ТЕМА 8: **МЕТОДОЛОГІЯ МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ**

1. СТРУКТУРНА МОДЕЛЬ ПРЕДМЕТНОЇ ОБЛАСТІ

В основі проектування ГІС лежить моделювання предметної області. Для того щоб одержати адекватний проект предметної області ГІС у вигляді системи правильно працюючих програм, необхідно володіти цілісним, системним поданням моделі, що відображає всі аспекти функціонування майбутньої геоінформаційної системи. При цьому під моделлю предметної області розуміється деяка система, що імітує структуру або функціонування досліджуваної предметної області та відповідає основній умові - бути адекватною для цієї області.

Попереднє моделювання предметної області дозволяє скоротити час і терміни проведення проектних робіт, а також отримати більш ефективний та якісний проект. Без проведення моделювання предметної області є велика ймовірність допущення великої кількості помилок у вирішенні стратегічних завдань, що призводить до економічних втрат і високих витрат на наступне перепроектування системи. Внаслідок цього всі сучасні технології проектування ГІС ґрунтуються на використанні методології моделювання предметної області.

До моделей предметних областей пред'являються наступні вимоги:

- *формалізація*, що забезпечує однозначний опис структури предметної області;
- *зрозумілість для замовників та розробників* на основі застосування графічних засобів відображення моделі;
- *гарантована реалізація*, яка передбачає наявність засобів фізичної реалізації моделі предметної області в ГІС;
- *забезпечення оцінки ефективності реалізації* моделі предметної області на основі певних методів і показників.

Для реалізації перерахованих вимог, як правило, будується **система моделей**, що відображає структурний та оціночний аспекти функціонування предметної області.

Для відображення структурного аспекту моделей предметних областей в основному використовуються графічні методи, які повинні гарантувати подання інформації про компоненти системи. *Головна вимога до графічних методів документування – простота.*

Структурний аспект передбачає створення:

- *об'єктної структури*, що відображає зміст взаємодіючих у процесах матеріальних та інформаційних об'єктів предметної області;
- *функціональної структури*, що відображає взаємозв'язок функцій (дій) перетворення об'єктів у процесах;
- *структури управління*, що відображає події та бізнес-правила, які впливають на виконання процесів;
- *організаційної структури*, яка відображає взаємодію організаційних одиниць підприємства і персоналу в процесах;
- *технічної структури*, яка описує топологію розміщення та способи комунікації комплексу технічних засобів.

Графічні методи повинні забезпечувати можливість структурної декомпозиції специфікацій системи з максимальним ступенем деталізації та погоджень опису на суміжних рівнях декомпозиції.

З моделюванням безпосередньо пов'язана проблема **вибору мови** подання проектних рішень, що дозволяє якнайбільше залучати майбутніх користувачів системи до її розробки. Мова моделювання – це нотація, в основному графічна, котра використається для опису проектів. Нотація являє собою сукупність графічних об'єктів, які використовуються у моделі. Нотація є синтаксисом мови моделювання. Мова моделювання, з одного боку, повина зробити рішення проектувальників зрозумілими користувачеві, з іншого боку, надавати проектувальникам засобів досить формалізованого й однозначного визначення проектних рішень, які підлягають реалізації у вигляді програмних комплексів, що утворять цілісну систему геоінформаційного програмного забезпечення.

Графічне відображення є найбільш ємною формою подання інформації. При цьому проектувальники повинні враховувати, що графічні методи документування не можуть повністю

забезпечити декомпозицію проектних рішень від постановки завдання проектування до реалізації проекту ГІС. Труднощі виникають при переході від етапу аналізу системи до етапу проектування й особливо до програмування.

Головний критерій адекватності структурної моделі предметної області полягає у функціональній повноті розроблювальної ГІС.

В основі різних методологій моделювання предметної області ГІС лежать принципи послідовної деталізації абстрактних категорій. Зазвичай, моделі грунтуються на трьох рівнях: на зовнішньому рівні (визначення вимог), на концептуальному рівні (специфікація вимог) і внутрішньому рівні (реалізація вимог). Так, на **зовнішньому рівні** модель відповідає на запитання, що повинна робити система, тобто встановлюються основні компоненти системи: об'єктів, функцій, подій, організаційних одиниць, технічних засобів. **На концептуальному рівні** модель відповідає на запитання, як повинна функціонувати система? **На внутрішньому рівні** модель відповідає на запитання: за допомогою яких програмно-технічних засобів реалізуються вимоги до системи? З позиції життєвого циклу ГІС описані рівні моделей відповідно будуються на етапах аналізу вимог, логічного (технічного) і фізичного проектування. Розглянемо особливості побудови моделей предметної області на трьох рівнях деталізації.

Визначимо ключові поняття структурного аналізу.

Операція – елементарна (неподільна) дія, яка виконується на одному робочому місці.

Функція – сукупність операцій, які згруповані за певною ознакою.

Бізнес-процес – пов'язана сукупність функцій, у ході виконання якого споживаються певні ресурси й створюється продукт (предмет, послуга, наукове відкриття, ідея), що представляє цінність для споживача.

Підпроцесс – це бізнес-процес, що є структурним елементом деякого бізнесу-процесу та є цінним для споживача.

Бізнес-модель – структурований графічний опис мережі процесів та операцій, пов'язаних з даними, документами, організаційними одиницями й іншими об'єктами, що відображають існуючу або передбачувану діяльність підприємства.

1.1. ОБ’ЄКТНА СТРУКТУРА

Об'єкт – це сутність, яка використовується при виконанні деякої функції або операції (перетворення, обробки, формування й т.д.). Об'єкти можуть мати динамічну або статичну природу: динамічні об'єкти використаються в одному циклі відтворення, наприклад замовлення на продукцію, платежі; статичні об'єкти використаються в багатьох циклах відтворення, наприклад, технічні засоби, персонал, запаси матеріалів.

На зовнішньому рівні деталізації моделі виділяються основні види матеріальних об'єктів (наприклад, сировина й матеріали, напівфабрикати, готові вироби, послуги) і основні види інформаційних об'єктів або документів (наприклад, замовлення, накладні, рахунки й т.д.).

На концептуальному рівні побудови моделі предметної області уточнюється склад класів об'єктів, визначаються їх атрибути та взаємозв'язки. У такий спосіб створюється узагальнене відображення структури предметної області.

Далі концептуальна модель **на внутрішньому рівні** відображається у вигляді файлів бази даних, вхідних і вихідних документів. Причому динамічні об'єкти представляються одиницями змінної інформації або документами, а статичні об'єкти – одиницями умовно-постійної інформації у вигляді списків, номенклатур, довідників, класифікаторів.

1.2. ФУНКЦІОНАЛЬНА СТРУКТУРА

Функція (операція) являє собою деякий перетворювач вхідних об'єктів у вихідні. Послідовність взаємозалежних на входах та виходах функцій становить бізнес-процес. Функція бізнесу-процесу може породжувати об'єкти будь-якої природи (матеріальні, грошові, інформаційні). Причому бізнеси-процеси й інформаційні процеси, як правило, нерозривні, тобто функції матеріального

процесу не можуть здійснюватися без інформаційної підтримки. Наприклад, відвантаження готової продукції здійснюється на основі документа «Замовлення», що, у свою чергу, породжує документ «Накладна», що супроводжує партію відвантаженого товару.

Функція може бути представлена однією дією або деякою сукупністю дій. В останньому випадку кожної функції може відповідати деякий процес, у якому можуть існувати свої підпроцессы, і т.д., доки кожна з підфункцій не буде представляти деяку недекомпозиційовану послідовність дій.

На зовнішньому рівні моделювання встановлюється список основних бізнесів-функцій або видів бізнесів-процесів. Звичайно таких функцій налічується 15-20.

На концептуальному рівні виділені функції декомпозиціонуються та будуються ієрархії взаємозалежних функцій.

На внутрішньому рівні відображається структура інформаційного процесу в комп'ютері: визначаються ієрархічні структури програмних модулів, які реалізують функції інформатизації проектів.

1.3. СТРУКТУРА УПРАВЛІННЯ

У сукупності функцій бізнесу-процесу можливі альтернативні або циклічні послідовності залежно від різних умов протікання процесу. Ці умови пов'язані з подіями, що відбуваються, у зовнішньому середовищі або в самих процесах і з утворенням певних станів об'єктів (наприклад, інформація прийнята, відкинута, відправлена на корегування). **Події** викликають виконання функцій, які, у свою чергу, змінюють стани об'єктів і формують нові події, і т.д., доки не буде завершено певний бізнес-процес. Тоді послідовність подій становить конкретну реалізацію бізнесу-процесу.

Кожна подія описується із двох позицій: **інформаційної** та **процедурної**. Інформаційно подія відображається у вигляді деякого повідомлення, що фіксує факт виконання деякої функції зміни стану або появи нового. Процедурно подія викликає виконання нової функції, і тому для кожного стану об'єкта повинні бути задані описи цих викликів. Таким чином, події виступають у дуальній ролі для виконання функцій бізнесів-процесів.

На зовнішньому рівні встановлюється список зовнішніх подій, які виникають через взаємодією підприємства із зовнішнім середовищем (платежі податків, відсотків по кредитах, поставки по контрактах і т.д.), і список цільових настанов, яким повинні відповідати бізнеси-процеси (регламент виконання процесів, підтримка рівня матеріальних запасів, рівень якості продукції й т.д.).

На концептуальному рівні встановлюються бізнеси-правила, що визначають умови виклику функцій при виникненні подій та досягненні станів об'єктів.

На внутрішньому рівні виконується формалізація бізнесів-правил у вигляді тригерів або викликів програмних модулів.

1.4. ОРГАНІЗАЦІЙНА СТРУКТУРА

Організаційна структура являє собою сукупність організаційних одиниць, як правило, пов'язаних ієрархічними відносинами та відносинами між процесами. Організаційна одиниця – це підрозділ об'єднання людей (персоналу) для виконання сукупності спільних функцій або бізнесів-процесів. У функціонально-орієнтованій організаційній структурі організаційна одиниця виконує набір функцій, які пред’являються до однієї функції управління, яка входить у різні процеси. У процесно-орієнтованій структурі організаційна одиниця виконує набір функцій, що входять в один тип процесу та пов’язана з різними функціями управління.

На зовнішньому рівні будується структурна модель підприємства у вигляді ієрархії підпорядкування організаційних одиниць або списків взаємодіючих підрозділів.

На концептуальному рівні для кожного підрозділу задається організаційно-штатна структура посад (ролей персоналу).

На внутрішньому рівні визначаються вимоги до прав доступу персоналу до автоматизируемым функцій інформаційної системи.

1.5. ТЕХНІЧНА СТРУКТУРА

Топологія визначає територіальне розміщення технічних засобів в структурних підрозділах підприємства, а комунікація – технічний спосіб реалізації взаємодії структурних підрозділів.

На зовнішньому рівні моделі визначаються типи технічних засобів обробки даних та їх розміщення в структурних підрозділах.

На концептуальному рівні визначаються способи комунікацій між технічними комплексами структурних підрозділів: фізичне переміщення документів, машинних носіїв, обмін інформацією каналами зв'язку й т.д.

На внутрішньому рівні будується модель «клієнт-серверної» архітектури мережі.

Описані моделі предметної області націлені на проектування окремих компонентів ГІС: даних, функціональних програмних модулів, програмних модулів управління, програмних модулів інтерфейсів користувачів, структури технічного комплексу. Для більш якісного проектування зазначених компонентів потрібно будувати моделі, які узгоджують різні компоненти ГІС між собою. У найпростішому випадку для таких моделей взаємодії можуть використовуватись матриці перехресних зсилок: «об'єкти – функції», «функції – події», «організаційні одиниці – функції», «організаційні одиниці – об'єкти», «організаційні одиниці – технічні засоби» і т.д. Такі матриці не наочні та не відбивають особливостей реалізації взаємодій.

Для правильного відображення взаємодій компонентів ІС важливо здійснювати спільне моделювання цих компонентів, особливо зі змістовної точки зору об'єктів і функцій. Методологія структурного системного аналізу істотно допомагає в вирішенні таких завдань.

Структурним аналізом називають метод дослідження системи, яки починається з її загального огляду, а потім деталізується, отримуючи ієрархічну структуру із все більшим числом рівнів. Для таких методів характерно: розбивка на рівні абстракції з обмеженим числом елементів (від 3 до 7); обмежений контекст, що включає тільки істотні деталі кожного рівня; використання строгих формальних правил запису; послідовне наближення до результату. Структурний аналіз грунтується на двох базових принципах - «поділи і володарюй» та принципі ієрархічної впорядкованості. Вирішення складних проблем шляхом їхньої декомпозиції на безліч менших незалежних завдань (так званих «чорних ящиків») та організація цих завдань у деревоподібні ієрархічні структури значно підвищують розуміння складних систем.

Існують різні методології структурного моделювання предметної області, серед яких варто виділити **функціонально-орієнтовані та об’єктно-орієнтовані методології.**

2. ФУНКЦИОНАЛЬНО-ОРІЄНТОВАНІ ТА ОБ’ЄКТНО-ОРІЄНТОВАНІ МЕТОДОЛОГІЇ ОПИСУ ПРЕДМЕТНОЇ ОБЛАСТІ

Процес моделювання може бути реалізований у рамках різних методик, які відрізняються насамперед своїм підходом до того, що являє собою організація для якої виконують моделювання. Відповідно до різного подання про організацію методики прийнято ділити на об'єктні й функціональні (структурні).

Об'єктні методики розглядають організацію, для якої виконують моделювання, як набір взаємодіючих об'єктів - виробничих одиниць. Об'єкт визначається як відчутна реальність - предмет або явище, що мають чітко обумовлену поведінку. Метою застосування такої методики є виділення об'єктів, що становлять організацію, і розподіляють між собою відповідальність за виконувані дії.

З функціональних методик найбільш відомою є методика IDEF. В ній розглядають організацію як набір функцій, що перетворює вхідний потік інформації у вихідний потік. Процес перетворення інформації споживає певні ресурси. Основна відмінність від об'єктної методики полягає в чіткому відділенні функцій (методів обробки даних) від самих даних.

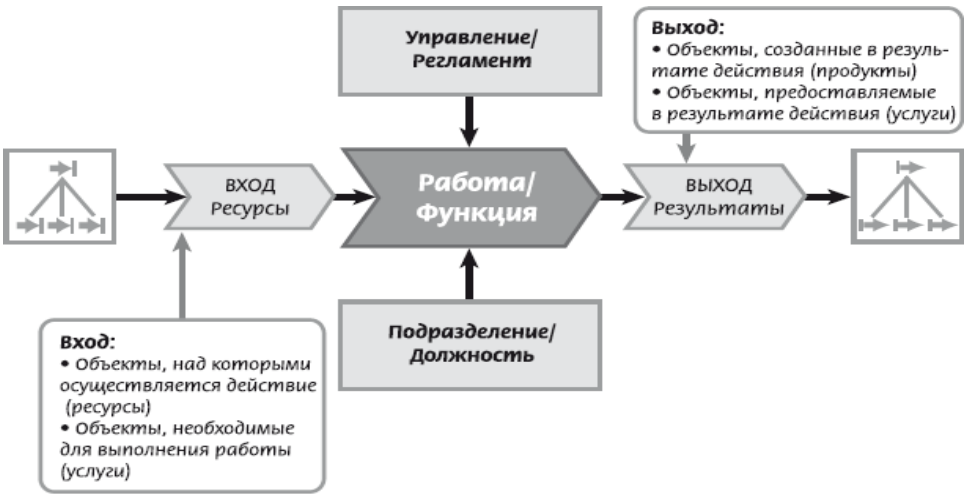


Рис. Основні об’єкти моделювання процесів

З погляду бізнес-моделювання кожний із представлених підходів має свої переваги. Об'єктний підхід дозволяє побудувати більше стійку до змін систему, краще відповідає існуючим структурам організації. Функціональне моделювання добре застосовувати в тих випадках, коли організаційна структура перебуває в процесі змін або взагалі слабко оформлена. Підхід з використанням виконуваних функцій інтуїтивно краще розуміється виконавцями при одержанні від них інформації про їхню поточну роботу.

Інтерес інженерів до опису технологічного ланцюжка став причиною появи та поширення методів процесного опису. У момент зародження процессного методу опису одержали популярність методи функціональної розбивки IDEF0 для опису «логічної послідовності» справ, і опису процесу IDEF3 для опису розгортання процесу в часі. IDEF0 - це метод і мова, що є варіантом методу структурного аналізу й проектування (Structural Analysis and Design Technique, SADT), які виникли наприкінці 60-х років минулого століття. IDEF0 відображає логічні відносини між роботами, але не дозволяє розглядати тимчасову послідовність. Пізніше з'явився IDEF3, що документує технологічні процеси. У ньому надана можливість описувати різні сценарії виконання роботи, а також описувати стан об'єктів та перехід між ними. IDEF0/IDEF3 успішно доповнюють один одного й широко використовуються системними інженерами.

Ідея необхідності безлічі груп описів призвела до появи мови UML. UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки застосунків. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

2.1 ФУНКЦІОНАЛЬНА МЕТОДИКА IDEF

IDEF - методології сімейства ICAM (Integrated Computer-Aided Manufacturing) для вирішення задач моделювання складних систем, дозволяють відображати і аналізувати моделі діяльності широкого спектру складних систем в різних розрізах. При цьому широта і глибина обстеження процесів в системі визначається самим розробником, що дозволяє не перевантажувати створювану модель зайвими даними.

IDEF - методології створювалися в рамках запропонованої ВПС США програми комп'ютеризації промисловості - ICAM, в ході реалізації якої виявилася потреба в розробці методів аналізу процесів

взаємодії у виробничих (промислових) системах. Принциповою вимогою при розробці даного сімейства методологій була можливість ефективного обміну інформацією між всіма фахівцями - учасниками програми ICAM (звідси назва: Icam DEFinition - IDEF інший варіант - Integrated DEFinition). Після опублікування стандарту він був успішно застосований в самих різних областях бізнесу, показавши себе ефективним засобом аналізу, конструювання і відображення бізнес-процесів. Більш того, власне з широким застосуванням IDEF і зв'язано виникнення основних ідей популярного нині поняття - BPR (бізнес-процес реінжиніринг).

Методології процесу моделювання за IDEF:

IDEF0 (Function Modeling) - даний метод використовується для створення функціональної моделі, яка є структурованим відображенням функцій виробничої системи або середовища, а також інформації та об'єктів, що зв'язують ці функції.

IDEF1 (Information Modeling) - даний метод застосовується для побудови інформаційної моделі, яка являє собою структуровану інформацію, необхідну для підтримки функцій виробничої системи або середовища.

IDEF2 (Simulation Model Design) - даний метод дозволяє побудувати динамічну модель, яка мінлива в часі та поведінці функцій, інформації та ресурсів виробничої системи або середовища. Дана модель використовується рідко. В основному затребувана на підприємствах, де необхідно описати безперервну дейтельності на конвеєрах або аналогічні функції.

IDEF3 (Process Description Capture) - даний метод використовується для збору інформації про стан системи, що моделюється.

Це структурний метод, який показує причинно-наслідкові зв'язки і події. Він також показує, як організована робота, і які користувачі працюють з модельованою системою. IDEF3 складається з двох методів. Process Flow Description (PFD) - опис процесів, з описом того, як організована робота між різними елементами системи, що моделюється. Object State Transition Description (OSTD) - опис переходів станів об'єктів, з описом того, які існують проміжні стани у об'єктів в моделюється системі.

IDEF4 (Object-Oriented Design) - даний метод об'єктно-орієнтованого планування був розроблений для підтримки об'єктно-орієнтованої ідеології. Докладніше - Технологія UML.

IDEF5 (Ontology Description Capture) - даний метод дозволяє розробляти, вивчати і підтримувати онтологію модельованої системи. Термін «онтологія» включає в себе каталог термінів галузі знань; правила, що пояснюють, як терміни можуть комбінуватися, створюючи при цьому коректні ситуації в галузі знань і узгоджені висновки, які використовуються в моделюється системі.

IDEF6 (Design Rational Capture Method) - даний метод дозволяє використовувати раціональний досвід проектування.

IDEF7 (Information System Auditing) - даний метод описує проведення методології аудиту інформаційної системи.

IDEF8 (User Interface Modeling) - даний метод дозволяє розробляти необхідні моделі Графічного Інтерфейсу Користувача (Human-System Interaction Design). Метод призначена для проектування взаємодії людини і технічної системи.

IDEF9 (Business Constraint Discovery) - дана модель призначена для аналізу наявних умов і обмежень (у тому числі фізичних, юридичних або будь-яких інших) і їх впливу на прийняття рішення в процесі реінжинірингу.

IDEF10 - Implementation Architecture Modeling.

IDEF11 - Information Artifact Modeling

IDEF12 - Organization Modeling

IDEF13 - Three Schema Mapping Design

IDEF14 (Network Design) - даний метод дозволяє моделювати обчислювальні мережі. Модель призначена для представлення та аналізу даних при проектуванні обчислювальних мереж на графічній мові з описом конфігурацій, черг, мережевих компонентів, вимог до надійності.

ПОНЯТТЯ МЕТОДОЛОГІЇ IDEF0

Історично **IDEF0** як стандарт був розроблений в 1981 році в рамках великої програми автоматизації промислових підприємств і остання його редакція була випущена в грудні 1993 року Національним Інститутом по Стандартах і Технологіям США (NIST).

Метою методики є побудова функціональної схеми досліджуваної системи, що описує всі необхідні процеси з точністю, яка достатня для однозначного моделювання діяльності системи.

В основі методології лежать чотири основних поняття:**функціональний блок, інтерфейсна дуга, декомпозиція, глосарій**.

Функціональний блок (Activity Box) являє собою деяку конкретну функцію в рамках розглянутої системи. За вимогами стандарту назва кожного функціонального блоку повинна бути сформульована як дієслово (наприклад, «робити послуги»). На діаграмі функціональний блок зображується прямокутником (рис. 6.1). Кожна із чотирьох сторін функціонального блоку має своє певне значення (роль), при цьому:

- верхня сторона має значення «Управління» (Control);
- ліва сторона має значення «Вхід» (Input);
- права сторона має значення «Вихід» (Output);
- нижня сторона має значення «Механізм» (Mechanism).

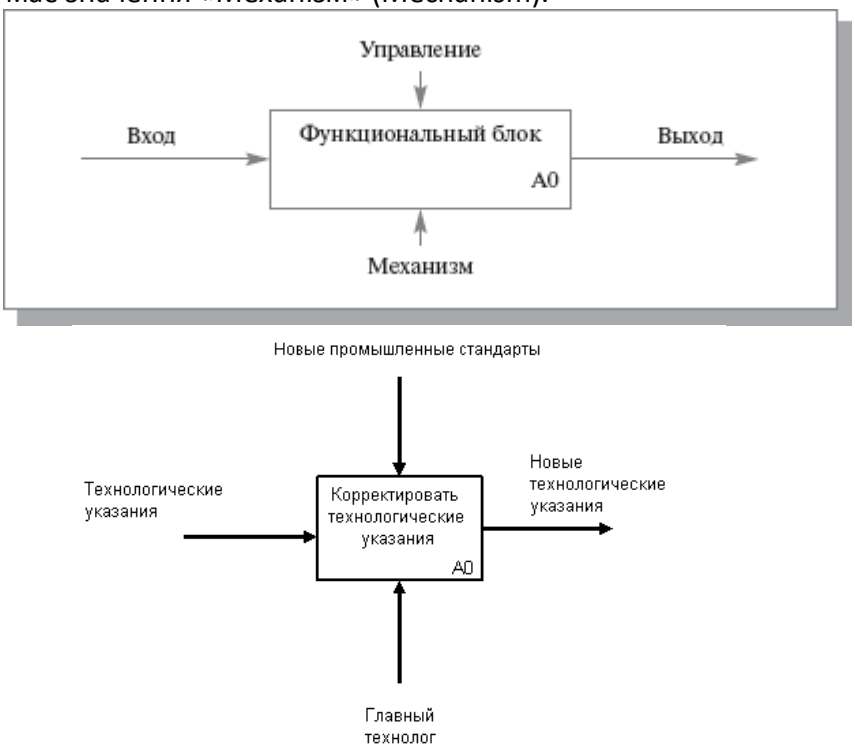


Рис. 6.1. Функціональний блок

Інтерфейсна дуга (Arrow) відображає елемент системи, що обробляється функціональним блоком або робить інший вплив на функцію, представлену даним функціональним блоком. Інтерфейсні дуги часто називають потоками або стрілками.

За допомогою Інтерфейсних дуг відображають різні об'єкти, які різним ступенем впливу визначають процеси, що відбуваються в системі. Такими об'єктами можуть бути елементи реального світу (ресурси, об’єкти, співробітники й т.д.) або потоки даних й інформації (документи, дані, інструкції й т.д.).

Залежно від того, до якої із сторін функціонального блоку підходить дана інтерфейсна дуга, вона зветься «вхідною», «вихідною» або «керуючою».

Необхідно відзначити, що будь-який функціональний блок за вимогами стандарту повинен мати, принаймні, одну керуючу інтерфейсную дугу та одну вихідну. Це й зрозуміло - кожен процес повинен

відбуватися за якимись правилами (які відображаються керуючою дугою) і повинен видавати деякий результат (вихідна дуга), інакше не має змісту його розглядати.

Обов'язкова наявність керуючих інтерфейсних дуг є одним з головних відмінностей стандарту IDEF0 від інших методологій класів DFD (Data Flow Diagram) і WFD (Work Flow Diagram).

Декомпозиція (Decomposition) є основним поняттям стандарту IDEF0. Принцип декомпозиції застосовується при розбивці складного процесу на складові його функції. При цьому рівень деталізації процесу визначається безпосередньо розробником моделі.

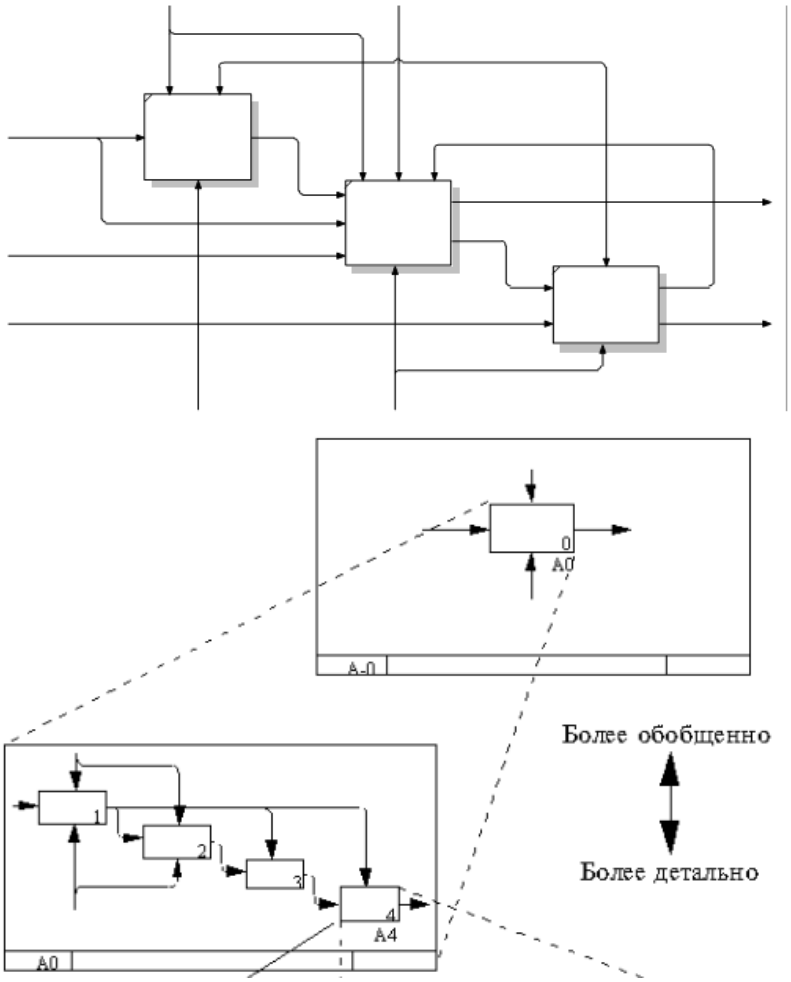


Рис. Декомпозиція проблеми за стандартом IDEF0

Декомпозиція дозволяє поступово та структуровано представляти модель системи у вигляді ієрархічної структури окремих діаграм, що робить її менш перевантаженою та легко засвоюваною.

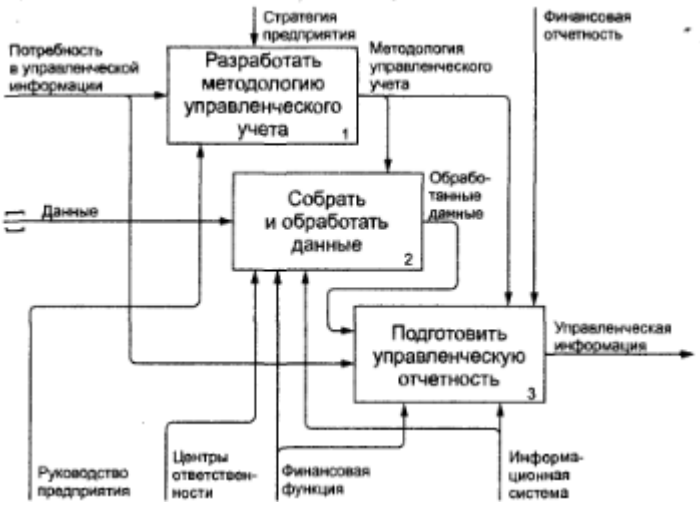


Рис. Приклад декомпозиції

Останнім з понять IDEF0 є **глосарій** (Glossary). Для кожного з елементів IDEF0 - діаграм, функціональних блоків, інтерфейсних дуг - існуючий стандарт має на увазі створення й підтримку

набору відповідних визначень, ключових слів, розповідей і т.д., які характеризують об'єкт, що відображається даним елементом. Цей набір називається глосарієй та є описом сутності даного елемента. Глосарій гармонійно доповнює наочна графічна мова, постачаючи діаграми необхідною додатковою інформацією.

Модель IDEF0 завжди починається з подання системи як єдиного цілого – одного функціонального блоку з інтерфейсними дугами, що виходять за межі розглянутої області. Така діаграма з одним функціональним блоком називається **контекстною діаграмою**.

У пояснювальному тексті до контекстної діаграми повинна бути зазначена **мета** (Purpose) побудови діаграми у вигляді короткого опису й зафіксована **точка зору** (Viewpoint).

Визначення й формалізація мети розробки IDEF0-моделі є вкрай важливим моментом. Фактично ціль визначає відповідні області в досліджуваній системі, на яких необхідно фокусуватися в першу чергу.

Точка зору визначає основний напрямок розвитку моделі й рівень необхідної деталізації. Чітке фіксування точки зору дозволяє розвантажити модель, відмовившись від деталізації та дослідження окремих елементів, що не є необхідними, виходячи з обраної точки зору на систему. Правильний вибір точки зору істотно скорочує тимчасові витрати на побудову кінцевої моделі.

Стандарт IDEF0 містить набір процедур, які дозволяють розробляти й погоджувати модель великою групою людей, що належать до різних областей діяльності системи, яка моделюється. Зазвичай процес розробки є ітеративним і складається з наступних умовних етапів:

1) Створення моделі групою фахівців, які відносяться до різних сфер діяльності підприємства. Ця група в термінах IDEF0 називається авторами (Authors). Побудова первісної моделі є динамічним процесом, протягом якого автори опитують компетентних осіб про структуру різних процесів, створюючи моделі діяльності підрозділів. При цьому них цікавлять відповіді на наступні питання:
   - Що надходить у підрозділ «на вході»?
   - Які функції й у якій послідовності виконуються в рамках підрозділу?
   - Хто є відповідальним за виконання кожної з функцій?
   - Чим керується виконавець при виконанні кожної з функцій?
   - Що є результатом роботи підрозділу (на виході)?

На основі наявних положень, документів і результатів опитувань створюється чернетка моделі (Model Draft).

2) Поширення чернетки для розгляду, погоджень і коментарів. На цій стадії відбувається обговорення чернетки моделі із широким колом компетентних осіб (у термінах IDEF0 - читачів) на підприємстві. При цьому кожна з діаграм чорнової моделі письмово критикується й коментується, а потім передається авторові. Автор, у свою чергу, також письмово погоджується із критикою або відкидає неї з викладом логіки ухвалення рішення й знову повертає відкоректовану чернетку для подальшого розгляду. Цей цикл триває доти, поки автори й читачі не прийдуть до єдиної думки.
3) Офіційне затвердження моделі. Затвердження погодженої моделі відбувається керівником робочої групи в тому випадку, якщо в авторів моделі й читачів відсутні розбіжності із приводу її адекватності. Остаточна модель являє собою погоджене подання про підприємство (систему) із заданої точки зору й для заданої мети.

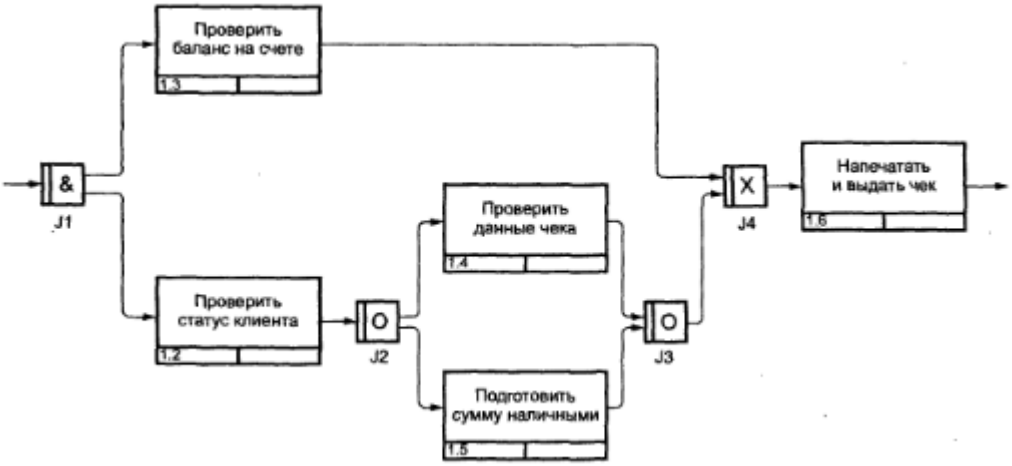


Рис. Опис процесу зняття грошей з банкомату в методології IDEF3

Наочність графічної мови IDEF0 робить модель цілком читабельною і для осіб, які не приймали участі в проекті її створення, а також ефективної для проведення показів і презентацій. Надалі на базі побудованої моделі можуть бути організовані нові проекти, націлені на виробництво змін у моделі.

IDEF0 найкраще використовувати як засіб аналізу та логічного моделювання систем, що, як правило, на ранніх стадіях роботи над проектом. Дані аналізу, які отримані з використаннм IDEF0-моделювання, зазвичай використовуються на стадії розробки моделей IDEF3 та діаграм потоків даних DFD (рис.)

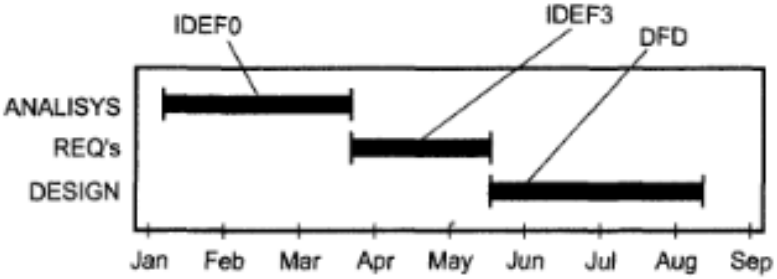


Рис. ..

2.2. ОБ’ЄКТНО-ОРІЄНТОВАНА МЕТОДИКА

ОСНОВНІ ПОНЯТТЯ

Принципова відмінність між функціональним та об'єктним підходами полягає в способі декомпозиції системи.

Об’єктно-орієнторований підхід використовує об'єктну декомпозицію, при цьому статична структура описується в термінах об'єктів і зв'язків між ними, а поводження системи описується в термінах обміну повідомленнями між об'єктами. Метою методики є побудова бізнес-моделі організації, яка дозволить перейти від моделі сценаріїв використання до моделі, що визначатиме окремі об'єкти, які братимуть участь у реалізації функцій.

Концептуальною основою об’єктно-орієнтованого підходу є об'єктна модель.

Модель - це певний (матеріальний або ні) об'єкт, який відображає лише найбільш значимі для даного завдання характеристики системи. Моделі бувають різні – матеріальні та нематеріальні, штучні та природні, декоративні та математичні...

Об’єктна модель будується з врахуванням наступних принципів:

- абстрагування;
- інкапсуляція — одна з трьох парадигм об'єктно-орієнтованого програмування і дизайну. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкту, замість цього надаючи інтерфейс взаємодії. Слугує передусім для того, щоб не давати можливості клієнту змінювати внутрішній стан об'єкта без відома останнього, залишаючи лише можливість впливу через «офіційні» методи роботи з полями (як правило, вони звуться setXXX і getXXX) та власне методи. Завдяки використанню інкапсуляції,

досягаються щонайменше дві переваги: можливість зміни внутрішнього представлення даних без необхідності зміни програмного коду клієнта; зменшення ймовірності помилок через приведення екземпляра до непередбаченого стану;

- модульність;
- ієрархія;
- типізація;
- паралелізм;
- стійкість (якість, що дозволяє системі витримувати зміни параметрів зовнішнього середовища, відмінні від розрахункових).

Основними поняттями об’єктно- орієнтованого підходу є об'єкт та клас.

Об'єкт — предмет або явище, який набуває різного стану, має чітку поведінку та індивідуальність. Структура та поведінка схожих об'єктів визначають загальний для них клас.

Клас – це множина об'єктів, зв'язаних спільною структурою та поведінкою.

Наступну групу важливих понять об'єктного підходу становлять успадкування й поліморфізм. Поняття **поліморфізм** може бути інтерпретовано як здатність класу належати більш ніж до одного типу. **Успадкування** означає побудова нових класів на основі існуючих з можливістю додавання або перевизначення даних і методів.

Важливою якістю об'єктного підходу є узгодженість моделей діяльності організації та моделей проектної інформаційної системи від стадії формування вимог до стадії реалізації. За об'єктними моделями може бути простежено відображення реальних сутностей предметної області, яка моделюється, в об'єкти й класи інформаційної системи.

Більшість існуючих методів об’єктно- орієнтованого підходу включають мову моделювання та опис процесу моделювання.

Процес – це опис кроків, які необхідно виконати при розробці проекту.

Як мова моделювання об'єктного підходу використається уніфікована мова моделювання UML, яка містить стандартний набір діаграм для моделювання.

Об’єктно- орієнтований підхід має наступні переваги:

- Об'єктна декомпозиція дає можливість створювати моделі меншого розміру шляхом використання загальних механізмів, які забезпечують необхідну економію засобів. Використання об'єктного підходу підвищує рівень уніфікації розробки й придатність для повторного використання.
- Об'єктна декомпозиція дозволяє уникнути створення складних моделей, тому що вона припускає еволюційний шлях розвитку моделі на базі невеликих підсистем.
- Об'єктна модель природна, оскільки орієнтирована на людське сприйняття середовища.

Мова UML – це мова візуального моделювання, яка розроблена для специфікації, візуалізації, проектування та документування компонентів програмного забезпечення, бізнесів-процесів та інших систем.

Мова UML –це простий та потужний засіб моделювання, який ефективно використовують для побудови концептуальних, логічних і графічних моделей складних систем різного цільового призначення. Ця мова увібрала в себе найкращі якості методів програмної інженерії, які з успіхом використовуються протягом останніх років при моделюванні великих та складних систем.

Мова UML грунтується на певному числі базових понять, які можуть бути вивчені й використані більшістю програмістів і розробників, які знайомі з методами об’єктно- орієнтованого аналізу та проектування. При цьому базові поняття можуть комбінуватися й розширюватися таким чином, що фахівці об'єктного моделювання одержують можливість самостійно розробляти моделі великих та складних систем у різних областях програмного проектування.

Конструктивне використання мови UML ґрунтується на розумінні загальних принципів моделювання складних систем й особливостей процесу об’єктно- орієнтованого аналізу та проектування. Вибір виразних засобів для побудови моделей складних систем визначає ті завдання, які можуть бути вирішені з використанням даних моделей. При цьому одним з основних принципів побудови моделей складних систем є принцип *абстрагування*, який пропонує включати в модель тільки ті аспекти проектованої системи, які мають безпосереднє відношення до виконання системою

своїх функцій або свого цільового призначення. При цьому всі другорядні деталі опускаються, щоб надмірно не ускладнювати процес аналізу й дослідження отриманої моделі.

Іншим принципом побудови моделей складних систем є принцип *багатомодельності*. Цей принцип означає, що ніяка єдина модель не може з достатнім ступенем адекватності описувати різні аспекти складної системи.

Ще одним принципом прикладного системного аналізу є принцип *ієрархічної побудови моделей* складних систем. Цей принцип пропонує розглядати процес побудови моделі на різних рівнях абстрагування або деталізації в рамках фіксованого подання. При цьому вихідна або первісна модель складної системи має найбільш загальне подання (метапредставлення). Така модель будується на початковому етапі проектування й може не містити багатьох деталей та аспектів системи, що моделюється.

Таким чином, процес об’єктно- орієнтованого аналізу можна представити як порівневий спуск від найбільш загальних моделей та представлень концептуального рівня до більш приватних та детальних представлень логічного й фізичного рівня. При цьому на кожному з етапів об’єктно-орієнтованого аналізу дані моделі послідовно доповнюються все більшою кількістю деталей, що дозволяє їм більш адекватно відображати різні аспекти конкретної реалізації складної системи. Загальна схема взаємозв'язків моделей ООАП представлена на мал. 3.2.

Рис. 3.2. Схема взаємозв'язків моделей та представлень складної системи в процесі об’єктно- орієнтованого аналізу й проектування

UML-моделювання дозволяє вирішити п'ять різних завдань:

1) формалізувати функціональні вимоги до системи;
2) візуалізувати систему в її поточному або цільовому стані;
3) визначити структуру та спостерігати за поведінкою системи;
4) отримати шаблон (інваріант), який дозволить синтезувати систему;
5) отримати формальні описи рішень на основі отриманих моделей.

ЗАГАЛЬНА СТРУКТУРА МОВИ UML

Діаграма – графічне подання набору елементів, яке зображається найчастіше у вигляді зв'язного графа (діаграми D) з вершинами (сутностями S) і ребрами (відносинами R).

$D = \langle S,R \rangle$, $S \neq \varnothing$, $R \subset S \times S$, $R = R^{-1}$.

Прикладів діаграм можна привести багато. Це й знайома нам усім зі шкільного років блок-схема, і схеми монтажу різного встаткування, які ми можемо бачити в руководствах користувача, і дерево файлів і каталогів на диску, що ми можемо побачити, виконавши в консолі Windows команду tree, і багато чого іншого. У повсякденному житті діаграми оточують нас із усіх боків, адже малюнок сприймається нами легше, ніж текст...

За допомогою діаграм можна візуалізувати систему з різних точок зору. Одна з діаграм, наприклад, може описувати взаємодію користувача із системою, інша – зміну станів системи в процесі її роботи, третя – взаємодію між собою елементів системи і т.д. Складну систему можна та й потрібно представляти у вигляді набору невеликих та частково незалежних моделей-діаграм, причому жодна з них не є достатньою для опису системи й одержання повного подання про неї, оскільки кожна з них фокусується на якомусь певному аспекті функціонування системи й виражає різний рівень абстракції. Інакше кажучи, кожна модель відповідає деякій певній, приватній точці зору на проекту систему.

Необхідно зазначити, що діаграма не є моделлю. Діаграми – це лише засіб візуалізації моделі, і ці два поняття необхідно розрізняти. Лише набір діаграм становить модель системи й найбільше повно її описує, а не одна діаграма, яка вирвана з контексту.

UML надає можливість залежно завдання дослідження формувати модель системи у вигляді сукупності діаграм, вибираючи з їхнього базового набору необхідні, які відображають особливі властивості системи.

Види діаграм

Стандарт UML 1.5 визначає дванадцять типів діаграм, розділених на три групи:

- чотири типи діаграм представляють статичну структуру додатків;
- п'ять представляють поведінкові аспекти системи;
- три представляють фізичні аспекти функціонування системи (діаграми реалізації).

У термінах мови UML визначені наступні види діаграм:

1) Діаграма варіантів використання (use case diagram)
2) Діаграма класів (class diagram)
3) Діаграми поводження (behavior diagrams)
4) Діаграма станів (statechart diagram)
5) Діаграма діяльності (activity diagram)
6) Діаграми взаємодії (interaction diagrams)
7) Діаграма послідовності (sequence diagram)
8) Діаграма кооперації (collaboration diagram)
9) Діаграми реалізації (implementation diagrams)
10) Діаграма компонентів (component diagram)
11) Діаграма розгортання (deployment diagram)

Втім, точне число канонічних діаграм для нас не є важливим, тому що ми розглянемо не всі з них, а лише деякі - з тієї причини, що кількість типів діаграм для конкретної моделі конкретного додатка не є строго фіксованим. Для простих додатків немає необхідності будувати всі без винятку діаграми. Наприклад, для локального додатка не обов'язково будувати діаграму розгортання. Важливо зрозуміти, що перелік діаграм залежить від специфіки розроблювального проекту та визначається самим розроблювачем.

Про всі діаграми UML можна перегляну на сайті, присвяченому стандарту (http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML).

Опис мови UML

Із самої загальної точки зору опис мови UML складається із двох взаємодіючих частин, таких як:

1) *Семантика мови UML* – деяка метамодель, яка визначає абстрактний синтаксис та семантику понять об'єктного моделювання мовою UML.
2) *Нотація мови UML* – графічна нотацію для візуального подання семантики мови UML.

Абстрактний синтаксис і семантика мови UML описуються з використанням деякої підмножини нотації UML. Нотація UML описує відповідність або відображення графічної нотації в базових поняттях семантики. Таким чином, з функціональної точки зору ці дві частини доповнюють одна одну. При цьому семантика мови UML описується на основі деякої метамоделі, яка має три окремих подання: абстрактний синтаксис, правила коректної побудови виразів та семантику. Розгляд семантики мови UML допускає певний «напівформальний» стиль викладу, що поєднує природний і формальну мови для подання базових понять і правил їх вирішення.

Семантика визначається для двох видів об'єктних моделей: структурних моделей і моделей поводження. Структурні моделі, відомі також як статичні моделі, описують структуру сутностей або компонентів деякої системи, включаючи їхні класи, інтерфейси, атрибути та відносини. Моделі поводження – це динамічні моделі, які описують поводження або функціонування об'єктів системи, включаючи їхні методи, взаємодію й співробітництво між ними, а також процес зміни станів окремих компонентів і системи в цілому.

Для вирішення такого широкого діапазону завдань моделювання розроблена досить повна семантика для всіх компонентів графічної нотації. Вимоги семантики мови UML конкретизуються при побудові окремих видів діаграм. Нотація мови UML містить у собі опис окремих семантичних елементів, які можуть застосовуватися при побудові діаграм.

З перерахованих вище діаграм деякі служать для позначення двох і більше інших підвидів діаграм.

Перелік цих діаграм й їхніх назв є канонічними в тому розумінні, що являють собою невід'ємну частину графічної нотації мови UML. Більше того, процес ООАП нерозривно пов'язаний із процесом побудови цих діаграм. При цьому сукупність побудованих у такий спосіб діаграм є самодостатньою в тому розумінні, що в них утримується вся інформація, що необхідна для реалізації проекту складної системи.

Кожна із цих діаграм деталізує й конкретизує різні подання про моделі складної системи в термінах мови UML. При цьому діаграма варіантів використання являє собою найбільш загальну концептуальну модель складної системи, що є вихідною для побудови всіх інших діаграм. Діаграма класів є, по своїй суті, логічною моделлю, що відбиває статичні аспекти структурної побудови складної системи.

Динамічні діаграми є різновидами логічної моделі, які відображають динамічні аспекти функціонування складної системи. І, нарешті, діаграми реалізації служать для подання фізичних компонентів складної системи й тому використовуються до її фізичної моделі. Таким чином, інтегрована модель складної системи в нотації UML (мал. 3.3) представляється у вигляді сукупності зазначених вище діаграм.

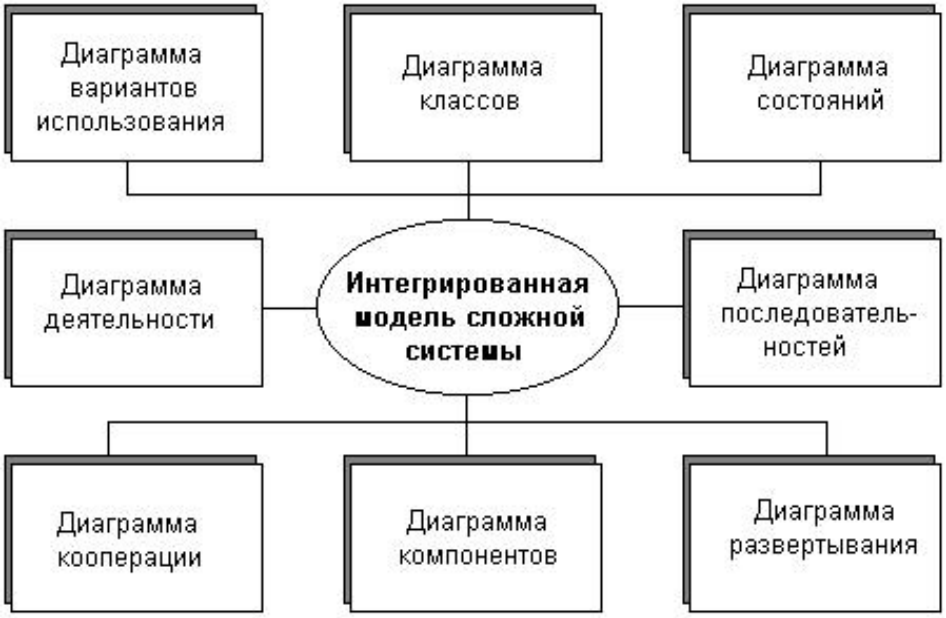


Рис. 3.3. Інтегрована модель складної системи в нотації UML

UML також містить кілька архітектурних конструкцій і механізмів розширення, які дозволяють додати системі модульну структуру, використовувану в процесі ітеративної й нарощуваної розробки:

При використанні нотації й семантики UML гарантується побудова *точної, семантично несуперечливої й повної моделі* (well-formed model), що гармоніює з усіма пов'язаними з нею моделями.

Розвиток або коректування побудованої моделі, які відображають модифікацію системи, здійснюється в UML без порушення устояної структури, шляхом видозміни поведінки об'єктів, що дозволяє усунути небажані наслідки у життєздатності системи.

Серед переліку загальцільових механізмів в UML передбачено механізми розширення (extensibility mechanisms), а саме: стереотипи (stereotypes), тегговані значення (tagged values) та обмежуючі умови (constraints), що принципово дозволяють розширювати:

- *словник* UML, вводячи нові типи конструктивних блоків, подібні до існуючих, але, які відображатимуть специфіку проблемної області;
- *властивості* конструктивних блоків, вводячи нову інформацію до специфікації елементу;
- *семантику* конструктивних блоків, які дозволяють вводити нові правила або модифікувати існуючі правила.

Діаграма прецентдентів (Use Case Diagram) перераховує вимоги, які повинна забезпечити система (інформаційна або виробнича). Основні елементи моделі - актор (користувач системи) і прецедент використання системи (сервіс, який надається).

Якщо мова йде про програмне забезпечення, то варіанти використання з деяким наближенням можна вважати автоматизованими операціями.

Основні поняття

Ектор (actor) – це множина логічно зв'язаних ролей, які виконують при взаємодії із прецедентами або сутностями (система, підсистема або клас). Ектором може бути людина або інша система, підсистема або клас, які представляють щось поза сутністю.

Слово «ектор» утворено від слова «action», що в перекладі означає дію. Дослівний переклад слова "ектор" – діюча особа.

Графічно ектор зображується "чоловічком".

Прецедент (use-case) – опис окремого аспекту поведінки системи з погляду користувача (*Booch*).

Прецедент (use case) - опис множини послідовних подій (включаючи варіанти), які виконуються системою та призводять до результату, за яким спостерігає ектор. Прецедент не показує, "як" досягається деякий результат, а тільки "що" саме виконується (*Zic Mentor*).

Прецеденти позначаються у вигляді еліпса, всередині якого вказуэться його назва. Прецеденти та ектори з'єднуються за допомогою ліній. Часто на одному з кінців лінії зображують стрілку, причому в напрямку, де запитують сервіс, інакше кажучи, чиїми послугами користуються.

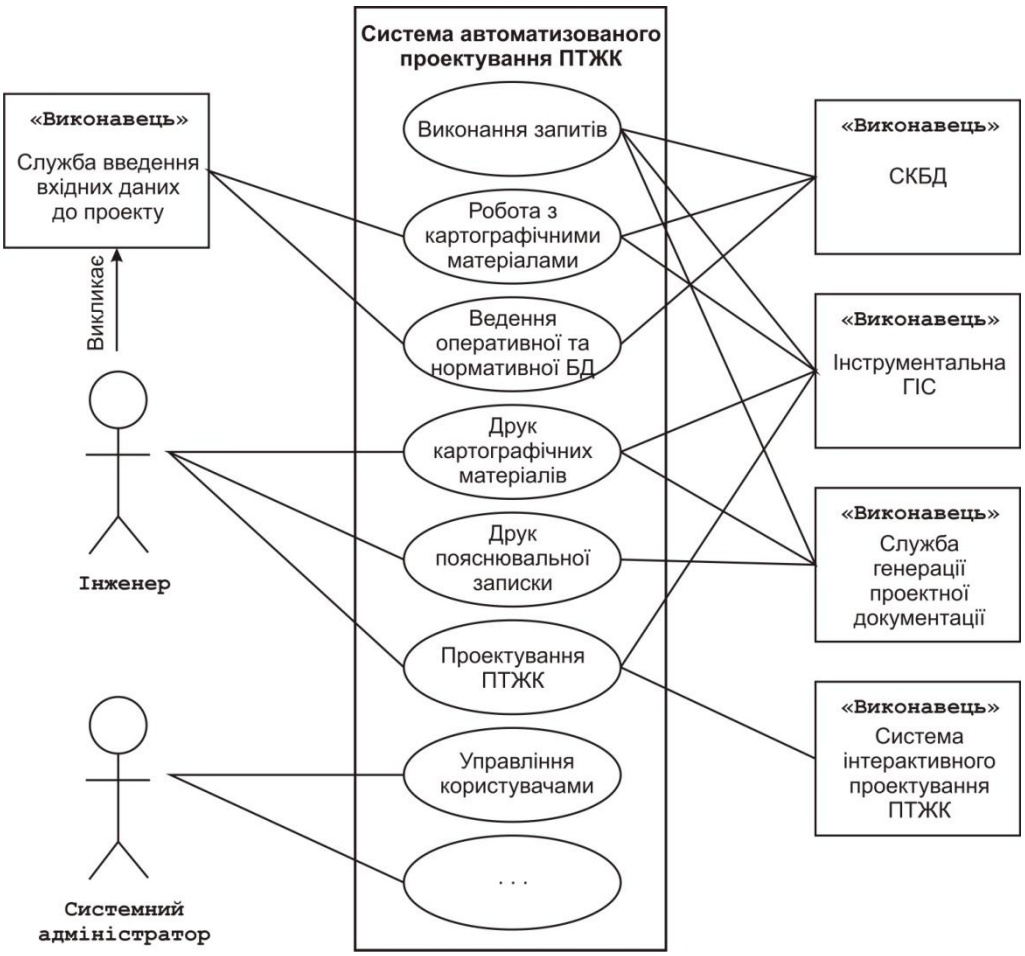


Рис. 3. Приклад діаграми прецендентів системи автоматизованого проектування прибудинкових територій

Підводячи підсумки, можна виділити такі області використання діаграм прецедентів:

- визначення меж та контексту моделювання предметної області на ранніх етапах проектування;
- формування загальних вимог до поводження проектованої системи;
- розробка концептуальної моделі системи для її наступної деталізації;
- підготовка документації для взаємодії із замовниками та користувачами системи.

Діаграма взаємодії (Interaction Diagram) описує порядок взаємодії учасників (об'єктів) у процесі реалізації варіанта використання системи. Існує два різновиди діаграм взаємодії - **діаграма послідовності** і **діаграма кооперації** (Collaboration Diagram). Ці діаграми фактично є поданнями однієї й тієї ж інформації в різному виді та часто доповнюють одна одну.

Діаграма послідовностей (Sequence Diagram) відображає тимчасові особливості передачі та прийому повідомлень об'єктами.

Діаграми послідовностей необхідно! використовувати для уточнення діаграм прецедентів, а саме детальнішого опису логіки сценаріїв використання. Це відмінний засіб документування проекту з погляду сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, та повертають результати, що, пов'язані з повідомленнями. Проте, результати, що повертають часто, позначають лише в тому випадку, якщо це не є очевидним з контексту.

Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів), *повідомлення* (виклики методів) – лініями зі стрілками, що повертають, а *результати* – пунктирними лініями зі стрілками. Прямокутники на вертикальних лініях під кожним з об'єктів показують "час життя" (фокус) об'єктів. Втім, досить часто їх не зображують на діаграмі, все це залежить від індивідуального стилю проектування.

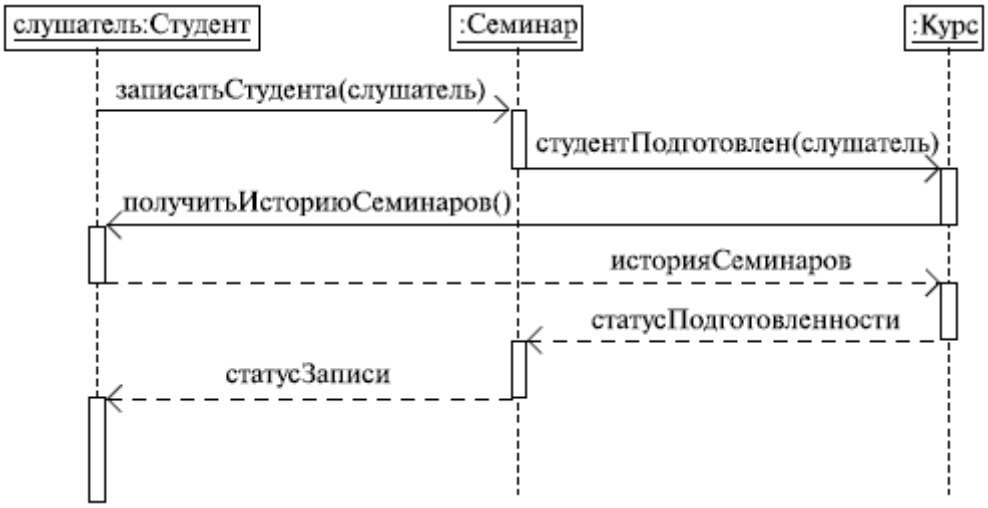


Рис. Приклад діагарми послідовностей

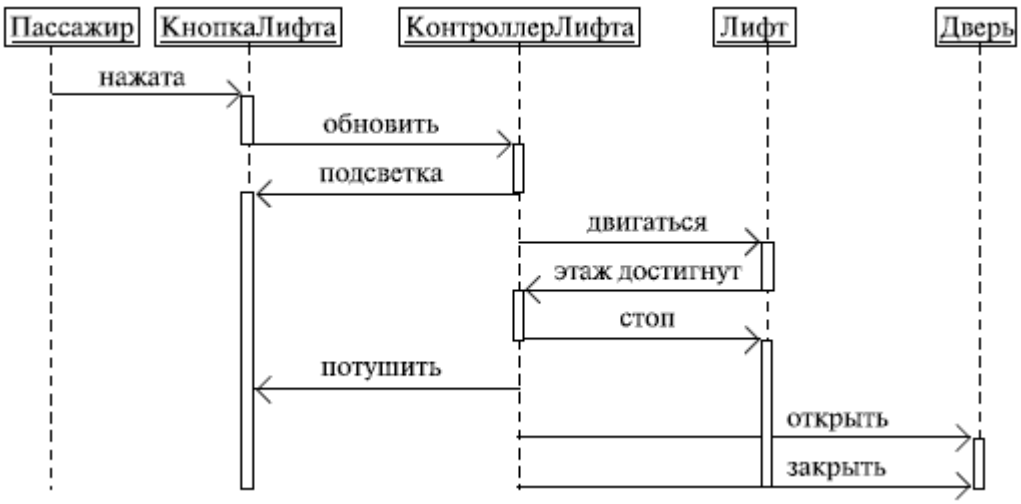


Рис. Приклад діагарми послідовностей

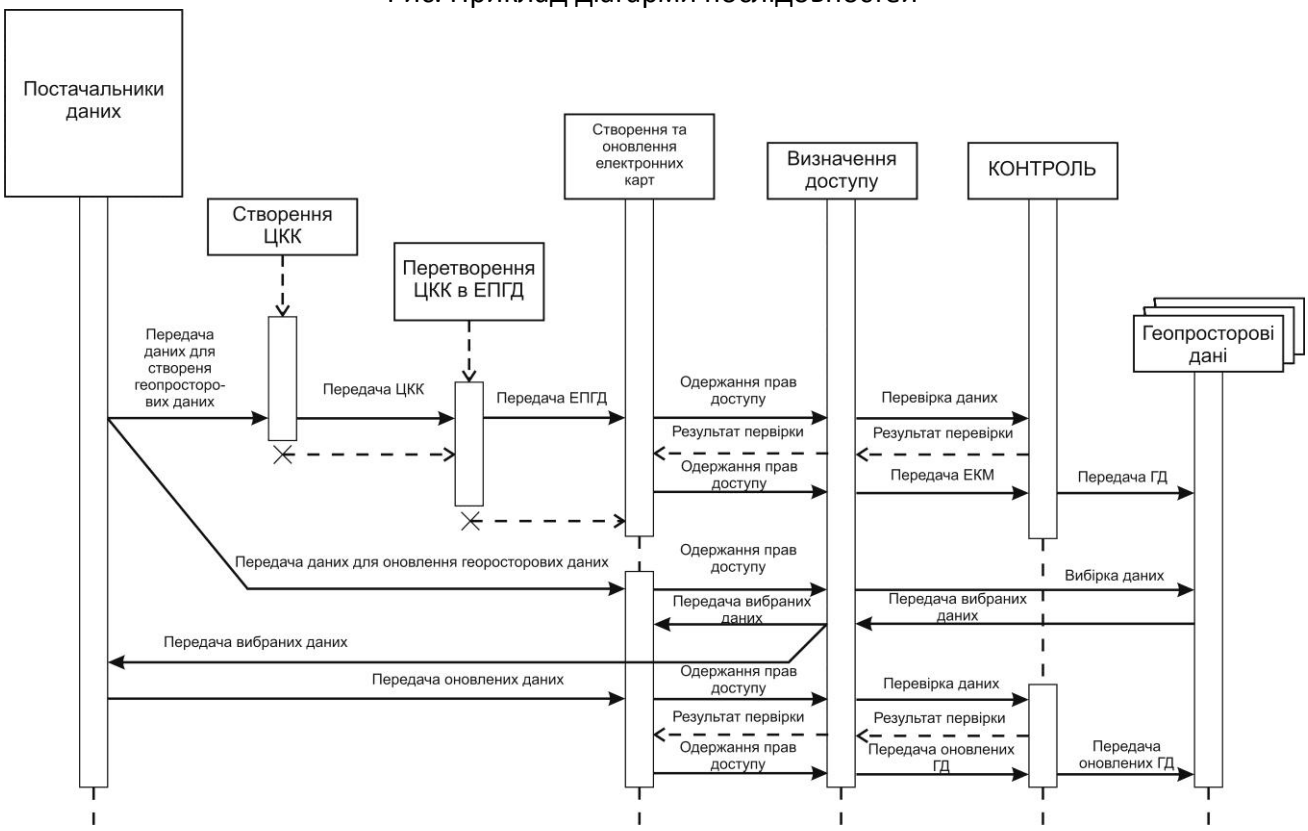


Рис. Приклад діаграми послідовностей

Необхідно зауважити, що використання діаграми послідовностей або діаграми взаємодії – це особистий вибір кожного проектувальника та залежить від індивідуального стилю проектування. При проектуванні геоінформаційних систем найчастіше віддають перевагу діаграмі послідовностей. Тому на діаграмі взаємодії зупинятися не будемо.

Діаграма класів (Class Diagram) є основною моделлю об'єктного методу. У деякому смислі можна вважати її розвитком діаграми «сутність - зв'язок».

Класи – це будівельні блоки будь-якої об’єктно-орієнтованої системи. Вони являють собою опис сукупності об'єктів із спільними атрибутами, операціями, відносинами й семантикою. При проектуванні об’єктно-орієнтованих систем діаграми класів обов'язкові!

Класи використовуються в процесі аналізу предметної області для складання словника предметної області розроблювальної системи. Це можуть бути як абстрактні поняття предметної області, так і класи, на які опирається розробка, та які описують програмні або апаратні сутності.

Діаграма класів - це набір статичних, декларативних елементів моделі. Діаграми класів можуть застосовуватися й при прямому проектуванні, тобто в процесі розробки нової системи, і при зворотному проектуванні - описі існуючих і використовуваних систем. Інформація з діаграми класів прямо відображається у вихідний код програми – у більшості існуючих інструментів UML-моделювання можлива кодогенерація для певної мови програмування (як правило Java або C++). Таким чином, діаграма класів – кінцевий результат проектування й відправна точка процесу розробки.

Основне застосування моделі класів - опис програмного забезпечення, хоча ця модель підходить і для опису виробничих систем.

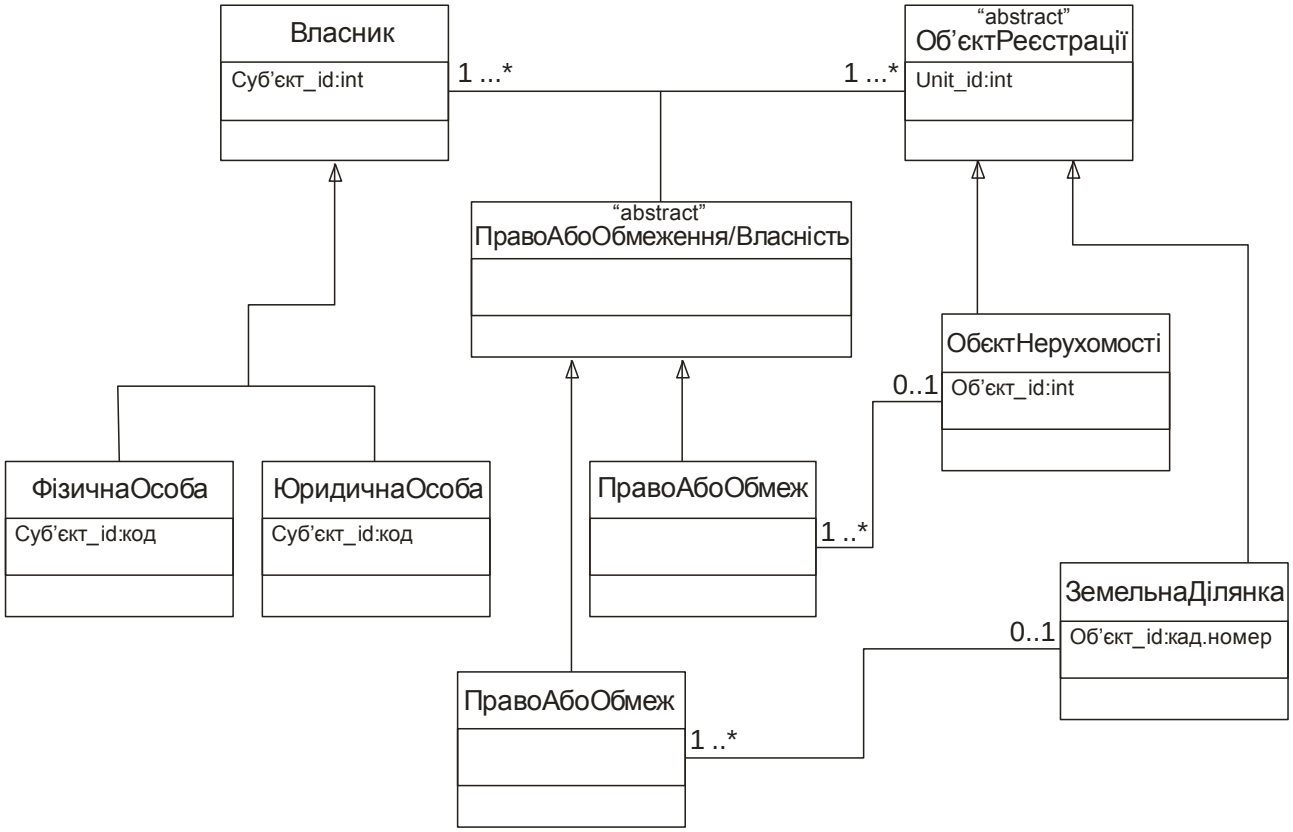


Рис. 2.11. Діаграма класів кадастрової моделі реєстрації нерухомості

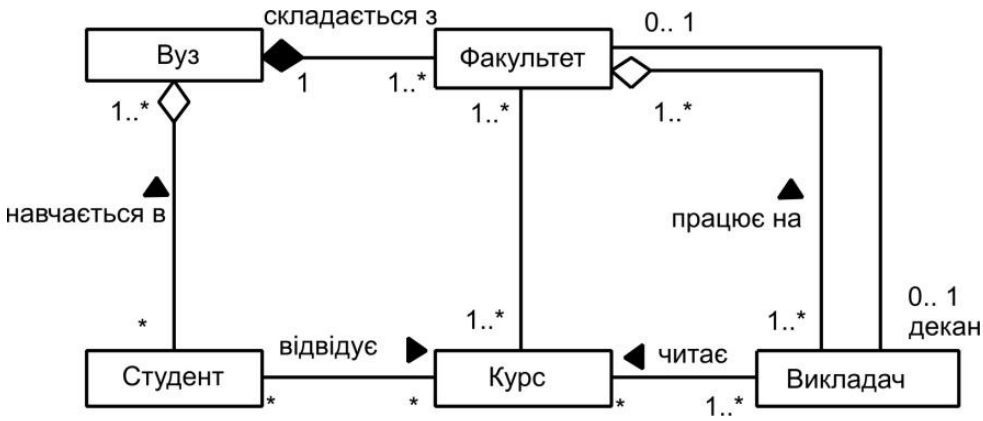
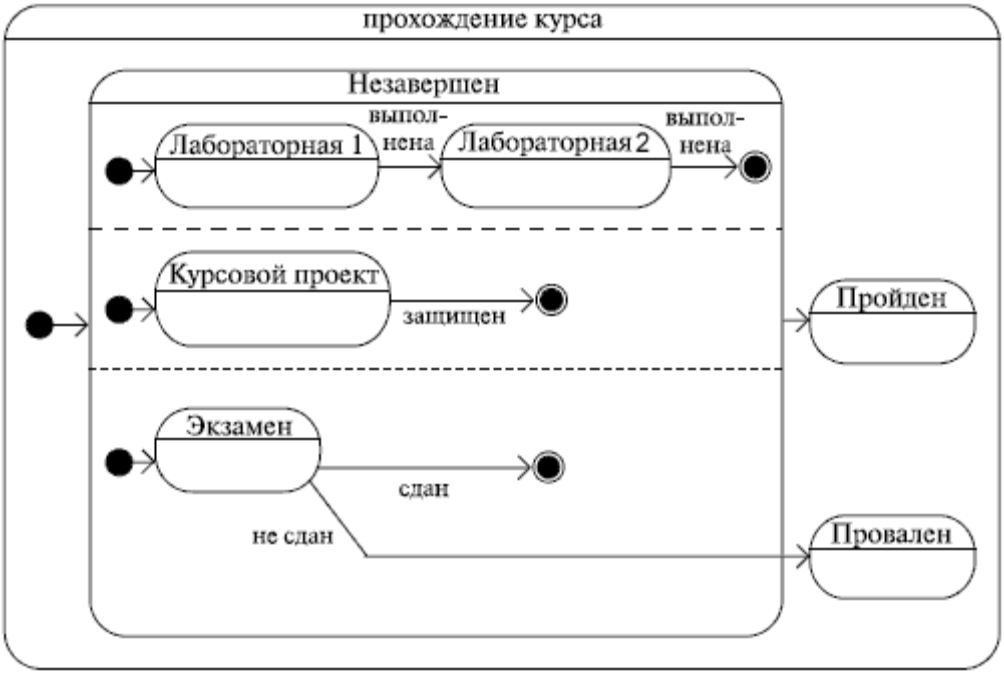


Рис. 2.11. Діаграма класів структури та процесу навчання

Діаграма станів (Statechart Diagram) – це модифікована діаграма переходів станів. Служить для специфікації станів об'єкта, що змінюються при різному перебігу подій.



Діаграма діяльності (Activity Diagram) – модель, яка схожа на блок-схему програми. Показує послідовність виконання операцій (діяльності, подій).

Моделювання складної системи об'єктними моделями зводиться до її опису в різних проекціях. Кожна з моделей є однією із проекцій системи в області вирішення задач, інтерпретуючи (синтаксично, семантично, якісно або кількісно) важливі для аналізу або синтезу властивості системи,

а також описує певний її аспект, а всі разом вони визначають систему з належним ступенем повноти, правильності та адекватності. Опис містить набір моделей, пов'язаних з поняттям класу/об'єкта, що поєднує дані (стан) і динаміку.

Досить часто функціональний (структурний) та об'єктний підходи протиставляють один одному. Багато в чому це результат конкуренції виробників технологій та інструментальних засобів. Однак користувачам продуктів та методик таке протиставлення нічого, крім шкоди, не приносить.

Самі по собі набори моделей ніяким чином не повинні нав'язувати принципів побудови систем. Проте, нерідко такі принципи вважаються невід'ємною характеристикою самих моделей.

Наприклад, структурний підхід часто помилково пов'язують із каскадним життєвим циклом. Це не так. І структурний, і об'єктний підходи успішно застосовуються як у спіральному, так і в каскадному життєвому циклі.

Необхідно чітко розрізняти моделі як засіб опису так і методи їх використання. Не потрібно недоліки окремих методів поширювати на моделі, які в них використовуються. Моделям функціонального (структурного) підходу ще далеко до забуття, і ніша у їх використанні буде зберігатися в майбутньому. Об'єктні моделі дозволили описати багато аспектів, перед якими структурні моделі були неспроможні. Але не можна вважати об'єктні моделі сучасною заміною моделей структурних. Вони також володіють деякими недоліками, що обмежують їх використання. Поки рано говорити про наявність дійсно універсальної мови опису предметних областей та їх програмного забезпечення. Потрібно використовувати всі доступні засоби. Чим багатший вибір таких засобів, тим кращого результату можна досягти.

Процес побудови окремих типів діаграм має свої особливості, які тісно пов'язані із семантикою елементів цих діаграм. Сам процес об’єктно-орієнтованого моделювання у контексті мови UML одержав спеціальну назву - раціональний уніфікований процес (Rational Unified Process, RUP). Концепція RUP й основні його елементи розроблені А. Джекобсоном у ході його роботи над мовою UML.

Суть концепції RUP полягає в послідовній декомпозиції або розбивці процесу об’єктно-орієнтованого моделювання на окремі етапи, на кожному з яких здійснюється розробка відповідних типів канонічних діаграм моделі системи. При цьому на початкових етапах RUP будуються логічні подання статичної моделі структури системи, потім - логічні подання моделі поведінки, і лише після цього - фізичні подання моделі системи. Зрозуміло, що у результаті RUP повинні бути побудовані канонічні діаграми мовою UML