Об’єктно-орієнтований аналіз

Що ми вкладаємо в поняття “добре спроектована об'єктно-орієнтована система”? Адже використання при розробці програмної системи об'єктно-орієнтованої мови програмування не гарантує створення об'єктно-орієнтованої програмної системи. Граді Буч у своїй книзі “Об'єктно-орієнтований аналіз і проектування” дає таке визначення об'єктно-орієнтованому програмуванню (object orienting programming, OOP):

> *“Об'єктно-орієнтоване програмування - методологія програмування, заснована на поданні програми у виді сукупності об'єктів, кожний із який є екземпляром певного класу, а класи утворять ієрархію спадкування.*

У даному визначенні можна виділити три частини: 1) ООР використовує в якості базових елементів об'єкти, а не алгоритми (ієрархія “бути частиною” …); 2) кожний об'єкт є екземпляром якогось класу; 3) класи організовані ієрархічно (ієрархія “is-a” …)...

Програма буде об'єктно-орієнтованої тільки при дотриманні всіх трьох зазначених вимог”.

Для того щоб створити програмне застосування, що відповідає цим вимогам необхідно на самих ранніх стадіях розробки притримуватися методології об'єктно-орієнтованого аналізу і проектування систем, що дозволить правильно виділити об'єкти, визначити їхню взаємодію й обов'язки, правильно спроектувати класи і їх функціональність.

> *“Об'єктно-орієнтоване проектування - це методологія проектування, що поєднує процес об'єктної декомпозиції і прийоми подання логічної і фізичної, а також статичної і динамічної моделей проектованої системи [Буч]”.*

Використання об'єктно-орієнтованої методології зараз нерозривно пов'язано з використанням мови UML (Unified Modeling Language - уніфікована мова моделювання), що “являє собою систему позначень, що базується на діаграмах і призначена для моделювання систем на основі об'єктно-орієнтованого підходу [Ларман]”.

Найбільш важним моментом об’єктно-орієнтованого аналізу і проектування є кваліфіковане розподілення обов’язків між компонентами програмної системи.

До розподілення обов’язків по степені важності більш всього примикають виділення об’єктів або абстракцій. Важливе місце займають обидва аспекти, але розподіленню обов’язків відводиться перше місце.

Для оволодіння навиками розподілення обов'язків Крег Ларман в [?] пропоную використовувати дев'ять фундаментальних принципів, що систематизовані в шаблони проектування GRASP.

Для створення програмного застосування необхідно описати проблему та вимоги до системи. Етап аналізу складається з дослідженні проблеми, а не в пошуках шляхів її розв'язку.

Під час розробки застосування необхідно також забезпечити високій рівень і детальний опис логіки розв'язку, що задовольняє вимоги до системи і існуючі обмеження. В процесі проектування основна увага приділяється логічному розв'язку, що забезпечує виконання основних вимог.

Основна ідея об'єктно-орієнтованого аналізу і проектування полягає втому, що предметна область і логічний розв'язок задачі розглядається розглядаються з точки зору об'єктів (понять або сутностей).

В процесі ООА основна увага приділяється визначенню та опису об'єктів (понять) в термінах предметної області. В процесі ОО проектування визначаються програмні об'єкти, які будуть реалізовані засобами об'єктно-орієнтованої мови програмування.

В процесі конструювання або ОО програмування забезпечується реалізація розроблених компонентів, таких як класи певною мовою програмування.

Об'єктно-орієнтований аналіз

Перш за все необхідно виділити економічні процеси , що відбуваються на підприємстві. В термінах ООА це називається *аналізом вимог*. Під цим розуміємо визначення економічних процесів та вимог і їх формулювання у вигляді прецедентів. Прецедент це текстовий опис процесів, що відбуваються на підприємстві або в системі.

Наступний крок – визначення видів діяльності виконавців або учасників процесу. В термінах ООА і П цей етап називається ООА предметної області і заключається в побудові концептуальної моделі. *Концептуальна модель* відображає різні категорії елементів предметної області: не тільки види діяльності учасників, але й всі поняття , що відносяться до цього.

Після ідентифікації процесів і видів діяльності співробітників підприємства настає час визначення, як виконуються процеси. Тут визначаються обов'язки співробітників по вирішенню задач, що є необхідними для виконання процесу, а також способи їх взаємодії.

Ця діяльність аналогічно ОО проектуванню, при якому основна увага приділяється розподіленню обов'язків. *Розподілення обов'язків* означає виділення задач і обов'язків різних програмних об'єктів в застосуванні подібно до розподілення функціональних обов'язків між співробітниками. Програмні об'єкти, як і люди, в процесі виконання своїх функцій зазвичай взаємодіють між собою.

Обов'язки об'єктів та їх взаємодію відображають за допомогою *діаграми класів* і *діаграми взаємодії*. На цих діаграмах відображаються класи та потоки повідомлень між програмними об'єктами.

Формування вимог до системи

Для того щоб вдало реалізувати проект, необхідно коректно сформулювати вимоги до системи.

Вимога (requirement) – це опис необхідних або бажаних властивостей продукту. Основною задачею на етапі визначення вимог є ідентифікація та документування необхідних властивостей системи у формі. На цій стадії рекомендується використовувати такі пункти:

- загальне формулювання задачі;
- споживачі системи;
- цілі;
- функції системи;
- атрибути системи. (стр. 63)

Опис процесів: прецеденти

Прецедент (use case) – документ, що описує послідовність подій, що пов'язані з виконавцем, який для здійснення необхідного процесу використовує систему. Прецедент є описом або варіантом використання системи.

За допомогою прецедентів описують певний процес, наприклад обробка ділової інформації. *Процес (process)* від початку і до кінця описує послідовність подій, дій та транзакцій, необхідних для досягнення певного результату або надання певного знання організації або виконавцю. Як приклад можна навести такі процеси:

- отримання грошей з банкомату;
- замовлення продукції;
- перевірка орфографії в документі;
- обробка телефонного дзвінка.

Типовою помилкою є подання прецедентом окремих кроків, операцій або транзакцій.

> Прецедент – це опис відносно великого завершеного процесу, в який зазвичай входить багато кроків або транзакцій. Як правило, окремі кроки або види діяльності у вигляді окремих прецедентів не подаються.

Види діяльності, складові частини або навіть окремі кроки певного прецеденту можна розділити на окремі більш дрібні прецеденти, що називаються *абстрактними (abstract use case)*, але така практика не є нормою і зараз розглядатися не буде.

Прецедент верхнього рівня – стисло описує процес в термінах предметної області.

Прецеденти описуються таблицею:

Прецедент	Назва прецеденту
Виконавець	Перелік виконавців (учасників)
Тип	(Див. далі)
Опис	Стислий опис процесу

Розгорнутий прецедент (expended use case) – надаю більш детальний опис, ніж прецедент верхнього рівня. Часто розгорнутий прецедент має форму діалогу між виконавцем та системою. Опис розгорнутого прецеденту складається з декількох частин.

1. Узагальнена інформація.

Прецедент	Назва прецеденту
Виконавець	Перелік виконавців (учасників)
Мета	Мета прецеденту
Опис	Стислий опис прецеденту високого рівня (копія)
Тип	1. Головний, другорядний або додатковий 2. Ідеальний або реальний
Посилання	Прецеденти або функції системи, що пов’язані з цим. Під час визначення функцій системи (етап формування вимог) всі вони мають бути розподілені по прецедентам. Це можна перевірити, переглянувши розділ *посилань (cross reference)* прецеденту.

2. Типовий хід подій.

Типовий хід подій	
Дії виконавця	**Відгук системи**
Пронумеровані дії виконавця	Пронумерований опис відгуків системи

3. Альтернативний хід подій.

Альтернативи	
Номер рядка в типовому ході події, для якого можливі альтернативні варіанти виконання	Відгук системи

Опис розгорнутого прецеденту слід починати за схемою:

1. Цей прецедент починається, коли < Виконавець > < ініціює подію >.
2. Цей прецедент починається, коли клієнт підходить до банкомату і вставляє картку.

Опис прецеденту може містити *точки прийняття рішень* або розгалуження. Якщо одне з цих рішень є типовим, а інші рідкісними, незвичайними або винятковими, то тільки типові випадок описується в типовому ході подій, а інші в альтернативному. Але в момент прийняття рішень можуть з'явитися рівноважні, рівноімовірні альтернативи, тоді використовується така схема опису прецеденту:

1. В основному розділі типового ходу подій вказуються всі можливі підрозділи.
2. Кожний підрозділ описується як типовий хід подій. Нумерація подій в кожному розділі починається з одиниці.
3. Якщо підрозділи також мають точки прийняття рішень, то вони записуються як альтернативи для кожного підрозділу.

Головний розділ	
Типовий хід подій	
Дії виконавця	**Відгук системи**
…	
Альтернативи	
…	
Підрозділ "Назва підрозділу"	
Типовий хід подій	
Дії виконавця	**Відгук системи**
…	
Альтернативи	
…	

Ідеальний прецедент (essential use case) – це розгорнутий прецедент, що виражає тільки сутність процесу без деталізації його реалізації. Проектні рішення, особливо ті, що пов'язані з реалізацією інтерфейсу користувача, при цьому не розглядаються.

Реальний прецедент (real use case) – конкретно описує процес в термінах проектних рішень, на основі конкретних технологій введення/ виведення інформації і т. п. Коли йдеться про інтерфейс користувача реальний прецедент визначає вміст діалогових вікон (або описує способи взаємодії з конкретним пристроєм) та обговорення низькорівневої взаємодії користувача з елементами інтерфейсу.

Назва прецеденту

Назви прецедентів повинні починатися з дієслова або іменника, що описує процес, наприклад:

- ввести замовлення;
- покупка товару.

Виконавці

Виконавці (actor) є зовнішнім відносно до системи поняттям, яке певним чином приймає участь в процесі, що описує прецедент. Зазвичай за допомогою вхідних подій виконавець стимулює систему або отримує від неї певну інформацію.

В прецеденті бере участь один *ініціюючий виконавець (initiator actor)*, який генерую початковий вплив і, можливо, декілька *виконавців, що беруть участь (participating actor)*.

Виконавці бувають такими:

- ролі, що виконують люди;
- комп'ютерні системи;
- електричні, або механічні пристрої.

Тип прецеденту

Прецеденти необхідно ранжувати за трьома категоріями: основні, другорядні і додаткові, що надалі дає можливість визначити пріоритет їх розробки.

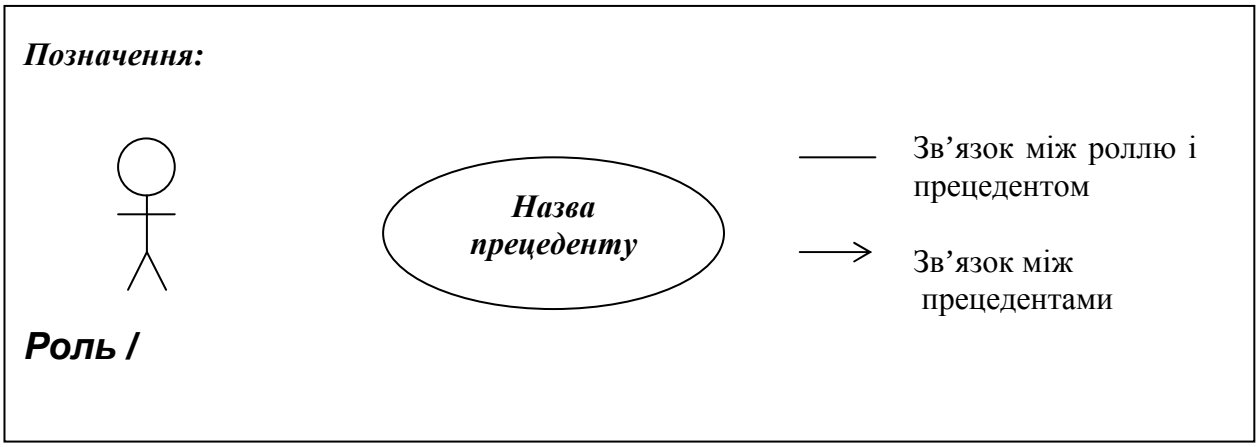
Основні прецеденти (primary use case) – являють найбільш загальні процеси.

Другорядні прецеденти (secondary use case) – являють менш значущі або додаткові процеси.

Додаткові прецеденти (optional use case) – описують процеси, які можуть не реалізовуватися системою.

Діаграма прецедентів

На *діаграмі прецедентів (use case diagram)* ілюструються набір прецедентів системи та їх виконавців, а також зв'язки між ними. Прецеденти зображуються овалами, а виконавці – умовним позначенням. Між прецедентами та виконавцями лініями вказуються зв'язки. Між прецедентами відображаються стрілками потоки даних або зв'язки між об'єкти, що впливають один на другий.



Призначення діаграми – надати певну контекстну діаграму, що дозволяє швидко визначити зовнішніх виконавців і ключові методи їх використання.

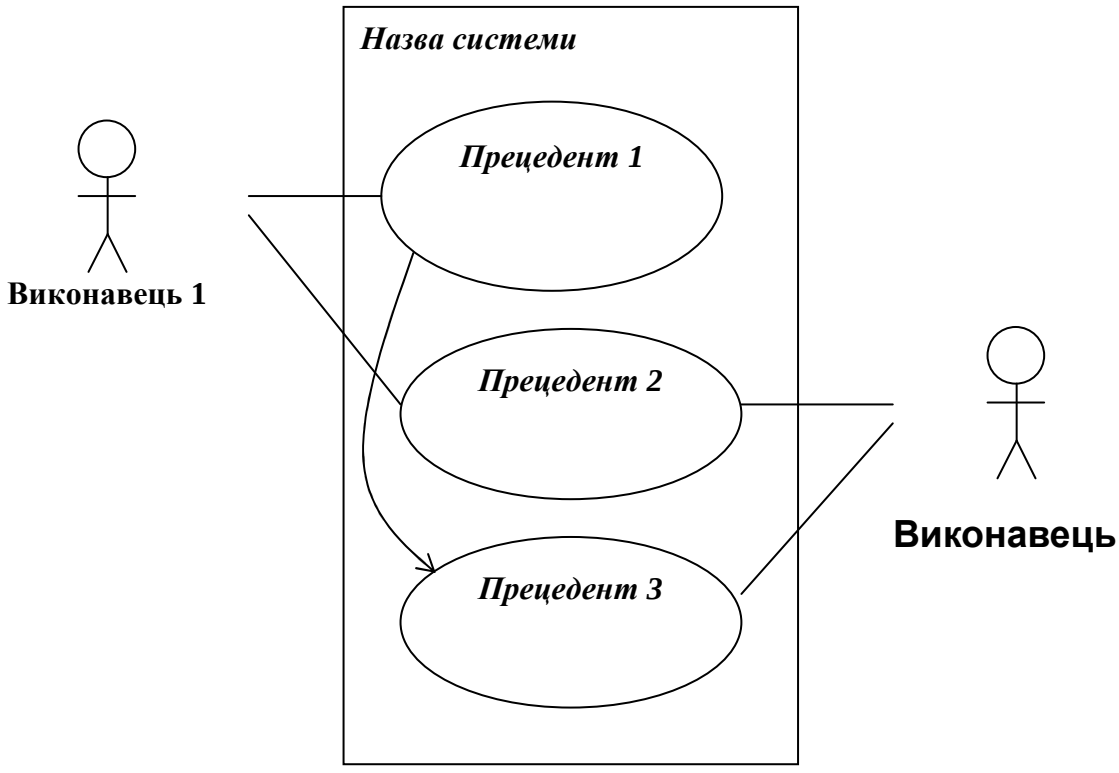


Рис. 1. Загальний вигляд діаграми прецедентів (варіантів використання)

СТР. 81???

Побудова концептуальної моделі

Концептуальна модель відображає основні (з точки зору того хто моделює) поняття предметної області. Основною задачею ООА є ідентифікація великої кількості різноманітних об'єктів або понять. Ідентифікація понять – основна частина дослідження предметної області. Концептуальна модель строїться на основі статичних структурних діаграм.

Важливою властивістю концептуальної моделі є подання понять реального світу, а не програмних компонентів.

Процес створення концептуальної моделі може виконуватися паралельно зі створенням прецедентів. У разі необхідності (якщо прецеденти не мають нічого спільного) для кожного прецеденту може бути побудована своя концептуальна модель.

Основною складовою ООА і дослідження є декомпозиція проблеми на відокремлені поняття або об'єкти. *Концептуальна модель* – це подання понять в термінах предметної області. Мовою UML концептуальна модель подається у вигляді *статичних структурних діаграм*.

Концептуальна модель може відображати:

- поняття;
- асоціації між поняттями;
- атрибути понять.

Концептуальна модель надає можливість не тільки здійснити декомпозицію проблеми на об'єкти (поняття), але і допомагає сформувати термінологію і скласти словник термінів предметної області. Цей словник допомагає розробнику виділити найбільш суттєві терміни і зв'язки між ними.

Поняття

Поняття – це подання ідей або об'єктів. Як вам вже відомо головне відмінність об'єктно-орієнтованого підходу від структурного є декомпозиція предметної області на об'єкти (поняття). Тому основною задачею ООА є визначення понять і подання результатів на концептуальній моделі.

Під час ідентифікації понять краще керуватися таким принципом:

Краще більш деталізувати концептуальну модель, ніж недовизначити її.

Не слід не брати до розгляду (виключати з концептуальної моделі) поняття, що, наприклад, не мають атрибутів або з аналізу вимог не будуть оброблятися системою (не грають інформаційної ролі).

Для того щоб визначитись, які поняття є в предметній області можна керуватися двома правилами:

1. вибирати поняття з предметної області, що належать до певних категорій, перелік яких пропонується в [Ларман]:

- ♦ фізичні або матеріальні об’єкти ();
- ♦ специфікації, елементи дизайну або опису об’єктів (специфікація товару, опис польоту);
- ♦ місця (магазин, аеропорт);
- ♦ транзакції або елементи транзакцій;
- ♦ ролі людей;
- ♦ контейнери інших об’єктів;
- ♦ вміст контейнерів;
- ♦ інші комп’ютери, електричні системи, що є зовнішніми відносно до даної;
- ♦ абстрактні поняття (голод, спрага);
- ♦ організації;
- ♦ події;
- ♦ і т. інше.

2. скласти список понять-кандидатів, виписавши всі іменники, які зустрічалися під час аналізу. Тепер треба визначитись, чи є всі перелічені іменники поняттями (або сутностями) предметної області. Існує декілька правил:

1. Якщо деякий об’єкт X в реальному мирі описується одним числом або рядком символів, то, скоріш за все, це буде атрибут, а не поняття.
2. Якщо деякий об’єкт X є підмножиною іншої множини але ця множина в іншому стані, то він не є окремим поняттям предметної області.
3. Якщо виникають сумніви при розподілі на поняття і атрибути, краще створіть окреме поняття.

Додавання асоціацій

Тепер до концептуальної моделі необхідно додати асоціації, де під *асоціацією (association)* розуміємо зв’язки між поняттями, що відображають певне суттєве та корисне відношення між ними [Ларман]. Зазвичай до концептуальної моделі включають такі асоціації:

1. Асоціації, знання про які необхідно зберігати протягом певного періоду.
2. Асоціації, похідні від тих, що в списку стандартних асоціацій.

Розглянемо основні з них:

- ♦ A є фізичною частиною B (літак – крило літака);

- A є логічною частиною B (список товарів – запис про товар);
- A використовує, взаємодіє або управляє B (касир – касовий апарат, робітник – компанія, завідувач кафедри – кафедра);
- A є власністю B (залікова книжка - студент).

Під час вибору асоціацій запобігайте використання асоціацій, що повторюються або є надлишковими.

На концептуальній моделі кожна асоціація позначається лінією, що проведена між поняттями. Асоціація моє своє ім'я, яке формулюється у дієслівній формі ("працює", "належить", "включає" і т.п.). Зазвичай асоціація двоспрямована, наприклад, поняття "студент" та "факультет" може пов'язувати асоціація "навчається". Це можна прочитати як "студент навчається на факультеті" або "на факультеті навчаються студент", тобто ця асоціація двоспрямована.

На кінцях лінії, що позначає асоціацію, можуть розташовуватися вирази, що визначають кількісний зв'язок між екземплярами понять. Ці вирази називають кратністю зв'язку та можуть бути:

- * - "багато";
- 1..* - один або багато;
- 1..10 – від одного до 10;
- 5 – рівно п'ять;
- 3, 5, 8 – рівно три, п'ять або вісім.

Крег Ларман дає такі рекомендації щодо пошуку асоціацій:

1. Зусередтеся на тих асоціаціях, для яких дані о зв'язках повинні зберігатися протягом певного часу (важливі асоціації).
2. Більш важливим є ідентифікація поняття, ніж асоціації.
3. Велика кількість асоціацій призводить до помилок в концептуальній моделі, а не до її спрощення. Вивчення асоціацій не повинно забирати багато часу, але повинно приносити максимальну користь.
4. Запобігайте збиткових асоціацій.

Додавання атрибутів

Останнім елементом концептуальної моделі є атрибути понять.

Атрибут (attribute) – це абстрактна властивість об'єкта.

В концептуальну модель включаються ті атрибути, для яких визначені відповідні вимоги (наприклад, прецеденти) або для яких передбачається, що необхідно зберігати певну інформацію.

В концептуальній моделі атрибут має бути простим атрибутом або простими даними. К стандартним простим типам відносять Boolean, Date, Number,

String(Text), Time [Ларман].
Іншими стандартними типами є Address, Color, Geometrics (Point, Rectangle), Phone Number, Universal Code, тип перелічення і т.п.
Помилкою є включення в якості атрибутів первинних ключів (foreign key)

Поведінка системи: діаграми послідовностей

Перш ніж розпочати розробку логіки роботи програмного застосування необхідно дослідити і визначити його поведінку як "чорної скрині". *Поведінка системи (system behavior)* являє собою опис того, які дії виконує система, без визначеного механізму їх реалізації. Однією з складових такого опису є діаграма послідовностей.

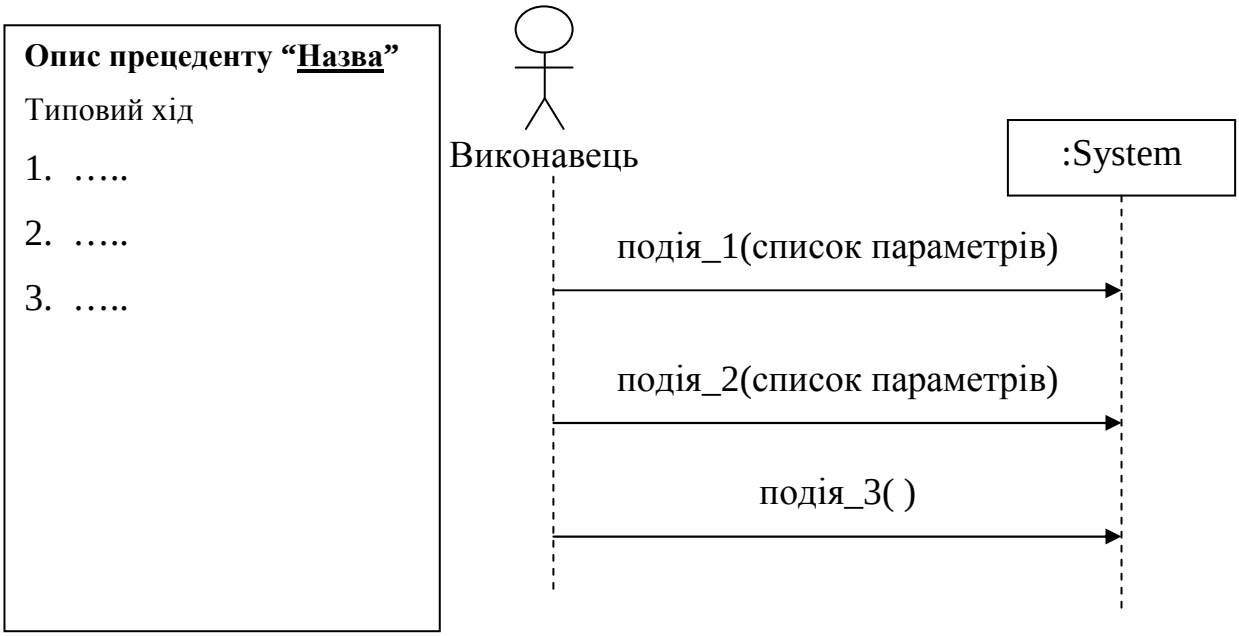
Прецеденти визначають як виконавець взаємодіє з програмною системою. Під час цієї взаємодії виконавець генерує певні події, що передаються системі. Ці події ініціюють в системі виконання певних операцій.

Діаграма послідовностей (system sequence diagram) – ще схема, яка для певного сценарію прецеденту показує події, що генеруються зовнішнім виконавцем, їх порядок, а також події, що генеруються в самій системі. При цьому системи розглядаються як "чорна скриня". Призначення цієї діаграми – відображення подій, що передаються зовнішніми виконавцями системі через її межі. Діаграма послідовностей створюється для типового ходу подій прецеденту, а у разі необхідності і для найбільш суттєвих альтернатив.

Для того щоб побудувати діаграму послідовностей необхідно:

1. Зобразити систему як "чорну скриню". Намалювати для неї вертикальну пунктирну лінію.
2. Ідентифікувати кожного виконавця, що безпосередньо впливає на систему (взаємодіє з нею). Для кожного з виконавців також намалюйте вертикальну лінію.
3. З опису типового ходу подій прецеденту визначте зовнішні системні події, які генеруються кожним з виконавців. Проілюструйте їх на діаграмі.

4. Додатково зліва на діаграмі можна розташувати опис прецеденту.



Імена системних подій краще розпочинати з дієслова, що відображає сенс події: add…(додати), enter…(ввести), make…(зробити), end…(закінчити).

Опис системних операцій

Перш ніж розпочинати розробку програмної реалізації доцільно дослідження поведінку системи як "чорної скрині". *Поведінка системи* – це опис функцій системи без пояснення принципів їх реалізації в термінах зміни стану системи при виконанні системних операцій.

Опис системної операції (contract) – це документ, що описує результати виконання операції. Зазвичай він складається в декларативному стилі і акцентує увагу на том, *що* має статися, а не на тому, *як* цього досягти.

Опис системної операції (system operation contract) – описує зміну у стані всієї системи при виконанні певної системної операції.

	Опис
Ім'я	ім'я_операції (список параметрів)
Обов'язки	Короткий опис змісту (обов'язків) операції або її цілей
Тип	Системна
Посилання	Функції системи: посилання на номера функцій Прецедент: назва прецеденту
Примітки	Певні пропозиція щодо алгоритмі, конструкторських рішень, деталей і т. п. виконання операції
Винятки	Опис виняткових ситуацій, що можуть виникнути під час виконання операції (наприклад, введені помилкові дані)

	Опис
Вивід	Вивід інформації, що не стосується інтерфейсу користувача (наприклад, повідомлення або записи) та відправляється за межі системи
Передумови	Опис стану системи, що необхідний для виконання операції. До цього відносяться: ♦ фактори, існування яких необхідно перевірити в програмі до початку виконання операції; ♦ фактори, від яких залежить успішне виконання операції, але їх неможна перевірити програмно. Такі фактори носять інформативний характер для майбутніх користувачів системи
Постумови	Опис змін, що відбулися в системі після виконання операції; опис змін стану об'єктів концептуальної моделі. В описі можна використовувати такі категорії постумов: ♦ створення/ видалення екземпляру; ♦ модифікація атрибута; ♦ формування/ розрив асоціації.

Опис передумов та постумов виконується в контексті концептуальної моделі. Екземпляри яких об'єктів утворюються? Об'єктів, що присутні на концептуальній моделі. Які асоціації можуть формуватися? Знов таки, асоціації з концептуальної моделі і т. д.

Дуже часто під час опису системних операцій виникає необхідність внесення змін до концептуальної моделі: нові поняття, нові атрибути, нові асоціації.

Рекомендації щодо складання опису системних операцій

Для того щоб скласти опис для кожного прецеденту, виконайте такі дії:

1. Визначте системні операції з діаграми послідовностей.
2. Складіть опис для кожної системної операції.
3. Розпочніть з опису розділу "Обов'язки", в якому неформально викладаються цілі операції.
4. Заповніть розділ "Постумови", в якому декларуються зміни в стані об'єктів концептуальної моделі.
5. Під час опису постумов використовуйте такі категорії:

♦ створення та видалення екземпляру класу;
♦ модифікація атрибута;
♦ формування або розрив асоціацій.

Постумови бажано описувати в декларативній формі з використанням дієслів минулого часу в завершеній формі пасивного залогу, щоб підкреслити факт зміни стану, а не спосіб його реалізації. Наприклад, краще сказати “Створений екземпляр класу …”, а не “Створюється екземпляр класу …”.

Не забувайте встановлювати відношення між існуючими і створеними сутностями концептуальної моделі шляхом формування асоціацій Наприклад, при створенні запису про студента не достатньо тільки створити екземпляр відповідного класу, а необхідно додати цей запис до списку. Тому однією з постумов буде “Створений екземпляр об’єкту … зв’язаний з об’єктом …”.

Найбільш типовою помилкою при формуванні постумов є невключення *формування асоціацій* в постумови операції.

6. Після заповнення розділу “Постумови” заповніть розділ “Передумови”, занесіть в нього всі умови, які необхідні для коректного виконання системної операції.