управляющий класс, который порождает потоки управления другими классами.

# Язык UML разрешает пользователю определять собственные стереотипы, если это нужно.

С классом связано множество всех операций над элементами класса. При программировании порождают методы класса

**Спецификация операций:**

1. Возвращаемое значение;
2. Список параметров;
3. Видимость операции;
4. Сигнатура (синтаксический прототип);
5. Время выполнения;
6. Параллелизм;
7. Пост- и предусловие;
8. Семантика (текстовое описание логики операции).

**Связи между классами**

Существует 4 вида связей между классами:

1. Связь ассоциации – смысловая связь, которая позволяет обеспечить процесс навигации (связь типа главная–детальная). Эти связи предполагают наличие атрибутов связи. Обозначаются просто линией.
2. Связи зависимости – они говорят о том, что экземпляры зависимого класса зависят от экземпляров независимого класса. Линия связи идет от зависимого класса к независимому.
3. Связи агрегации – связь типа целое–часть. Например,
4. Связи обобщения (наследования) – связь между классами, один из которых родитель, а другой наследник.

Для каждой связи также указываются спецификации:

Имя связи, стереотип, ролевое имя, видимость, ограничения (описывают, какие именно экземпляры класса могут участвовать в связи).

**3.4 Диаграммы состояний**

На протяжении жизненного цикла объект находится в различных состояниях, которые сменяют друг друга. Например, объект «сотрудник» может находится в состояниях: прием на работу, прохождение испытательного срока, выполнение работы, нахождения на больничном, нахождения в отпуске, увольнения. В каждом из этих состояний объект обладает характерным для состояния поведением. Для моделирования жизненного цикла объекта используется диаграмма состояний.

Диаграмма состояний подсказывает программисту алгоритм обработки объектов.

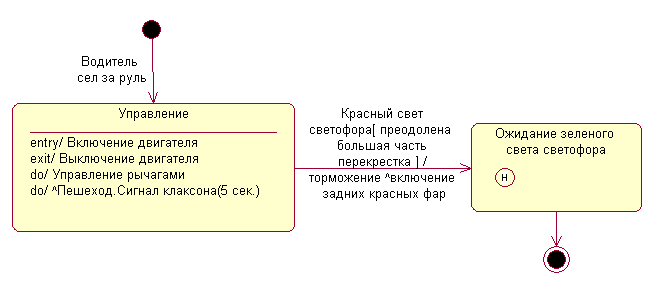
Пример диаграммы

Диаграмма начинается со специального значка – начальное состояние – это состояние объекта, в котором он находится после создания.

Каждое состояние имеет :

имя (Name).

Деятельность (do: Activity) – поведение объекта во время данного состояния, не выводящее его из этого состояния. Если деятельность прерывается, то значит, объект выходит из этого состояния.

Входное действие, событие (entry: Event) – действие, с которого начинается состояние.

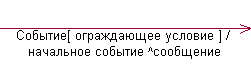
Выходное событие (exit: Event) – событие, которым заканчивается состояние, оно приводит к выходу из состояния.

Отправление сообщения (Do: ^Target.Message(Param)) другим объектам, как правило, вызывает операции этого другого объекта.

Диаграмма состояний завершается конечным состоянием – состоянием объекта перед его удалением.

exit

Помимо состояния на диаграмме состояний отображаются переходы объектов в другие состояния. Переходом называется реакция объекта на внешние события, переводящие его в другие состояния (обозначается стрелочкой).



Спецификация перехода

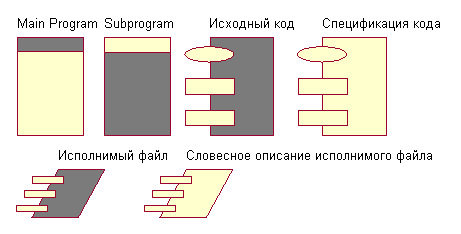
1. *Event* (само событие, вызывающее переход). Например, красный свет светофора.
2. *Ограждающие условия*, то есть условия, при котором состояние не изменится, несмотря на то, что событие перехода произошло. Например, красный свет светофора 🡪 управление заканчивается 🡪 начинается состояние ожидания другого сигнала. Ограждающее условие – преодолена большая часть перекрестка.
3. *Начальное событие* перехода. Например, торможение.
4. *Message* – сообщение, посылаемое во время перехода. Например, включение задних красных фар.

**3.5 Диаграмма компонентов.**

Для распределенных систем для отображения размещения частей системы на различных технических устройствах построить диаграмму компонент и диаграмму размещения (развертывания).

*Компонент* – физическая часть системы которая может быть заменена, она отвечает интерфейсу и выполняет его реализацию. Компоненты имеют место физической реализации, а не являются абстракциями и тех самым отличаются от классов

*Узел* – элемент фізічної системы которой есть вычислительным ресурсом – как правило компютер

 **На диаграмме отображаются компоненты и классы которые используют приведенные компоненты** Для отображения физического хранения системы строится диаграмма компонентов, на которой отображаются компоненты и связи между ними. В зависимости от инструментальной среды проектирования и программирования, набор компонент различен. Язык UML предполагает построение диаграммы размещения, используя следующие элементы диаграмм.

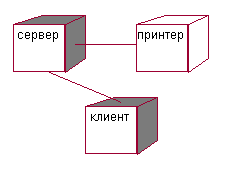
На диаграмме показаны связи между компонентами. Есть только один тип связи – связь зависимости. Говорят, что компонент А зависит от компонента В, если для работы компонента А необходим компонент В. Компонент А не транслируется без компонента В.

Для каждого элемента задаются спецификации. Важнейшие:

* Стереотип;
* Язык;
* Адрес, местонахождение.

Диаграмма компонентов нужна для четкого представления о размещении системы. Эта задача становится сверхважной и сверхактуальной, если команда разработчиков часто обновляется. Вторая причина – наличие вариантов реализации системы. Основная проблема – четкость хранения версий и подбор соответствующих названий.

**3.6 Диаграмма размещения проекта (**диаграммы развертывания).

**На диаграмме развертывания отображается расположение компонентов системы на узлах системы на узлах.** Строится, как правило, для распределенных систем и отображает размещение частей системы на различных технических устройствах. Как правило, эти устройства – либо компьютеры локальной сети, либо отдельные сетевые устройства (например, сетевые принтеры).

Продажи

Sales.exe

FrageAgent.dll

*Связи* между элементами диаграммы двух видов:

1. Физические каналы локальной сети
2. Internet-ссылки, если система глобальная.

Для каждого элемента диаграммы указывается *спецификация*:

* + Стереотип (ключевое слово);
  + Важнейшие технические требования узлов;
  + Адресная часть (местонахождение).

Кроме технических устройств на диаграммах размещения изображаются *процессы* – потоки управления (например, исполняемые или пакетные командные файлы).

*Спецификация процессов***:**

* Стереотип;

Приоритетность.

**4. Этапы проектирования информационных систем**

Построение модели проводится следующими этапами:

1. Сформулировать назначение системы одним предложением, которое кратко и точно указывает на цель создания системы.
2. Сделать декомпозицию назначения по функциональному критерию. Это есть ответить на вопрос, какие функции должна выполнять система для реализации своего назначения. Не забыть служебные функции (например, сервисное обслуживание, администрирование.)

* Определить которое именно виды системы необходимо представить в модели (как правило это выше указанные основные пять видов а может и дополнительные).
* Решить какие диаграммы необходимо создать и яки элементы на них отобразить.
* Построить необходимые диаграммы.
* Построить диаграмму прецедентов (диаграмму вариантов использования).
* Для каждого прецедента построить документ, который называется поток событий.
* Каждый из потоков отразить на диаграммах взаимодействия: и на диаграмме последовательности сообщений, и на диаграмме кооперации (объектов и сообщений).
* Рассматривая диаграмму взаимодействия, выделить множество объектов, обладающих одинаковой структурой и одинаковым поведением сформировать классы. Построить диаграмму классов.
* Для наиболее важных объектов построить диаграмму состояний.
* Соответственно полученной модели системы построить соответствующую концептуальную схему базы данных за соответствующей нотацией.
* Для отображения физического хранения системы построить диаграмму компонентов.
* Приступить к реализации системы, то есть к построению интерфейса пользователя, разработки и заполнение базы данных, разработку и реализацию алгоритмов обработки данных