UML

UML — унифицированный язык моделирования.

Мова UML є общецелевого мову візуального моделювання, яка розроблений для специфікації, візуалізації, проектування та документування компонентів програмного забезпечення, бізнес-процесів і інших систем.

UML є об'єктно-орієнтована мова моделювання, що володіє наступними основними характеристиками:

• є мовою візуального моделювання, який забезпечує розробку репрезентативних моделей для організації взаємодії замовника і розробника ІС, різних груп розробників ІС;

• містить механізми розширення і спеціалізації базових концепцій мови.

UML - це стандартна нотація візуального моделювання програмних систем, прийнята консорціумом Object Managing Group (OMG) восени 1997 року, і на сьогоднішній день вона підтримується багатьма об'єктно-орієнтованими CASE-продуктами.

UML включає внутрішній набір засобів моделювання (модулів?) («Ядро»), які зараз прийняті в багатьох методах і засобах моделювання. Ці концепції необхідні в більшості прикладних задач, хоча не кожна концепція необхідна в кожній частині кожної програми. Користувачам мови надані можливості:

• будувати моделі на основі засобів ядра, без використання механізмів розширення для більшості типових програм;

• додавати при необхідності нові елементи і умовні позначення, якщо вони не входять в ядро, або спеціалізувати компоненти, систему умовних позначень (нотацію) і обмеження для конкретних предметних областей.

На зображенні чітко просматріваютя проблеми, які виникають при отсутвие чіткої специфікації створюваного програмного продукту. Використання UML дозволяє уникнути більшості з цих проблем.

Синтаксис і семантика основних об'єктів UML

класи

Класи - це базові елементи будь-якої об'єктно-орієнтованої системи. Класи є опис сукупностей однорідних об'єктів з притаманними їм властивостями - атрибутами, операціями, відносинами і семантикою.

В рамках моделі кожному класу привласнюється унікальне ім'я, що відрізняє його від інших класів. Якщо використовується складене ім'я (на початку імені додається ім'я пакета, куди входить клас), то ім'я класу має бути унікальним в пакеті.

Атрибут - це властивість класу, яке може приймати безліч значень. Безліч допустимих значень атрибута утворює домен. Атрибут має ім'я і відображає деяке властивість моделюється сутності, загальне для всіх об'єктів даного класу. Клас може мати довільну кількість атрибутів.

Операція - реалізація функції, яку можна запросити у будь-якого об'єкта класу. Операція показує, що можна зробити з об'єктом. Виконання операції часто пов'язано з обробкою і зміною значень атрибутів об'єкта, а також зміною стану об'єкта.

На рис. 11.1 наведено графічне зображення класу «Замовлення» в нотації **UML.**
Рис. 11.1.

Зображення класу в UML

Синтаксис UML для властивостей класів (в окремих програмних засобах, наприклад, в IBM UML Modeler, порядок запису параметрів може бути іншим):

<

ознака видимості> <ім'я атрибута>: <тип даних>

= <Значення за замовчуванням>

<Ознака видимості> <ім'я операції> <(список аргументів)>

Видимість властивості вказує на можливість його використання іншими класами. Один клас може «бачити» інший, якщо той перебуває в області дії першого і між ними існує явне або неявне відношення. В UML визначено три рівня видимості:

• public (загальний) - будь-який зовнішній клас, який «бачить» цей, може користуватися його загальними властивостями. Позначаються знаком «+» перед ім'ям атрибута або операції;

• protected (захищений) - тільки будь-який нащадок даного класу може користуватися його защіщннимі властивостями. Позначаються знаком «#»;

• private (закритий) - тільки даний клас може користуватися цими властивостями. Позначаються символом «-».

Ще однією важливою характеристикою атрибутів і операцій класів є область дії. Область дії властивості вказує, чи буде воно проявляти себе по-різному в кожному примірнику класу, або одне і те ж значення властивості буде спільно використовуватися всіма екземплярами:

• instance (екземпляр) - у кожного екземпляра класу є власне значення даного властивості;

• classifier (класифікатор) - всі екземпляри спільно використовують загальне значення даної властивості (виділяється на діаграмах підкресленням).

Можлива кількість примірників класу називається його кратністю. В UML можна визначати такі різновиди класів:

• що не містять жодного примірника - тоді клас стає службовим (Abstract);

• містять рівно один екземпляр (Singleton);

• містять задане число примірників;

• містять довільне число примірників.

Принципове призначення класів характеризують стереотипи. Це, фактично, класифікація об'єктів на високому рівні, що дозволяє визначити деякі основні властивості об'єкта (приклад стереотипу - клас «дійова особа»). Механізм стереотипів є також засобом розширення словника UML за рахунок створення на основі існуючих блоків мови нових, специфічних для вирішення конкретної проблеми.

Діаграма класів

Класи в UML зображуються на діаграмах класів, які дозволяють описати систему в статичному стані - визначити типи об'єктів системи і різного роду статичні зв'язки між ними.

Класи відображають типи об'єктів системи.

Між класами можливі різні відносини, представлені на рис. 11.2:

• залежності, які описують існуючі між класами відносини використання;

• узагальнення, що зв'язують узагальнені класи зі спеціалізованими;

• асоціації, що відображають структурні відносини між об'єктами класів.

Мал. 11.2. Відображення зв'язків між класами

Залежністю називається відношення використання, згідно з яким зміна в специфікації одного елемента (наприклад, класу «товар») може вплинути на використовує його елемент (клас «рядок замовлення»). Часто залежності показують, що один клас використовує інший як аргумент.

Узагальнення - це відношення між загальною сутністю (батьком - клас «клієнт») і її конкретним втіленням (нащадком - класи «корпоративний клієнт» або «приватний клієнт»). Об'єкти класу-нащадка можуть використовуватися всюди, де зустрічаються об'єкти класу-батька, але не навпаки. При цьому він успадковує властивості батька (його атрибути і операції). Операція нащадка з тієї ж сигнатурою, що і у (?) Із батьків, заміщає (?) Операцію з батьків; це властивість називають поліморфізмом. Клас, у якого немає батьків, але є нащадки, називається кореневим. Клас, у якого немає нащадків, називається листовим.

Асоціація - це відношення, яке показує, що об'єкти одного типу якимось чином пов'язані з об'єктами іншого типу («клієнт» може зробити «замовлення»). Якщо між двома класами встановлено зв'язок, то можна переміщатися від об'єктів одного класу до об'єктів іншого. При необхідності направлення навігації може здаватися стрілкою. Допускається завдання асоціацій на одному класі. В цьому випадку обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу можна зв'язати інші об'єкти з того ж класу. Асоціації може бути присвоєно ім'я, яке описує семантику відносин. Кожна асоціація має дві ролі, які можуть бути відображені на діаграмі (рис. 11.3). Роль асоціації має властивість множинності, яке показує, скільки відповідних об'єктів може брати участь в цих питань.

Рис. 11.3. властивості асоціації

Мал. 11.3 ілюструє модель формування замовлення. Кожне замовлення може бути створений єдиним клієнтом (множинність ролі 1.1). Кожен клієнт може створити один і більше замовлень (множинність ролі 1..n). Напрямок навігації показує, що кожне замовлення повинен бути «прив'язаний» до певного клієнта.

Такого роду асоціація є простою і відображає відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і жоден з них не є більш важливим, ніж інший. Якщо доводиться моделювати відношення типу «частина-ціле», то використовується спеціальний тип асоціації - агрегування. У такій асоціації один з класів має більш високий ранг (ціле - клас «замовлення», рис. 11.2) і складається з декількох менших за рангом класів (частин - клас «рядок замовлення»). В UML використовується і сильніша різновид агрегації - композиція, в якій об'єкт-частина може належати тільки єдиному цілому. У композиції життєвий цикл частин і цілого збігаються, будь-яке видалення цілого обов'язково захоплює і його частини.

Для асоціацій можна задавати атрибути і операції, створюючи за звичайними правилами UML класи асоціацій.

діаграми використання

Діаграми використання описують функціональність ІС, яка буде видна користувачам системи. «Кожна функціональність» зображується у вигляді

«прецедентів використання» (use case) або просто прецедентів. Прецедент - це типове взаємодію користувача з системою, яка при цьому:

• описує видиму користувачем функцію,

• може представляти різні рівні деталізації,

• забезпечує досягнення конкретної мети, важливої для користувача.

Прецедент позначається на діаграмі овалом, пов'язаним з користувачами, яких прийнято називати діючими особами (акторами, actors). Дійові особи використовують систему (або використовуються системою) в даному прецеденті. Дійова особа виконує деяку роль в даному прецеденті. На діаграмі зображується тільки одна дійова особа, однак реальних користувачів, які виступають у цій ролі по відношенню до ІС, може бути багато. Список всіх прецедентів фактично визначає функціональні вимоги до ІС, які лежать в основі розробки технічного завдання на створення системи.

На діаграмах прецедентів, крім зв'язків між діючими особами і прецедентами, можливе використання ще двох видів зв'язків між прецедентами: «використання» і «розширення» (рис. 11.4). Зв'язок типу «розширення» застосовується, коли один прецедент подібний до іншого, але має трохи більше функціональне навантаження. Її слід застосовувати при описі змін в нормальному поведінці системи. Зв'язок типу «використання» дозволяє виділити якийсь фрагмент поведінки системи і включати його в різні прецеденти без повторного опису.

На рис. 11.4 показано, що при виконанні прецеденту «формування замовлення» можливе використання інформації з попереднього замовлення, що дозволить не вводити всі необхідні дані. А при виконанні прецедентів «оцінити ризик угоди» і «узгодити ціну» необхідно виконати одну і ту саму дію - розрахувати вартість замовлення.

Мал. 11.4. Зв'язки на діаграмах прецедентів

Динамічні аспекти поводження системи відображаються наведеними нижче діаграмами.

На відміну від деяких підходів об'єктного моделювання, коли і стан, і поведінка системи відображаються на діаграмах класів, UML відокремлює опис поведінки в діаграми взаємодії. В UML діаграми класів не містять повідомлень, які ускладнюють їх читання. Потік повідомлень між об'єктами виноситься на діаграми взаємодії. Як правило, діаграма взаємодії охоплює поведінку об'єктів в рамках одного варіанту використання.

Прямокутники на діаграмі представляють різні об'єкти і ролі, які вони мають в системі, а лінії між класами відображають відносини (або асоціації) між ними. Повідомлення позначаються ярликами біля стрілок, вони можуть мати нумерацію і показувати повернені значення.

Існують два види діаграм взаємодії: діаграми послідовностей і кооперативні діаграми.

діаграми послідовностей

Цей вид діаграм використовується для точного визначення логіки сценарію виконання прецеденту. Діаграми послідовностей відображають типи об'єктів, що взаємодіють при виконанні прецедентів, повідомлення, які вони посилають один одному, і будь-які повертаються значення, асоційовані з цими повідомленнями. Прямокутники на вертикальних лініях показують «час життя» об'єкта. Лінії зі стрілками і написами назв методів означають виклик методу в об'єкта.

Мал. 11.5. Діаграма послідовності обробки замовлення

• вводяться рядки замовлення;

• по кожному рядку перевіряється наявність товару;

• якщо запас достатній - ініціюється поставка;

• якщо запас недостатній - ініціюється дозаказ (повторне замовлення).

Мал. 11.6. Кооперативна діаграма проходження замовлення

Повідомлення з'являються в тій послідовності, як вони показані на діаграмі - зверху вниз. Якщо передбачається відправка повідомлення об'єктом самому собі (самоделегіроване), то стрілка починається і закінчується на одній «лінії життя».

На діаграми може бути додана керуюча інформація: опис умов, при яких надсилається повідомлення; ознака багаторазової відправки повідомлення (маркер ітерації); ознака повернення повідомлення.

кооперативні діаграми

На кооперативних діаграмах об'єкти (або класи) показуються у вигляді прямокутників, а стрілками позначаються повідомлення, якими вони обмінюються в рамках одного варіанту використання. Тимчасова послідовність повідомлень відбивається їх нумерацією.

діаграми станів

Діаграми станів використовуються для опису поведінки складних систем. Вони визначають всі можливі стани, в яких може перебувати об'єкт, а також процес зміни станів об'єкта в результаті деяких событій.Еті діаграми зазвичай використовуються для опису поведінки одного об'єкта в декількох прецедентах.

Прямокутниками представляються стану, через які проходить об'єкт під час своєї поведінки. Станам відповідають певні значення атрибутів об'єктів. Стрілки являють переходи від одного стану до іншого, які викликаються виконанням деяких функцій об'єкта. Є також два види псевдо-станів: початковий стан, в якому знаходиться щойно створений об'єкт, і кінцевий стан, яке об'єкт не покидає, як тільки туди перейшов.

Переходи мають мітки, які синтаксично складаються з трьох необов'язкових частин (див. Рис. 11.7):

Мал. 11.7. Діаграма станів об'єкта «замовлення»

<

Подія> <[Умова]> </ Дія>.

На діаграмах також відображаються функції, які виконуються об'єктом в певному стані. Синтаксис мітки діяльності:

виконати / <діяльність>.

діаграми діяльності

Діаграма діяльності - це окремий випадок діаграми станів. На діаграмі діяльності представлені переходи потоку управління від однієї діяльності до іншої всередині системи. Цей вид діаграм зазвичай використовується для опису поведінки, що включає в себе безліч паралельних процесів.

Основними елементами діаграм діяльності є (рис. 11.8):

Мал. 11.8. Діаграма діяльності - обробка замовлення

• овали, що зображують дії об'єкта;

• лінійки синхронізації, що вказують на необхідність завершити або почати кілька дій (модель логічного умови «І»);

• ромби, що відображають прийняття рішень по вибору одного з маршрутів виконання процесу (модель логічного умови «АБО»);

• стрілки - відображають послідовність дій, можуть мати мітки умов.

На діаграмі діяльності можуть бути представлені дії, відповідні кількома варіантами використання. На таких діаграмах з'являється безліч початкових точок, оскільки вони відображають тепер реакцію системи на безліч зовнішніх подій. Таким чином, діаграми діяльності дозволяють отримати повну картину поведінки системи і легко оцінювати вплив змін в окремих випадках використання на кінцеве поведінку системи.

Будь-яка діяльність може бути піддана подальшій декомпозиції і представлена у вигляді окремої діаграми діяльності або специфікації (словесного опису).

діаграми компонентів

Діаграми компонентів дозволяють зобразити модель системи на фізичному рівні.

Елементами діаграми є компоненти - фізичні заміщаються модулі системи. Кожен компонент є повністю незалежним елементом системи. Різновидом компонентів є вузли. Вузол - це елемент реальної (фізичної) системи, який існує під час функціонування програмного комплексу і являє собою обчислювальний ресурс, зазвичай володіє як мінімум деяким об'ємом пам'яті, а часто ще й здатністю обробки. Вузли поділяються на два типи:

• пристрої - вузли системи, в яких дані не обробляються.

• процесори - вузли системи, що здійснюють обробку даних.

Для різних типів компонентів передбачені відповідні стереотипи в мові UML.

Компонентом може бути будь-який досить великий модульний об'єкт, такий як таблиця або екстент бази даних, підсистема, бінарний виконуваний файл,