**Лекція 6. Основи UML – проектування розподілених систем**

1. **Процес проектування ІС**
2. **Формування архітектури інформаційної системи**
3. **Основні діаграми мови UML, які використовуються для проектування програмних систем з розподіленими базами даних**
4. **Етапи проектування інформаційних систем**

**Процес проектування ІС** - це процес побудови такого опису цієї системи, який дає можливість її вивчення та створення. Ці описи називаються поданнями (представленнями) системи.

Існують наступні види подань системи:

1. Подання з погляду користувача. Воно відповідає на запитання, що робить система?
2. Подання з погляду проектувальника. Відповідає на запитання, з чого складається і як побудована система?
3. Подання з погляду процесів. Відповідає на запитання, як поводиться система?
4. Подання з погляду реалізації. Відповідає на запитання, з яких кінцевих частин складається система (файли запуску, файли ініціалізації)?
5. Подання з погляду розміщення. Де встановлюються конкретні частини системи?

Спроектувати систему - значить створити 5 цих подань.

Усі види подань у процесі проектування представляються у виді конструкцій деякої мови (у визначеній нотації).

Існує кілька мов проектування. Ці мови, як правило, графічні, тобто елементами мови є зображення. Більшість мов проектування мають інструментальну підтримку, що автоматизує процес проектування.

Наприклад:

1. BPWІ, ERWІ - програмні середовища. Дозволяють створити проект системи.
2. SіlverRun:
   1. BMP - моделює процеси;
   2. ERX - будує схему "сутність - зв'язок";
   3. RDM - будує концептуальну схему бази даних;
   4. WRM - розробляє форми додатків.

Для проектування інформаційних систем широко використовується мова UML - уніфікована мова візуального моделювання. Це мова для візуалізації, специфікації, конструювання і документування програмних систем.

Мова UML при розробці об’єктно-орієнтованих систем використовує власну стандартизовану нотацію. Для автоматизації візуального проектування систем мовою UML використовується :

* програмний продукт Ratіonal Rose;
* програмний продукт Vіsual UML, що поставляється разом із системою DELPHІ.

Візуальним моделюванням системи називається процес графічного представлення системи за допомогою деякого стандартного набору графічних символів. Основною метою візуального моделювання є організація спілкування між користувачами, розробниками, аналітиками, тестувальниками та менеджерами. Створюючи діаграму, ми показуємо взаємодію між користувачем і системою, взаємодію об'єктів усередині системи. Кожна зацікавлена сторона витягає з моделі необхідну для себе інформацію:

* + Аналітики - склад об'єктів і взаємодію між ними;
  + Розробникки - які об'єкти треба створювати і що вони повинні робити;
  + Тестувальники - аналіз взаємодії між об'єктами і що потрібно тестувати;
  + Менеджери - як система улаштована, як організувати служби та їх взаємодію;
  + Керівники - як необхідно організувати роботу організації в цілому.

Мовою UML система представляється у вигляді набору діаграм на яких зображені сутності, зв'язані відношенннями. Ці діаграми мають у різних авторів різні назви.

Сутності – абстракції, що представляють об'єкти предметної області і системи.

Сутності UML

Структурні

Поведінки

Групування

Аннотаційні

Класи

Інтерфейси

Вузли

Кооперації

Прецеденти

Активні класы

Компоненти

Взаємодії

Автомати

Пакети

Примітки

Підсистеми

Моделі

Каркаси

Вимоги

**Відношення - відображення залежностей між об’єктами**

У мові UML визначено чотири типи відношень між елементами:

**Відношення залежності** - таке відношення між сутностями, при якому зміна стану першої (незалежної) впливає на стан іншої (залежної) сутності.

Незалежна

(Касир)

Залежна

(Співробітник)

**Відношення асоціації** - структурне відношення, що описує зв'язок, тобто з'єднання між об'єктами (з різними типами зв'язків). На основі асоціацій виконується навігація по елементах відношення.

Головна

(ВУЗ)

Детальна

(Студент)

Ім’я

ролі

Ім’я

ролі (входити)

Можлива асоціація з подвійним напрямком навігації

Головна

(ВУЗ)

Детальна

(Спонсор)

Ім’я

ролі

Ім’я

ролі (входити)

**Відношення узагальнення (агрегування)** - відношення показує що один елемент (дочірній, частина) є структурною частиною іншого (батьківського, цілого)

Ціле

(ВУЗ)

Частина

(Студент)

Можливо композитне агрегування. Сутність є частина цілого і має з ним єдиний час життя.

Ціле

(ВУЗ)

Частина

(Відділ)

**Відношення реалізації** - відношення показує що один елемент визначає контракт (зобов'язання), а інший виконує цей контракт.

Визначник

(Інтерфейс)

Виконавець

(Клас)

**Діаграми UML**

Мова UML передбачає використання сукупності діаграм. Приведемо загальний опис цієї сукупності. Діаграми будуються у вигляді зв'язаного графа у якому сутності – вершини графа, а відношення – дуги графа.

###### ДІАГРАМИ UML

Стуктурні

Поведінкові

Розгортання

Компонентів

Об’єктів

Класів

Станів

Кооперації

Послідовностей

Прецендентів

|  |
| --- |
| Взаємодії |

Діяльності

**.2 Формування архітектури інформаційної системи**

Архітектура інформаційної системи описується за допомогою 5 видів чи представлень системи, кожний з яких є одна з можливих проекцій організації і структури системи і відповідає окремому аспекту її функціонування. Приведемо ці представлення:

- Вид з погляду ***прецедентів використання***. Цей вид призначений для опису системи, як вона сприймається кінцевим користувачем, аналітиком та тестувальником системи. У мові UML статичні аспекти цього виду представлені діаграмами прецедентів, а динамічні - діаграмами взаємодій, станів і дій.

- Вид з погляду ***проектування***. Цей вид призначений для опису функціональних вимог до системи, тобто функцій, які вона повинна надавати користувачу. У мові UML статичні аспекти цього виду представлені діаграмами класів і об'єктів, а динамічні - діаграмами взаємодій, станів і дій.

- Вид з погляду ***процесів***. Цей вид призначений для опису механізмів, що виконуються паралельно і проблем їхньої синхронізації. В мові UML статичні аспекти цього виду представлені діаграмами активних класів і об'єктів, а динамічні - діаграмами взаємодій, станів і дій.

- Вид з погляду ***реалізації***. Цей вид призначений для опису компонентів і файлів, що використовуються для зборки і кінцевої конфігурації системи. У мові UML статичні аспекти цього виду представлені діаграмами компонентів, а динамічні - діаграмами взаємодій, станів і дій.

- Вид з погляду ***розгортання*** ***системи***. Цей вид призначений для опису топології апаратних засобів на яких виконується система. У мові UML статичні аспекти цього виду представлені діаграмами розгортання, а динамічні - діаграмами взаємодій, станів і дій.

**5.3. Основні діаграми мови UML, які використовуються для проектування програмних систем з розподіленими базами даних**

**5.3.1 Діаграма прецедентів**

На діаграмі прецедентів представлені користувачі (АКТОРИ) системи і варіанти використання системи (ПРЕЦЕНДЕНТИ), а також зв'язки між ними.

Таке представлення - це загальний погляд на систему незалежний від способу її реалізації. Головна мета представлення - відобразити що саме система може виконувати, а не те як вона це буде робити. Представлення містить:

- акторів (ACTORS) - діючі особи, що працюють із системою;

- прецеденти - (USE CASE) варіанти використання - функції верхнього рівня, які система надає користувачу;

- зв'язки між акторами і варіантами використання;

- документацію про варіанти використання, що уточнює опис процесів, необхідних для виконання функцій системи, включно з процедурами обробки помилок.

Приведемо приклад діаграми на рисунку.

Актор

(Клієнт банка)

Зв’язок

коментар

Прецедент

(Отримати гроші)

Для елементів діаграми прецедентів задаються відповідні специфікації.

Специфікації акторів: ім'я актора, кількість екземплярів, стереотип.

Специфікації прецедентів: ім'я прецеденту, опис, стереотип.

Мова UML передбачає використання декількох типів зв'язків між елементами діаграми прецедентів, а саме:

* зв'язок комунікації, тобто зв'язок між актором і прецедентом, напрямок стрілки визначає елемент, що є ініціатором комунікації;
* зв'язок використання, зв'язок що показує як один прецедент завжди використовує інший;
* зв'язок розширення, зв'язок що показує як один прецедент періодично використовує інший;
* зв'язок узагальнення актора.

**5.3.2 Діаграми взаємодій і послідовностей**

Діаграми взаємодії використовуються для моделювання динамічних аспектів поведінки системи та відображають мнножину об'єктів, відношень між ними і повідомлення, якими вони обмінюються. Існує два види таких діаграм - діаграми послідовностей і діаграми кооперації.

На ***діаграмі послідовностей*** увага акцентується на послідовності впорядкування повідомлень у часі. Графічно така діаграма являє собою таблицю, у якій об'єкти розташовуються уздовж осі ОХ, а повідомлення в порядку зростання часу уздовж осі ОУ.

На ***діаграмі кооперації*** увага приділяється структурній організації об'єктів, що приймають або відправляють повідомлення. Графічно така діаграма зображується у виді графа.

Приклад:

**діаграма послідовностей повідомлень**

С : Клієнт

:Транзакція

р:ODBCProxy

“Create”

setaction(a,b,c)

Set value(a)

Set value(b)

commit

“destroy”

Приклад:

**діаграма кооперації**

С : Клієнт

:Транзакція

P:ODBCProxy

“local”

1. “create”
2. setaction
3. “destroy”

2.1Setvalue(a)

2.2 Setvalue(b)

“global”

**5.3.3 Діаграми класів**

Розглядаючи діаграму взаємодії, ми знайдемо на ній об'єкти предметної області. Розглядаючи об'єкти більш узагальнено можна виділити множину об'єктів, які мають однакову структуру і однакову поведінку. Таке виділення називають типізацією, у результаті якої формуються класи - це абстракція, що представляє певний набір об'єктів.

У мові UML для кожного проекту будується діаграма класів. Класи, а також зв'язки між класами можуть показуватися з різним ступенем деталізації.

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

При створенні класу для нього визначають ***специфікації***:

1. ім'я (name) класу;
2. область видимості (vіsіble): рublіc, рrіvate, рackage - видимий усередині пакета.

Множинність (cardіnalіty) - указівка кількості екземплярів класу.

1. обсяг пам'яті, що виділяється під екземпляр класу (space);
2. стійкість класу: стійкий, тимчасовий.

Зі статичних класів формується база даних програмної системи. І в майбутньому безліч статичних класів породжує концептуальну схему бази даних. Паралелізм :sequentіal (послідовний), synchronіze (ситнхронный).

Стереотип класу:

1. <Boundary> - граничний клас для спілкування з іншими системами;
2. <Entіty> - клас сутності - статичний клас, з таких класів створюється база даних;
3. <Control class> - керуючий клас, що породжує потоки керування іншими класами.

Мова UML дозволяє користувачу визначати власні стереотипи, якщо це потрібно.

З класом зв'язана множина операцій над елементами класу. При програмуванні породжують методи класу.

**Специфікація операцій:**

1. значення, що повертається;
2. список параметрів;
3. видимість операції;
4. сигнатура (синтаксичний прототип);
5. час виконання;
6. паралелізм;
7. посада- та передумова;
8. семантика (текстовий опис логіки операції).

**Зв'язки між класами**

Існує 4 види зв'язків між класами:

1. зв'язок асоціації - смисловий зв'язок, що дозволяє забезпечити процес навігації (зв'язок типу головна-детальна). Ці зв'язки припускають наявність атрибутів зв'язку. Позначаються просто лінією;
2. зв'язки залежності - вони говорять про те, що екземпляри залежного класу залежать від екземплярів незалежного класу. Лінія зв'язку йде від залежного класу до незалежного.
3. зв'язок агрегації - зв'язок типу ціле-частина.
4. зв'язок узагальнення (успадкування) - зв'язок між класами, один із яких батько, а інший спадкоємець.

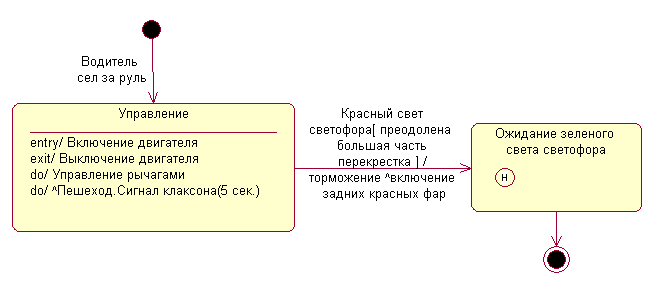
Для кожного зв'язку також указуються специфікації:

Ім'я зв'язку, стереотип, рольове ім'я, видимість, обмеження (описують, які саме екземпляри класу можуть брати участь у зв'язку).

**5.3.4 Діаграми станів**

Протягом життєвого циклу об'єкт знаходиться в різних станах, що змінюють один одного. Наприклад, об'єкт "співробітник" може знаходитись у стані: прийом на роботу, проходження іспитового терміну, виконання роботи, перебування на лікарняному, перебування у відпустці, звільнення. У кожному з цих станів об'єкт має характерну для стану поведінку. Для моделювання життєвого циклу об'єкта використовується ***діаграма станів***.

Діаграма стану підказує програмісту алгоритм обробки об'єктів.

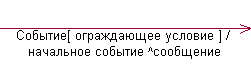
Приклад діаграми

Діаграма починається зі спеціального значка - початковий стан - це стан об'єкта, у якому він знаходиться після створення.

Кожен стан має :

* Ім'я (Name).
* Діяльність (do: Actіvіty) - поведінка об'єкта під час даного стану, яка не виводить його з цього стану. Якщо діяльність переривається, то об'єкт виходить з цього стану.
* Вхідна дія, подія (entry: Event) - дія, з якої починається стан.
* Вихідна подія (exіt: Event) - подія, якою закінчується стан, він приводить до виходу зі стану.
* Відправлення повідомлення (Do: ^Target.Message(Param)) іншим об'єктам, як правило, викликає операції цього об'єкта.

exitДіаграма станів завершується кінцевим станом - станом об'єкта перед його видаленням.

Крім того на діаграмі станів відображаються переходи об'єктів в інші стани. Переходом називається реакція об'єкта на зовнішні події, що переводять його у інші стани (позначається стрілкою).

Специфікація переходу

1. *Event* (сама подія, що викликає перехід). Наприклад, червоне світло світлофора.
2. *Обмежуючі умови*, тобто умови, при яких стан не зміниться, незважаючи на те, що подія переходу відбулася. Наприклад, червоне світло світлофора 🡪 керування закінчується 🡪 починається стан чекання іншого сигналу. Обмежуюча умова - пройдена більша частина перехрестя.
3. *Початкова подія* переходу. Наприклад, гальмування.
4. *Message* - повідомлення, що посилається під час переходу. Наприклад, включення задніх червоних фар.

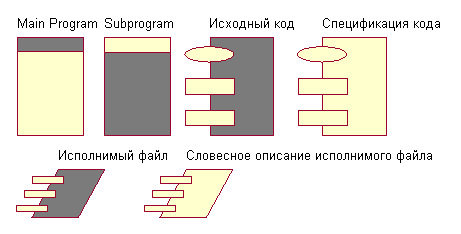
**5.3.5 Діаграма компонентів**

Для розподілених систем для відображення розміщення частин системи на різних технічних пристроях будують діаграми компонентів і діаграми розміщення (розгортання).

***Компонент*** - фізична частина системи яка може бути замінена, вона відповідає інтерфейсу і виконує його реалізацію. Компоненти мають місце фізичної реалізації, але не є абстракціями і тих самим відрізняються від класів.

***Вузол*** - елемент фізічної системи яка є обчислювальним ресурсом - як правило компютер.

**На діаграмі відображаються компоненти і класи які використовують наведені компоненти.** Для відображення фізичного розміщення системи будується діаграма компонентів, на якій відображаються компоненти і зв'язки між ними. У залежності від інструментального середовища проектування і програмування, набір компонентів різний. Мова UML припускає побудову діаграми розміщення, використовуючи наступні елементи діаграм.



На діаграмі показані зв'язки між компонентами. Є тільки один тип зв'язку - ***зв'язок залежності***. Говорять, що компонент А залежить від компонента В, якщо для роботи компонента А необхідний компонент В. Компонент А не транслюється без компонента В.

Для кожного елемента задаються специфікації. Найважливіші:

· стереотип;

· мова;

· адреса, місцезнаходження.

Діаграма компонентів потрібна для чіткого представлення про розміщення системи. Ця задача стає надважливою та надактуальною, якщо команда розробників часто зміюється. Друга причина - наявність варіантів реалізації системи. Основна проблема - чіткість збереження версій та підбір відповідних назв.

**5.3.6 Діаграма розміщення проекту (діаграми розгортання)**

**На діаграмі розгортання відображається розташування компонентів системи на вузлах системи**. Ця діагррама будується, як правило, для розподілених систем і відображає розміщення частин системи на різних технічних пристроях. Як правило, ці пристрої - або комп'ютери локальної мережі, або окремі мережні пристрої (наприклад, мережні принтери).

Продажі

Sales.exe

FrageAgent.dll

север

клієнт

принтер

Зв'язки між елементами діаграми бувають двох видів:

1. Фізичні канали локальної мережі.
2. Іnternet-посилання, якщо система глобальна.

Для кожного елемента діаграми вказується специфікація:

· стеріотип (ключове слово);

· найважливіші технічні вимоги вузлів;

· адресна частина (місцезнаходження).

Крім технічних пристроїв на діаграмах розміщення зображуються процеси - потоки керування (наприклад, виконувані або пакетні командні файли).

Специфікація процесів:

* стереотип;
* пріоритетність.

**5.4. Етапи проектування інформаційних систем**

Побудова моделі проводиться за наступними етапами:

1. Сформулювати призначення системи однією пропозицією, що коротко і точно вказує на мету створення системи.
2. Зробити декомпозицію призначення за функціональним критерієм. Це є відповіддю на питання, які функції повинна виконувати система для реалізації свого призначення. Не забути службові функції (наприклад, сервісне обслуговування, адміністрування.)

* Визначити які саме види системи необхідно представити в моделі (як правило це вище зазначені основні п'ять видів, а може і додаткові).
* Вирішити які діаграми необхідно створити і які елементи на них відобразити.
* Побудувати необхідні діаграми.
  + Побудувати діаграму прецедентів (діаграму варіантів використання).
  + Для кожного прецеденту побудувати документ, що називається потік подій.
  + Кожний з потоків відбити на діаграмах взаємодії: і на діаграмі послідовності повідомлень, і на діаграмі кооперації (об'єктів і повідомлень).
  + Розглядаючи діаграму взаємодії, виділити множину об'єктів, що володіють однаковою структурою і однаковою повдінкою. Сформувати класи. Побудувати діаграму класів.
  + Для найбільш важливих об'єктів побудувати діаграму станів.

1. Відповідно до отриманої моделі системи побудувати відповідну концептуальну схему бази даних за відповідною нотацією.
2. Для відображення фізичного збереження системи побудувати діаграму компонентів.
3. Приступити до реалізації системи, тобто до побудови інтерфейсу користувача, розробки та заповнення бази даних, розробки та реалізації алгоритмів обробки даних.