Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни
“СИСТЕМНИЙ АНАЛІЗ”
для студентів
напряму підготовки 6.050101
“Комп’ютерні науки”
всіх форм навчання

2014

Методичні вказівки до лабораторних робіт з дисципліни “Системний аналіз” для студентів напряму підготовки 6.050101 “Комп’ютерні науки” всіх форм навчання / Т.В. Юр, Ж.К. Камінська. – Запоріжжя: ЗНТУ, 2014. – 50 с.

Автори: Т. В. Юр, доцент, к.т.н.
Ж.К. Камінська, асистент

Рецензент: А.О. Олійник, доцент, к.т.н.

Відповідальний
за випуск: А.В.Притула, професор, к.т.н.

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 11
від “26” червня 2014 р.

ЗМІСТ

Вступ.....4
Лабораторна робота № 1 Системний опис об’єкту автоматизації.....5
Лабораторна робота № 2 Побудова діаграм потоків даних інформаційної системи.....10
Лабораторна робота № 3 Створення логічної моделі даних інформаційної системи.....20
Лабораторна робота № 4 Побудова діаграм прецедентів.....31
Лабораторна робота № 5 Побудова діаграм діяльності.....41
Література.....49
Додаток А Перелік тем лабораторних робіт.....50

ВСТУП

Дисципліна «Системний аналіз» розглядає методи та інструментальні засоби системного аналізу для вирішення формалізованих задач аналізу предметної області у сфері науки, техніки, освіти, бізнесу та економіки на предмет автоматизації інформаційних і технологічних процесів з використанням сучасних інформаційних та комп'ютерних технологій.

У сфері інформаційних технологій системний аналітик - це професійна роль, відповідальна за аналіз інтересів зацікавлених осіб на предмет можливості їх задоволення технічними властивостями створюваної інформаційної системи. Іншими словами, системний аналітик – це постановник завдань.

Продукт такого системного аналізу – організаційно-технічні рішення, оформлені у вигляді технічного завдання на систему чи програмне забезпечення або у вигляді специфікації вимог [1-4].

Завданням лабораторних робіт курсу «Системний аналіз» є освоєння системного підходу до проектування інформаційних систем. Курс присвячений освоєнню методології структурного системного аналізу і проектування.

Для цілей моделювання систем взагалі і структурного аналізу зокрема використовуються три групи засобів, що ілюструють функції, які система повинна виконувати, відносини між даними та залежну від часу поведінку системи (аспекти реального часу).

У методологіях структурного системного аналізу найбільш часто і ефективно вживаними є наступні методи:

– моделювання процесів із застосуванням DFD (Data Flow Diagrams) – діаграм потоків даних спільно зі словниками даних;

– моделювання даних із застосуванням ERD (Entity-Relationship Diagrams) – діаграм «сутність - зв'язок»;

– об'єктне моделювання із застосуванням UML (Unified Modeling Language) – уніфікованої мови моделювання.

Наведені засоби дають повний опис системи незалежно від того, чи є вона існуючою або розроблюється з початку. Таким чином будується логічна функціональна специфікація – докладний опис того, що повинна робити система, звільнений, наскільки це можливо, від розгляду шляхів реалізації. Це дає проектувальнику чітке уявлення про кінцеві результати, яких слід досягти.

Лабораторна робота № 1
Системний опис об’єкту автоматизації

1.1 Мета роботи

Вивчення та системне подання реальних об'єктів і процесів з метою автоматизації та комп'ютеризації вирішуваних в них завдань.

1.2 Основні теоретичні відомості

Систему, що реалізує функції збору, зберігання, обробки і передачі інформації, називають інформаційною системою (ІС). Найважливішими функціями цих системам є прогнозування, планування, облік, аналіз, контроль і регулювання. Технології виконання таких функцій називаються інформаційними технологіями.

Потреба у створенні ІС може бути обумовлена або необхідністю автоматизації чи модернізації існуючих інформаційних процесів, або необхідністю докорінної реорганізації діяльності підприємства (проведенні бізнес-реінжинірингу).

У процесі створення ІС дослідники прагнуть найбільш повного та об'єктивного подання об'єкту або процесу комп'ютеризації – опису його внутрішньої структури, що пояснює причинно-наслідкові закони функціонування і дозволяє передбачити, а значить, і керувати його поведінкою. Загальні закономірності функціонування та властивості систем є предметом вивчення системного аналізу.

Бізнес-логика в розробці ІС – це сукупність правил, принципів, залежностей поведінки об’єктів предметної області, яку підтримує система.

До задач системного аналізу в процесі створення ІС входять задачі декомпозиції, аналізу та синтезу.

Задача декомпозиції означає представлення системи у вигляді підсистем. Часто задачу декомпозиції розглядають як складову частину аналізу.

Задача аналізу полягає у визначенні властивостей системи і навколишнього середовища. Метою аналізу може бути визначення закону перетворення інформації, що задає поведінку системи. В останньому випадку мова йде про ***агрегацію*** (композицію) системи в один єдиний елемент.

Задача синтезу системи протилежне задачі аналізу. У задачі

синтезу на основі закону перетворення необхідно побудувати систему, яка буде реально виконувати це перетворення за певним алгоритмом. При цьому попередньо повинен бути визначений клас елементів, з яких будуватиметься система, що реалізує алгоритм функціонування.

Сукупність стадій і етапів, через які проходить ІС в своєму розвитку від моменту прийняття рішення про створення системи до моменту припинення функціонування системи, називається ***життєвий циклом ІС.***

Зміст життєвого циклу розробки ІС в різних підходах є однаковим і зводиться до проходження наступних стадій.

1. Планування та аналіз вимог (передпроектна стадія) - системний аналіз. Дослідження та аналіз існуючої інформаційної системи, визначення вимог до створюваної ІС, оформлення техніко-економічного обґрунтування і технічного завдання на розробку ІС.

2. Проектування (технічне проектування, логічне проектування). Розробка у відповідності зі сформульованими вимогами складу функцій, що автоматизуються (функціональна архітектура), і складу забезпечувальних підсистем (системна архітектура), оформлення технічного проекту ІС.

3. Реалізація (робоче проектування, фізичне проектування, програмування). Розробка та налаштування програм, наповнення баз даних, створення робочих інструкцій для персоналу, оформлення робочого проекту.

4. Впровадження (тестування, дослідна експлуатація). Комплексне налагодження підсистем ІС, навчання персоналу, поетапне впровадження ІС в експлуатацію по підрозділах економічного об'єкта, оформлення акту про приймально-здавальні випробування ІС.

5. Експлуатація ІС (супровід, модернізація). Збір рекламацій і статистики про функціонування ІС, виправлення помилок і недоробок, оформлення вимог до модернізації ІС та її виконання (повторення стадій 2-5).

Розглянемо основний зміст стадій і етапів життєвого циклу ІС, пов'язаних з системним аналізом.

І. Задача аналізу. До основних цілей процесу аналізу відноситься наступне:

– формулювання потреби в новій ІС або ідентифікація всіх недоліків існуючої ІС;

– обрання напрямку і визначення економічної доцільності проектування ІС.

Системний аналіз ІС починається з опису та аналізу функціонування об'єкту (системи), що розглядається, відповідно до вимог, які пред'являються до нього, та цілей проектування. У результаті цього етапу виявляються основні недоліки існуючої ІС, на основі яких формулюється потреба у вдосконаленні системи управління цим об'єктом і ставиться задача визначення економічно обґрунтованої необхідності автоматизації певних функцій управління, тобто створюється техніко-економічне обґрунтування проекту. Після визначення цієї потреби виникає проблема вибору напрямків вдосконалення об'єкту шляхом вибору програмно-технічних засобів. Результати оформляються у вигляді технічного завдання на проект, в якому відображаються технічні умови і вимоги до ІС, а також обмеження на ресурси проектування. Вимоги до ІС визначаються в термінах функцій, що реалізуються системою, і в термінах інформації, що надається нею.

ІІ. Задача синтезу. Цей процес передбачає:

– розробку функціональної архітектури ІС, яка відображає структуру функцій, що виконуються системою;

– розробку системної архітектури ІС, тобто склад забезпечувальних підсистем;

– виконати реалізацію проекту.

ІС складається з двох частин: функціональної, до якої належать елементи системи, що визначають її функціональні можливості, тобто що може система, її призначення, функції управління тощо, та забезпечувальної – програмно-апаратних засобів та інструментів, які реалізують функціональну частину. Частина забезпечення в свою чергу складається з підсистем технічного, математичного, програмного, інформаційного, лінгвістичного, правового, організаційно-методичного та ергономічного забезпечення.

Етап складання функціональної архітектури, що представляє собою сукупність функціональних підсистем і зв'язків між ними, є найбільш відповідальним з точки зору якості всієї наступної розробки.

Функціональні вимоги до системи визначають дії, які система повинна бути здатною виконати, зв’язок між входами та виходами в поведінці системи.

Побудова системної архітектури на основі функціональної архітектури передбачає виділення елементів і модулів інформаційного, технічного, програмного забезпечення та інших забезпечувальних підсистем, визначення інформаційних зв'язків та зв’язків управління між виділеними елементами і розробку технології обробки інформації.

Етап конструювання (фізичного проектування системи) включає розробку інструкцій користувачам і програм, створення інформаційного забезпечення, включаючи наповнення баз даних.

1.3 Завдання на лабораторну роботу

1.3.1. Обрати функціональну тобто предметну область для проведення системного аналізу.

Функціональна область може бути пов'язана з продажами, виробництвом, проведенням платежів, зберіганням на складах, наданням послуг тощо. Метою системного аналізу є проведення дослідження можливості впровадження або удосконалення інформаційної системи в обраній предметній області з метою комп'ютеризації задач, що вирішуються.

1.3.2. Виконати системний опис структури і процесу функціонування об'єкта комп'ютеризації: описати вхідні, вихідні дані, можливі стану системи, вказати підсистеми і зв'язки між ними, системні функції та системні цілі.

1.3.3. Обґрунтувати необхідність створення або удосконалення існуючої інформаційної системи.

1.4 Зміст звіту

1.4.1. Тема та мета роботи.

1.4.2. Результати роботи в довільній структурованої формі:

– розширений опис предметної області (системи), її цілі, функції і перелік задач, що вирішуються;

– призначення та цілі створення ІС;

– склад підсистем (модулів), опис їх призначення та функцій;

– відносини підсистем (модулів) між собою;

– вхідна і вихідна інформація для кожної підсистеми;

– опис основних сценаріїв робіт підсистем, опис основних операцій, які виконуються при зборі та обробці інформації;

– функціональні вимоги до ІС та перелік можливих запитів до неї;

– ілюстрація або модель, що характеризує об'єкт і процес комп'ютеризації: це може бути зовнішній вигляд об'єкта з вказівкою основних вузлів, функціонально-структурна схема, організаційна діаграма тощо.

1.4.3. Висновки, що містять обґрунтування необхідності створення або удосконалення інформаційної системи. Вказати ті очікувані позитивні результати, які будуть отримані в результаті

В цілому з матеріалу звіту має бути видно, що собою представляє об'єкт і процес комп'ютеризації, з якою метою функціонує об'єкт, які завдання при цьому вирішуються, які функції виконуються, чому і на якому рівні необхідна комп'ютеризація зазначених функцій. Таким чином, в роботі повинні бути відображені функціональні вимоги до інформаційної системи.

Лабораторна робота № 2
Побудова діаграм потоків даних інформаційної системи

2.1 Мета роботи

Ознайомитися з методологією побудови діаграм потоків даних DFD з метою моделювання процесів, що відбуваються в інформаційній системі.

2.2 Основні теоретичні відомості

Діаграми потоків даних (Data Flow Diagrams, DFD) – це один з основних інструментів графічного структурного аналізу і проектування ІС.

DFD є основним засобом моделювання функціональних вимог до системи, що проектується. З їх допомогою вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, пов'язаної потоками даних.

Головна мета таких діаграм – продемонструвати, як кожен процес перетворить свої вхідні дані у вихідні, і виявити відносини між цими процесами.

Модель DFD є *ієрархічною*. Кожен процес може бути підданий декомпозиції, тобто розбиттю на структурні складові, відносини між якими в тій же нотації можуть бути показані на окремій діаграмі. Коли досягнута необхідна глибина декомпозиції, процес нижнього рівня супроводжується міні-специфікацією, тобто текстовим описом.

2.2.1 Основні символи діаграми

Діаграма потоків даних містить наступні елементи:

– процеси, які перетворюють дані;

– потоки даних, що переносять дані;

– активні об'єкти, які виробляють і споживають дані;

– сховища даних, які пасивно зберігають дані.

Для зображення DFD використовуються дві різні нотації: Йодана (Yourdon) і Гейна-Сарсона (Gane-Sarson), що відрізняються синтаксисом.

Основні символи DF-діаграм зображені в таблиці 2.1.

Таблиця 2.1 – Основні символи DF-діаграм

Компонента	Нотація Йодана	Нотація Гейна-Сарсона
Поток даних (Data Flow)	Ім'я (пунктирна лінія – керуючий потік)	Ім'я
Процес (Process)	Ім'я id	id Ім'я
Сховище (Data Store)	Ім'я	id Ім'я
Зовнішня сутність (External Entity)	Ім'я	Ім'я

На діаграмах функціональні вимоги представляються за допомогою процесів і сховищ, пов'язаних потоками даних (рис.2.1).

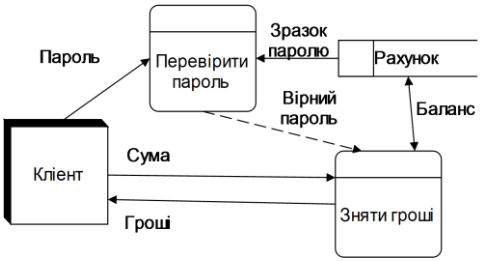


Рисунок 2.1 – Приклад діаграми потоків даних

Опишемо призначення компонентів DF-діаграм.

Потоки даних є механізмами, що використовуються для моделювання передачі інформації (або навіть фізичних компонентів) з однієї частини системи в іншу. Важливість цього об'єкта очевидна: він дає назву цілому інструменту. Потоки на діаграмах зазвичай зображуються іменованими стрілками, орієнтація яких вказує напрямок руху інформації.

Іноді інформація може рухатися в одному напрямку, а може оброблятися і повертатися назад до її джерела. Така ситуація може моделюватися або двома різними потоками, або одним – двонаправленим. Крім того, потоки даних на діаграмах можуть розщеплятися та об’єднуватися.

Призначення **процесу** полягає в продукуванні вихідних потоків з вхідних у відповідності до дії, що задається ім'ям процесу. Це ім'я має містити дієслово у невизначеній формі з наступним доповненням (наприклад, *обчислити максимальну висоту*). Крім того, кожен процес повинен мати унікальний номер для посилань на нього всередині діаграми. Цей номер може використовуватися спільно з номером діаграми для отримання унікального індексу процесу у всій моделі.

Сховище (накопичувач) даних дозволяє на певних ділянках визначати дані, які будуть зберігатися в пам'яті між процесами. Фактично сховище представляє "зрізи" потоків даних у часі. Ім'я сховища має ідентифікувати його вміст і має бути іменником.

Зовнішня сутність (або **термінатор**) представляє собою сутність поза контекстом системи, що є джерелом або приймачем системних даних. Її ім'я повинно містити іменник, наприклад, *склад товарів*. Передбачається, що об'єкти, представлені такими вузлами, не повинні брати участь ні в якій обробці.

Потоки управління не є стандартними але іноді включаються в розширену нотацію DFD. DFD показує всі шляхи обчислення значень але не показує, в якому порядку ці значення обчислюються. Рішення про порядок обчислень виробляються спеціальними функціями (предикатами). Такі функції визначають, чи буде виконаний той чи інший процес, але при цьому вони не передають процесу ніяких даних. Іноді буває корисно включати зазначені предикати у функціональну модель, щоб у ній були відображені умови виконання відповідного процесу. Функція, що приймає рішення про запуск процесу, будучи включеною в DF-діаграму, породжує потік управління, що зображується пунктирною стрілкою.

2.2.2 Контекстна діаграма і деталізація процесів

Декомпозиція DF-діаграми здійснюється на основі процесів: кожен процес може розкриватися та пояснюватися за допомогою DF-діаграми нижнього рівня.

Важливу роль в моделі грає спеціальний вид DFD – **контекстна діаграма** (context), що моделює систему найбільш загальним чином. Контекстна діаграма відображає інтерфейс системи із зовнішнім світом, а саме інформаційні потоки між системою і зовнішніми сутностями, з якими вона повинна бути пов'язана. Вона ідентифікує ці зовнішні сутності, а також, як правило, єдиний процес, що відображає

головну мету або природу системи. Контекстна діаграма встановлює межі системи, що аналізується. Кожен проект повинен мати одну єдину контекстну діаграму, при цьому її єдиний процес має номер 0.

DF-діаграма нульового рівня будується як декомпозиція нульового процесу, який присутній на контекстній діаграмі. Побудована діаграма нульового рівня містить множину процесів, які, в свою чергу, можуть бути розкриті на DF-діаграмах нижнього рівня. Таким чином будується ієрархія DF-діаграм з контекстною діаграмою в корені дерева. Процес декомпозиції продовжується допоки процеси зможуть бути ефективно описані за допомогою коротких мініспеціфікацій.

Мініспеціфікація (опис логіки) процесу повинна сформулювати його основні функції таким чином, щоб надалі фахівець, що буде виконувати реалізацію проекту, зміг розробити відповідну програму.

При такій побудові ієрархії DF-діаграм використовуються структуровані номера процесів. Так, наприклад, якщо ми деталізуємо процес номер 2 діаграми нульового рівня, розкриваючи його за допомогою DF-діаграми, що містить три процеси, то їх номери будуть мати наступний вигляд: 2.1, 2.2 і 2.3. При необхідності можна перейти на наступний рівень, тобто для процесу 2.2 отримаємо 2.2.1, 2.2.2 і т.д.

Проілюструємо контекстну діаграму на прикладі домашньої системи безпеки. У нас є зовнішні сутності: панель управління, датчики, звукова сигналізація, дисплей і телефонна лінія.

На контекстній діаграмі опишемо потоки даних, якими обмінюється наша система із зовнішніми об'єктами (рис. 2.2), а на діаграмі першого рівня деталізуємо систему і розпишемо процеси, що відбуваються в ній (рис. 2.3).

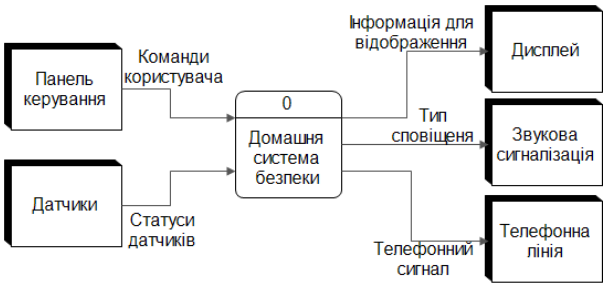


Рисунок 2.2 – Контекстна DF-діаграма домашньої системи безпеки

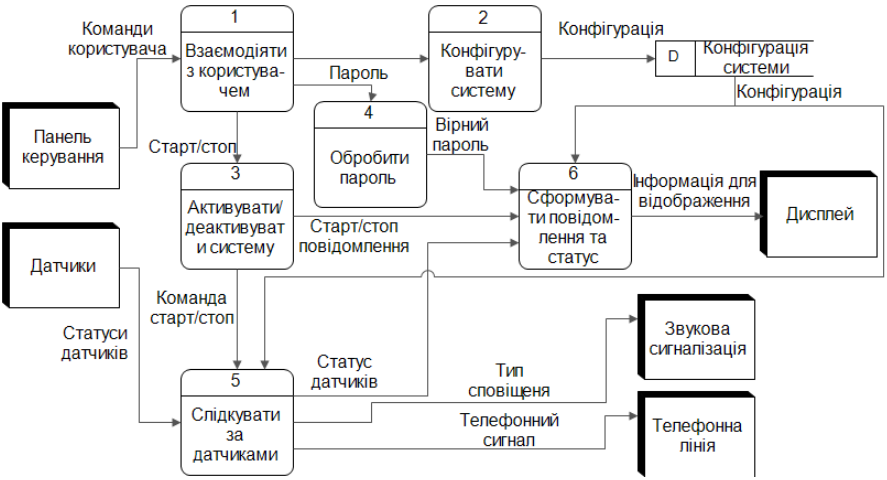


Рисунок 2.3 – Декомпозиція нульового рівня (0 DFD)

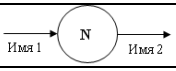
2.2.3 Декомпозиція даних та відповідні розширення діаграм потоків даних

Дані в системі часто є незалежними. Однак іноді необхідно мати справу з декількома незалежними даними одночасно.

Наприклад, в системі є потоки *яблука*, *апельсини* і *груші*. Ці потоки можуть бути згруповані за допомогою введення нового потоку *фрукти*. У свою чергу, потік *фрукти* сам може міститися в потоці *їжа* разом з потоками *овочі*, *м'ясо* та ін. Такі потоки, що поєднують кілька потоків, отримали назву групових.

Зворотна операція, розщеплення потоків на підпотоки, здійснюється з використанням групового вузла, що дозволяє розщепити потік на будь-яке число підпотоків. Можливий спосіб зображення групових вузлів наведено в табл. 2.2.

Таблиця 2.2 – Елементи, що використовуються при розщепленні

Компонента	Позначка
Груповий вузол (OR)	
Груповий вузол (AND)	
Вузол зміни імені	

Аналогічним чином здійснюється і декомпозиція потоків через кордони діаграм, що дозволяє спростити DFD більш низьких рівнів. Нехай є потік *фрукти*, що входить до деякого процесу. На діаграмі, що деталізує цей процес, потоку *фрукти* може не бути зовсім, але замість нього можуть бути потоки *яблука* і *апельсини*. У цьому випадку для цілей балансування в словнику має існувати визначення потоку *фрукти*, що складається з підпотоків *яблука* і *апельсини*.

Застосування цих операцій над даними дозволяє забезпечити структуризацію даних, збільшує наочність і читабельність діаграм.

2.2.4 Побудова моделі

Головна мета побудови ієрархії DF-діаграм полягає в тому, щоб зробити ясними і зрозумілими вимоги до проектованої системи на кожному рівні її деталізації.

Під час побудови ієрархії діаграм потоків даних слід дотримуватися наступних правил.

1. Правило балансування – означає, що при деталізації процесу деталізуюча діаграма буде містити тільки ті компоненти інформаційних потоків, які визначені на діаграмі верхнього рівня.
2. На кожній діаграмі може бути розміщено від 3 до 7 процесів.
3. Несуттєві на даному рівні деталі не повинні використовуватися.
4. Декомпозиція потоків даних проводиться одночасно з декомпозицією процесів.
5. Імена процесів і потоків даних повинні відображати їхню суть.
6. Функціонально ідентичні процеси слід визначати один раз на самому верхньому рівні де процес необхідний, а потім на більш низьких рівнях на цей процес посилатися.
7. Слід розділяти керуючі та вхідні потоки.
8. Правило нумерації полягає в тому, що при деталізації процесів повинна підтримуватися їх ієрархічна нумерація.

У відповідності з цими рекомендаціями процес побудови моделі розбивається на наведені нижче етапи.

1. Розбиття множини вимог до системи і організація їх в основні функціональні групи.
2. Ідентифікація зовнішніх об'єктів, з якими система повинна бути пов'язана.

3. Ідентифікація основних видів інформації, що циркулює між системою і зовнішніми об'єктами.

4. Попередня розробка контекстної діаграми, на якій основні функціональні групи представляються процесами, зовнішні об'єкти – зовнішніми сутностями, основні види інформації – потоками даних між процесами і зовнішніми сутностями.

5. Побудова контекстної діаграми шляхом об'єднання всіх процесів попередньої діаграми в один процес, а також групування потоків.

6. Формування DF-діаграми нульового рівня на базі процесів попередньої контекстної діаграми.

7. Декомпозиція кожного процесу поточної DF-діаграми за допомогою деталізуючої діаграми або специфікації процесу.

8. Додавання визначень нових потоків в словник даних при кожній їх появі на діаграмах.

9. Після побудови двох-трьох рівнів проведення ревізії з метою перевірки коректності та поліпшення зрозумілості моделі.

10. Побудова специфікації процесу (а не найпростішої діаграми) у разі, якщо деяку функцію складно або неможливо виразити комбінацією процесів.

2.2.5 Стратегія моделювання, заснована на подіях

Замість того, щоб будувати систему зверху-донизу чи навпаки знизу-доверху, сучасні стратегії структурного аналізу використовують розбиття системи на основі бізнес-подій та відгуків на ці події. Опишемо цю стратегію (рис. 2.4).

1. Будується контекстна діаграма системи.

2. Будується функціональна декомпозиція системи на логічні підсистеми та функції.

3. Для функцій створюється перелік подій, на які система повинна дати відгук.

4. Для кожної події створюється процес, який буде обробляти цю подію та будується проста DF-діаграма події, яка містить процес-обробник події та його входи, виходи та сховища даних.

5. Будується одна чи декілька системних DF-діаграм шляхом об’єднання діаграм подій.

6. Будуються деталізуючі діаграми для обробників подій.

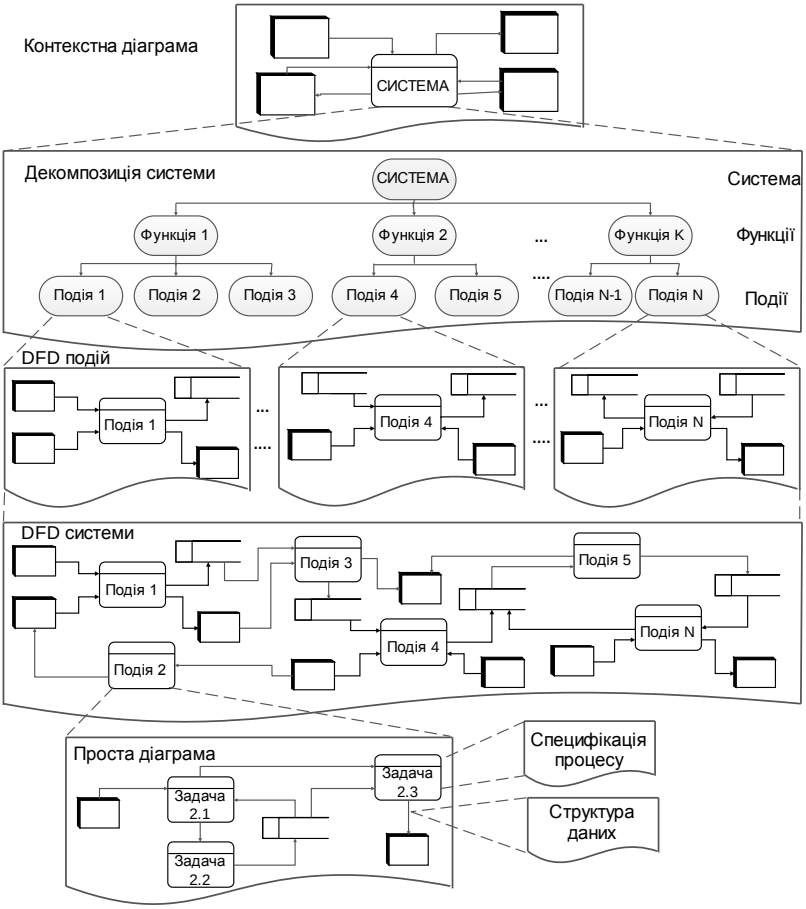


Рисунок 2.4 – DFD-моделювання на основі подій в системі

2.2.6 Методи надання специфікацій процесів

Специфікація процесу (СП) є кінцевою вершиною ієрархії DFD. Рішення про завершення деталізації процесу і використання мініспецифікації приймається аналітиком виходячи з таких критеріїв:

– наявності у процесу відносно невеликої кількості вхідних і вихідних потоків даних (2-3 потоки);

– можливості опису перетворення даних процесом у вигляді послідовного алгоритму;

– виконання процесом єдиної логічної функції перетворення

вхідної інформації у вихідну;

– опис процесу займає не більше однієї сторінки тексту.

Фактично СП – це алгоритми опису задач, що виконуються процесами: множина всіх СП є повною специфікацією системи. СП містять номер та/або ім'я процесу, списки вхідних і вихідних даних і тіло (опис): алгоритм або операції, що трансформують вхідні потоки даних у вихідні. Тіло процесу може бути задане за допомогою структурованої природної мови, псевдокоду, візуальних мов проектування або формальних комп'ютерних мов.

Іноді в СП задаються перед- та постумови виконання процесу. У передумові записуються об'єкти, значення яких повинні бути істинні перед початком виконання процесу, що забезпечує певні гарантії безпеки для користувача. Аналогічно, у разі наявності постумови гарантується, що значення всіх об'єктів, що входять до нього, будуть істинні по завершенню процесу.

Специфікації повинні задовольняти наступним вимогам:

– для кожного процесу нижнього рівня повинна існувати одна і тільки одна специфікація;

– специфікація повинна визначати спосіб перетворення вхідних потоків у вихідні;

– на даному етапі немає необхідності визначати метод реалізації цього перетворення;

– набір конструкцій для побудови специфікації повинен бути простим і стандартним.

2.2.7 Словник даних

Діаграми потоків даних забезпечують зручний опис функціонування компонент системи, але не надають аналітику засобів опису деталей цих компонент, а саме, яка інформація перетворюється процесами і як вона перетворюється. Для вирішення першої з наведених задач призначені текстові засоби моделювання, що служать для опису структури перетворюваної інформації та отримали назву словників даних.

Словник даних – це певним чином організований список всіх елементів даних системи з їх точними визначеннями і описами, що дає можливість різним категоріям користувачів (від системного аналітика до програміста) мати спільне розуміння всіх вхідних і вихідних

потоків і компонентів сховищ. Кожен елемент словника містить інформацію про його призначення, зв'язки з іншими елементами, походження, застосуванні і формат представлення.

2.2.8 Представлення даних у формі Бекуса–Наура

Описи компонент даних в потоках даних і в сховищах зберігаються в словнику даних у формі Бекуса-Наура (БНФ). Її синтаксис має наступний вигляд:

Елемент даних = <БНФ-вираз>,

де <БНФ-вираз> – вираз в формі Бекуса-Наура, що допускає наступні операції співвідношень:

= – означає "композиція із",

+ – означає *"І",*

[|]– означає *"АБО",*

( ) – означає, що компонент в скобках не обов'язковий,

{ } – означає літерацію компонента в скобках,

" " – означає літерал.

Наприклад:

«Запис студента» = «ІД студента» + «ПІП» + «Адреса» + «Дата народження» + «Стать» + «Телефон» + («Мобільний телефон»);

«ПІП» = «Прізвище» + «Ім'я» + «По батькові»;

«Стать» = {«ч» | «ж»}.

2.3 Завдання на лабораторну роботу

2.3.1. Побудувати для інформаційної системи, що розробляється, серію діаграм потоків даних (контекстна діаграма, діаграма нульового рівня, діаграми першого рівня і при можливості одну діаграму другого рівня, всього не менше 5 діаграм).

2.3.2. Для одного процесу діаграми нижнього рівня скласти міні-специфікацію: блок-схему, що описує логіку процесу, дерево рішень або текстовий опис.

2.3.3. Скласти словник даних для діаграм.

2.4 Зміст звіту

2.4.1. Тема і мета роботи.

2.4.2. Результати роботи.

2.4.3. Висновки по виконаній роботі.

Лабораторна робота № 3
Створення логічної моделі даних інформаційної системи

3.1 Мета роботи

Ознайомитися з технологією побудови логічної моделі даних інформаційної системи.

3.2 Основні теоретичні відомості

Діаграми «сутність-зв'язок» (entity-relationship diagram, ERD) призначені для опису концептуальної схеми предметної області, розробки моделей даних. Вони забезпечують стандартний спосіб визначення даних і відносин між ними. Фактично за допомогою ERD здійснюється деталізація сховищ даних системи, що проектується, а також документуються сутності системи і способи їх взаємодії, включаючи ідентифікацію об'єктів , важливих для предметної області (сутностей), властивостей цих об'єктів (атрибутів) та їх відносин з іншими об'єктами (зв'язків).

ER-діаграма дозволяє розглянути систему цілком і з'ясувати вимоги, необхідні для її розробки, що стосуються зберігання інформації. Під час проектування баз даних відбувається перетворення ER-моделі в конкретну схему бази даних на основі обраної моделі даних (реляційної, об'єктної, мережевої або ін).

ER-діаграми можна розділити на окремі частини, що відповідають окремим завданням, які розв'язуються системою, що проектується. Це дозволяє розглядати систему з точки зору функціональних можливостей і робить процес проектування керованим.

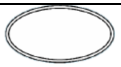
3.2.1 Нотація Пітера Чена

Ця нотація була введена Ченом (Chen) і отримала подальший розвиток у роботах Баркера (Barker). Нотація Чена надає багатий набір засобів моделювання даних, включаючи власне ERD, а також діаграми атрибутів і діаграми декомпозиції.

Розглянемо сутності, атрибути і зв'язки в нотації Чена.

Символи ERD в нотації Чена наведено в таблиці 3.1.

Таблиця 3.1 – Символи ERD в нотації Чена

Символ	Значення	Символ	Значення
	Сутність		Асоціативна сутність
	Слабка сутність		Атрибут
	Зв'язок		Багатозначний атрибут
	Зв'язок, що ідентифікує		Похідний атрибут

Сутність представляє собою множину екземплярів реальних або абстрактних об'єктів (людей, подій, станів, ідей, предметів тощо), що володіють спільними ***атрибутами*** або ***характеристиками***. Будь-який об'єкт системи може бути представлений тільки однією сутністю, яка повинна бути унікально ідентифікована.

Ім'я сутності – це ім'я типу, а не конкретного екземпляру даного типу. Екземпляром сутності називається конкретний представник даної сутності. Ім’я сутності виражається іменником. Ім’я сутності має бути унікальним в рамках однієї моделі.

Деталізація сутності здійснюється з використанням діаграм атрибутів (рис. 3.1). Кожна сутність має один або кілька атрибутів (їх називають ***ключовими***), які однозначно ідентифікують кожен екземпляр сутності. Для ідентифікації ключового атрибута використовується підкреслення імені атрибута.

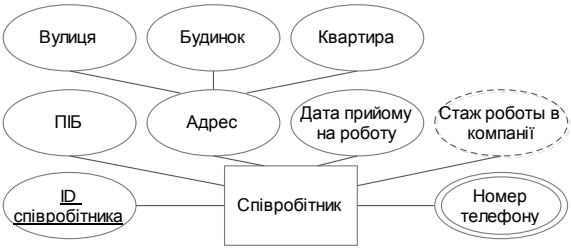


Рисунок 3.1 – Діаграма атрибутів

Зв’язок показує відношення між двома і більше сутностями. Іменування зв’язків здійснюється за допомогою дієслова (*Має*, *Визначає*, *Володіє* тощо).

Іншими словами, сутності представляють собою базові типи інформації, що зберігається в базі даних, а зв’язки показують, як ці типи даних взаємопов'язані один з одним.

Значення зв'язку характеризує його тип і, як правило, вибирається з наступної множини: *{«0 або 1»,«0 або більше», «1», «1 або більше»,«p:q» (діапазон)}*.

Будь-який зв'язок двох сутностей характеризується мінімальним і максимальним значеннями кількості сутностей, що взаємодіють, а саме ординалітетом і кардиналітетом відповідно. Ця пара значень визначає ***тип*** цього зв’язку. Практика показала, що у більшості випадків достатньо використати наступні типи.

1. *Один-до-одного.* Зв'язок цього типу використовується, як правило, на верхніх рівнях ієрархії моделі даних, а на нижніх рівнях зустрічаються порівняно рідко. Зв'язок один-до-одного найчастіше свідчить про те, що насправді ми маємо всього одну сутність, неправильно розділену на дві.

2. *Один-до-багатьох.* Зв'язок даного типу використовується найчастіше. Сутність з боку "один" називається *батьківською,* сутність з боку "багато" – *дочірньою.*

3. *Багато-до-багатьох.* Зв'язок даного типу зазвичай використовується на ранніх етапах проектування з метою прояснення ситуації і називається неспецифічним. Надалі кожен з таких зв’язків має бути перетворений на комбінацію зв’язків типів 1 і 2 (можливо, з додаванням допоміжних сутностей і введенням нових зв’язків).

Значення зв'язку відображається цифрою над лінією зв'язку з боку сутності, найчастіше – тільки кардиналітет.



Рисунок 3.2 – Приклад бінарного зв’язку типу багато-до-одного

Значення зв'язку «1» і «багато» також можуть включати значення 0. Це означає, що сутність може не перебувати в даному зв’язку. Кожен зв'язок може мати одну з двох *модальностей:*

– модальність "може" означає, що екземпляр однієї сутності може бути пов'язаний з одним або декількома екземплярами іншої сутності, а може бути і не пов'язаний ні з одним екземпляром (позначається лінією);

– модальність "повинен " означає, що екземпляр однієї сутності зобов'язаний бути пов'язаним не менше ніж з одним екземпляром іншої сутності (позначається подвійною лінією).

Зв'язок може мати різну модальність з різних кінців.



Рисунок 3.3 – Кожний кредит повинен мати клієнта

Ступінь зв'язку показує кількість сутностей, що входять до нього. Зв'язки бувають *унарні*, *бінарні*; *тернарні* і *n-арні*.



Рисунок 3.4 – Приклад унарного зв'язку

Зв'язки можуть супроводжуватися текстовими мітками, що описують ролі сутностей у відношенні. Особливе значення мають рольові імена для відношень на одній сутності. У цьому випадку немає інших засобів крім рольових імен для ідентифікації різних входжень однієї сутності у відношення.

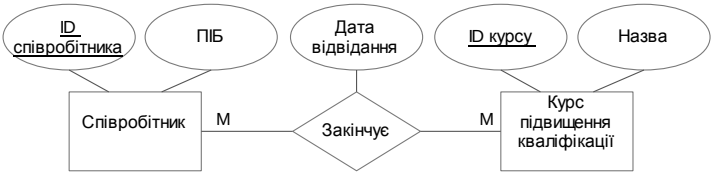


Рисунок.3.5 – Приклад бінарного зв'язку з атрибутом

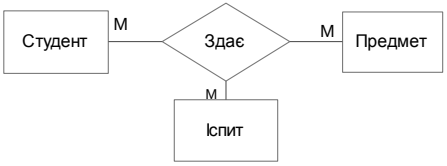


Рисунок 3.6 – Приклад тернарного зв'язку

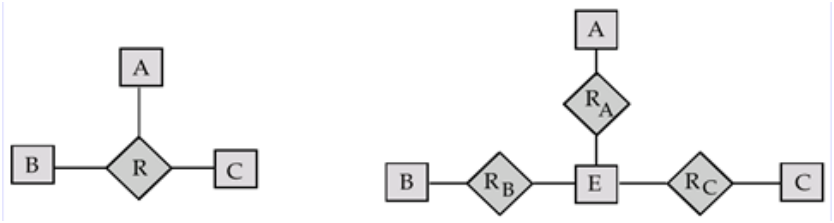


Рисунок 3.7 – Перетворення тернарного зв’язку на три бінарних

3.2.2 Нотація Гордона Евереста

Ця нотація була запропонована Гордоном Еверестом (Gordon Everest) під назвою Inverted Arrow (перевернута стрілка), однак зараз її частіше називаються Crow's Foot (вороняча лапка) або Fork (вилка).

Згідно даної нотації, сутність зображується у вигляді прямокутника, що містить її ім'я, яке виражається іменником.

Зв'язок зображується лінією, яка пов'язує дві сутності, що беруть участь у відношенні. Значення кінця зв’язку відображається графічно: множинність зв'язку зображується у вигляді «вилки» на кінці зв'язку,необов'язковість зв'язку позначається кружком на кінці зв'язку (табл. 3.2).

Таблиця 3.2 – Символи ERD в нотації Гордона Евереста

Значення зв’язку	**Ординалітет**	**Кардиналітет**	**Нотація**
«1»	1	1	сутність або сутність
«0 або 1»	0	1	сутність
«1 або більше»	1	багато(>1)	сутність
«0 або більше»	0	багато (>1)	сутність
«багато, більше 1»	>1	багато (>1)	сутність

Іменування зв’язку зазвичай виражається одним дієсловом дійсного способу теперішнього часу: «*Має*», «*Належить*» тощо, або дієсловом з пояснюючими словами: «*Включає в себе*». Найменування може бути одне для всього зв'язку або два для кожного з кінців зв'язку. У другому випадку, назва лівого кінця зв'язку вказується над лінією зв'язку, а правого – під лінією. Кожна з назв розташовуються поруч з сутністю, до якої вона відноситься.

Атрибути сутності записуються усередині прямокутника, що зображує сутність, і виражаються іменником в однині (можливо, з уточнюючими словами). Серед атрибутів виділяється ключ сутності – ненадлишковий набір атрибутів, значення яких в сукупності є унікальними для кожного екземпляра сутності.

На рис. 3.8 наведено приклад ER-діаграми у нотації Евереста.

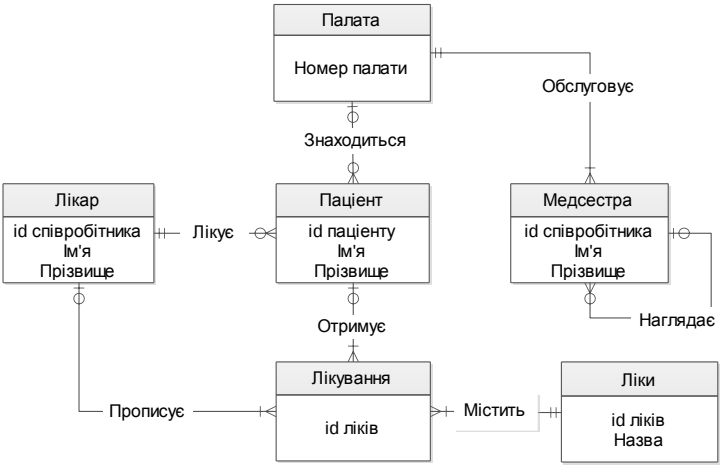


Рисунок 3.8 – ER-діаграма у нотації Гордона Евереста

3.2.3 Поняття сильної, слабкої і асоційованої сутностей

Незалежна (сильна) сутність представляє незалежні дані, які завжди присутні в системі. При цьому зв’язки з іншими сутностями у неї можуть як існувати, так і бути відсутніми.

У свою чергу, слабка сутність представляє дані, що залежать від інших сутностей в системі. Тому вона повинна завжди мати зв’язки з іншими сутностями (*ідентифікуючі зв’язки)*. Тип зв'язку, який з'єднує дані типи сутностей, називається *слабким типом.*

Слабкі сутності можуть самі по собі не мати ключових атрибутів, оскільки вони залежать від сильних сутностей (які мають ключові атрибути). У слабких сутностей зовнішній ключ є частиною їх ключа.

На діаграмах в нотації Чена слабкі сутності та їх зв'язки позначаються подвійними лініями.

На діаграмах в нотації Евереста слабка сутність позначається подвійною лінією, а слабкий зв'язок - пунктирною лінією.

Наприклад, сутність *Підлеглий* є слабкою по відношенню до сутності *Співробітник*: якщо буде видалений запис, відповідний деякому співробітникові, що має підлеглих, то відомості про підпорядкування також повинні бути видалені (рис. 3.9).

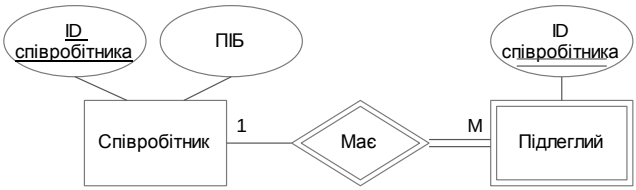


Рисунок 3.9 – Приклад слабкої сутності

Слабкі сутності ідеально підходять для представлення багатозначних атрибутів сутностей.

Асоційована сутність (асоціація) – це зв'язок виду "багато -до-багатьох " між двома або більше сутностями. Асоціації розглядаються як повноправні суті : вони можуть брати участь в інших асоціаціях точно так само, як незалежні сутності; можуть мати властивості, тобто мати не тільки набір ключових атрибутів , необхідних для вказівки зв'язків , але і будь-яке число інших атрибутів , що характеризують зв'язок (рис. 3.10).

Асоційована сутність частіше зустрічається в нотації Евересту.

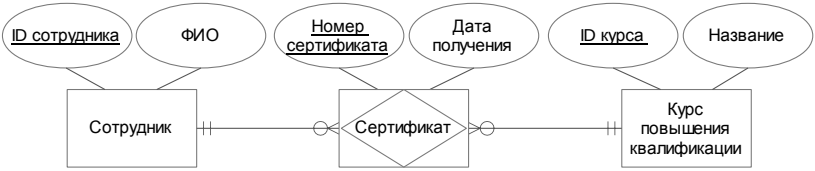


Рисунок 3.10 – Асоційована сутність

3.2.4 Перетворення неспецифічних зв’язків

Кожний неспецифічний зв'язок перетворюється на два специфічних зв’язка з введенням нових (простих або асоціативних) сутностей.

Розглянемо приклади. *Студент* може вивчати багато *Предметів*, а *Предмет* може вивчатися багатьма *Студентами*. Ми не можемо визначити, який *Студент* який *Предмет* вивчає. Отже, у нас є неспецифічний зв'язок, який можна розщепити на два специфічних зв’язка шляхом введення асоціативної сутності *Вивчення_Предмета*. Кожен екземпляр введеної сутності пов'язаний з одним *Студентом* і одним *Предметом* (рис.3.11).

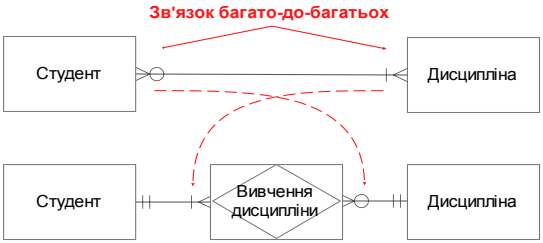


Рисунок 3.11 – Приклад №1 розв’язання неспецифічних зв’язків

На рисунках 3.12 та 3.13 наведено ще два приклади розв’язування неспецифічних зв’язків.



Рисунок 3.12 – Приклад №2 розв’язання неспецифічних зв’язків

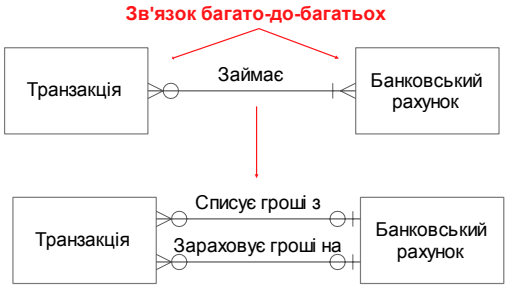


Рисунок 3.13 – Приклад №3 розв’язання неспецифічних зв’язків

3.2.5 Категоризація сутностей

Сутність може бути розділена і представлена у вигляді двох або більше сутностей – категорій, кожна з яких має спільні атрибути та/або зв’язки, які визначаються один єдиний раз на верхньому рівні і успадковуються на нижньому, а також свої власні атрибути та/або відносини. Сутність, що розщеплюється на категорії (***підтипи***), отримала назву ***загальної сутності (або супертипу).*** Відзначимо, що на проміжних рівнях декомпозиції одна і та ж сутність може бути як загальною сутністю, так і сутністю-категорією.

Атрибут загальної сутності, значення якого визначає цільовий підтип, називається дискримінатором.

Для демонстрації декомпозиції сутності на категорії використовуються діаграми категоризації. Така діаграма містить загальну сутність, дві і більше сутності-категорії і спеціальний вузол, який описує способи декомпозиції сутностей (рис. 3.12).

Існують наступні типи декомпозиції:

а) категоризація може бути повною та частковою, тобто:

1) екземпляр супертипу повинен також бути членом підтипу (total specialization);

2) екземпляр супертипу може належати, а може не належати жодному з підтипів (partial specialization);

б) екземпляр супертипу може належати лише одному підтипу (disjoint) чи відразу декільком (overlapping).

На рисунках 3.14-3.16 наведено приклади категоризації сутностей.

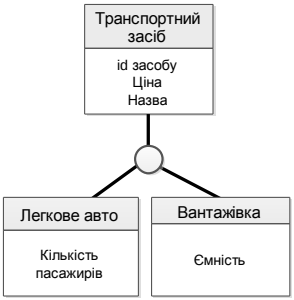


Рисунок 3.14 – Приклад часткової категоризації

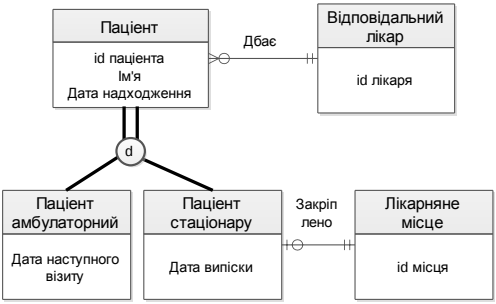


Рисунок 3.15 – Приклад повної категоризації №1

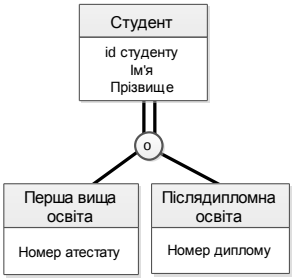


Рисунок 3.16 – Приклад повної категоризації №2

3.2.6 Побудова моделі

Розробка ER-моделі предметної області включає наступні етапи.

Етап 1. Проводиться ідентифікація сутностей, їх первинних і альтернативних ключів, при необхідності основних атрибутів. Цей етап є визначальним при побудові моделі, його вихідною інформацією

служить вміст сховищ даних, що визначається його вхідними та вихідними потоками даних.

Якщо отримана схема володіє надмірністю, то відбувається її спрощення схеми шляхом нормалізації.

Етап 2 полягає в ідентифікації відношень між. Даний етап служить для виявлення і визначення зв'язків між сутностями, а також для ідентифікації типів зв'язків. Деякі зв'язки на цьому етапі можуть бути неспецифічними (багато-до-багатьох). Такі зв'язки потребують подальшої деталізації на етапі 3.

Етап 3 полягає у розв'язанні неспецифічних зв'язків.

На етапі 4 проводиться категоризація сутностей і визначення всіх необхідних атрибутів сутностей.

3.3 Завдання на лабораторну роботу

3.3.1 Побудувати серію ER-діаграм для всієї інформаційної системи в цілому і для окремих сценаріїв робіт, що відображають логіку і взаємовідношення підсистем.

3.4 Зміст звіту

3.4.1 Тема і мета роботи.

3.4.2 Результати роботи.

3.4.3 Висновки за результатами роботи.

Лабораторна робота № 4
Побудова діаграм прецедентів

4.1 Мета роботи

Ознайомитися з методологією моделювання прецедентів на основі мови UML.

4.2 Основні теоретичні відомості

UML (Universal Modeling Language) – універсальна мова моделювання, яка була розроблена компанією Rational Software з метою створення найбільш оптимальної та універсальної мови для опису як предметної області, так і конкретної задачі в програмуванні.

UML – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML є мовою широкого профілю і відкритим стандартом, що використовує графічні позначення для створення абстрактної моделі системи, названої UML-моделлю. UML був створений для визначення, візуалізації, проектування та документування програмних систем.

Візуальне моделювання в UML можна представити у вигляді процесу спуску по рівнях від найбільш загальної і абстрактної концептуальної моделі системи до логічної, а потім і до фізичної моделі відповідної системи. Будь-яка задач, таким чином, моделюється за допомогою деякого набору ієрархічних діаграм, кожна з яких представляє собою деяку проекцію системи.

Діаграма (Diagram) – це графічне представлення багатьох елементів. Найчастіше вона зображується у вигляді зв'язного графа з вершинами (сутностями) і ребрами (відношеннями).

У UML визначені діаграми двох категорій: структурні і поведінкові. Серед діаграм поведінки виділяють підкатегорію діаграм взаємодії.

З точки зору системного аналізу цікаві наступні типи діаграм:

– ***діаграма прецедентів*** (Use case diagram, діаграма варіантів використання) – діаграма поведінки, на якій показана множина прецедентів та акторів, а також відносини між ними;

– ***діаграма діяльності*** (Activity diagram) – діаграма поведінки, на якій показаний автомат і підкреслені переходи потоку управління від однієї діяльності до іншої;

– ***діаграма послідовностей*** (Sequence diagram) – діаграма поведінки, на якій показано взаємодію і підкреслена часова послідовність подій;

– ***діаграма станів*** (Statechart diagram) – діаграма поведінки, на якій показаний автомат і підкреслено поведінку об'єктів з точки зору порядку отримання подій;

– ***діаграма класів*** (Class diagram) – структурна діаграма, на якій показано множину класів, інтерфейсів, кооперацій і відносини між ними.

4.2.1 Діаграма прецедентів

Діаграми прецедентів застосовуються для концептуального моделювання системи з точки зору зовнішнього спостерігача.

Прецеденти – це технологія визначення функціональних вимог до системи. Робота прецедентів полягає в описі типових взаємодій між користувачами системи і самою системою та наданні опису процесу її функціонування.

Прецедент – це можливість (тобто частина функціональності) системи, що моделюється, завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат.

При моделюванні системи за допомогою діаграми прецедентів системний аналітик прагне:

– чітко відокремити систему від її оточення;

– визначити дійових осіб (акторів), їх взаємодію з системою і очікуваний функціонал системи;

– визначити в глосарії предметної області поняття, що відносяться до детального опису функціонала системи (тобто, прецедентів).

Подібний вид діяльності зазвичай виконується в такій послідовності:

– визначення дійових осіб;

– визначення прецедентів;

– складання опису кожного прецеденту;

– опис моделі прецедентів в цілому (цей етап включає в себе створення словника предметної області).

Розглянемо основні елементи діаграми прецедентів.

Рамки системи (system boundary) – прямокутник з назвою

системи або підсистеми у верхній частині і прецедентами всередині, позначає межі автоматизації або комп'ютеризації. Часто може бути опущений без втрати корисної інформації.

Суб'єкт (actor) – це будь-яка сутність, що взаємодіє з системою ззовні або множина логічно пов'язаних ролей, що виконуються при взаємодії з прецедентами. Стандартним графічним позначенням суб'єкта на діаграмах є фігурка "чоловічка", під якою записується конкретне ім'я суб'єкту. Суб'єктом може бути не тільки людина, але й технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на систему, що моделюється. Актори не можуть бути пов'язані один з одним. Єдине допустиме відношення між акторами – це генералізація (успадкування).

Прецеденти (use case) – це опис множини дій (включаючи їх варіанти), які виконуються системою для того, щоб актор отримав результат, що має для нього певне значення. При цьому нічого не говориться про те, яким чином буде реалізовано ця дія системи, що є однією з найважливіших особливостей розробки прецедентів. Стандартним графічним позначенням прецеденту на діаграмах є еліпс, всередині якого міститься коротка назва прецеденту чи ім'я у формі дієслова з пояснювальними словами. Ім'я прецеденту – це завжди опис з точки зору актора, опис послуг, що надаються системою користувачеві.

Ім'я прецеденту пов'язано з безперервним (атомарним) *сценарієм* – конкретною послідовністю дій, що ілюструє поведінку. Сценарій може бути приведений на діаграмі прецедентів у вигляді UML- коментаря. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

Сутність концепції прецедентів припускає декілька важливих пунктів:

– прецедент представляє собою **завершений** фрагмент функціональних можливостей, який включає основний потік логіки управління, його будь-які варіації (підпотоки) і виняткові умови (альтернативні потоки);

– прецедент представляє собою фрагмент **зовні спостережуваних функцій** (відмінних від внутрішніх функцій);

– прецедент представляє собою **ортогональний** фрагмент функціональних можливостей, тобто це означає, що прецеденти

можуть при виконанні спільно використовувати об'єкти, але виконання кожного прецеденту незалежно від інших прецедентів і не може бути перервано;

– прецедент представляє собою фрагмент функціональних можливостей, **ініційований суб'єктом**; будучи ініційованим, прецедент може взаємодіяти з іншими суб'єктами, при цьому можливо, що суб'єкт виявиться тільки на приймаючому кінці прецеденту, опосередковано ініційованого іншим суб'єктом;

– прецедент являє собою фрагмент функціональних можливостей, який надає суб'єкту **відчутний корисний результат** (і цей результат досягається в межах одного прецеденту).

У прикладі на рисунку 4.1 пасажир може купити в сервісній касі квиток на деякий вид транспорту. Купівля квитка – це назва сценарію, за яким актор (пасажир) може взаємодіяти з системою (касою). Зауважте, це не опис сценарію, а саме назва – вона говорить нам, що робить актор у процесі взаємодії, але не говорить, як саме.

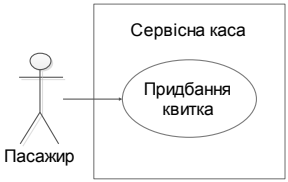


Рисунок 4.1 – Приклад діаграми прецедентів

Між суб'єктами і прецедентами – основними компонентами діаграми прецедентів – можуть існувати різні відношення, які описують взаємодію екземплярів одних суб'єктів і прецедентів з екземплярами інших суб'єктів і прецедентів. У мові UML є декілька стандартних видів відношень між суб'єктами та прецедентами:

Відношення асоціації (association) – визначає наявність каналу зв'язку між екземплярами суб'єкта і прецеденту (або між екземплярами двох суб'єктів). Позначається суцільною лінією, можливо наявність стрілки і вказівка потужності зв'язку.

Відношення розширення (extend) – визначає взаємозв'язок екземплярів окремого прецеденту з більш загальним прецедентом, властивості якого визначаються на основі способу спільного об'єднання даних екземплярів. Позначається пунктирною лінією зі стрілкою, спрямованою від того прецеденту, який є розширенням для

вихідного прецеденту, і позначається ключовим словом "extend" ("розширює").

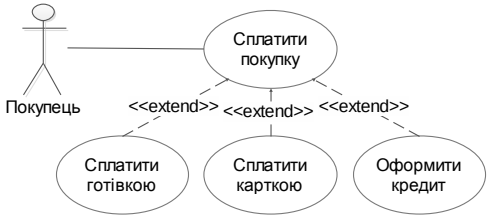


Рисунок 4.2 – Приклад відношення розширення

Відношення включення (include) – вказує, що деяка задана поведінка для одного прецеденту включає (використовує) як складовий компонент поведінку іншого прецеденту. Це відношення є спрямованим бінарним відношенням в тому сенсі, що пара екземплярів прецедентів завжди впорядкована відносно включення. Позначається пунктирною лінією зі стрілкою, спрямованої від базового прецеденту до такого, що включається, і позначається ключовим словом "include" ("включає").

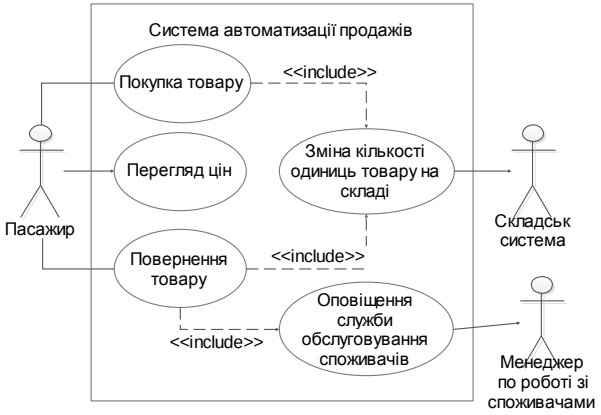


Рисунок 4.3 – Приклад відношення включення

Відношення узагальнення (generalization) – служить для указання того факту, що деякий прецедент А може бути узагальнений до прецеденту В. У цьому випадку прецедент А буде спеціалізацією прецеденту В. При цьому В називається предком або батьком по відношенню до А, а прецедент А – нащадком по відношенню до

прецеденту В. Слід підкреслити, що нащадок успадковує всі властивості і поведінку свого батька, а також може бути доповнений новими властивостями і особливостями поведінки. Графічно це відношення позначається суцільною лінією зі стрілкою у формі незафарбованого трикутника, яка вказує на батьківський прецедент.

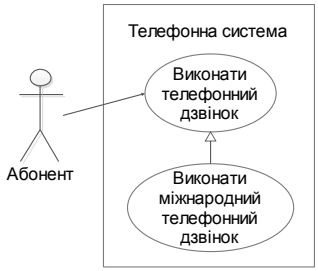


Рисунок 4.4 – Приклад відношення узагальнення

Подальший розвиток моделі поведінки системи передбачає специфікацію прецедентів. Для цього традиційно використовують два способи. Перший - опис за допомогою текстового документа. Такий документ описує, що повинна робити система, коли суб'єкт ініціював прецедент. Типовий опис містить такі розділи:

– короткий опис;

– суб'єкти, що беруть участь;

– передумови, необхідні для ініціювання прецеденту;

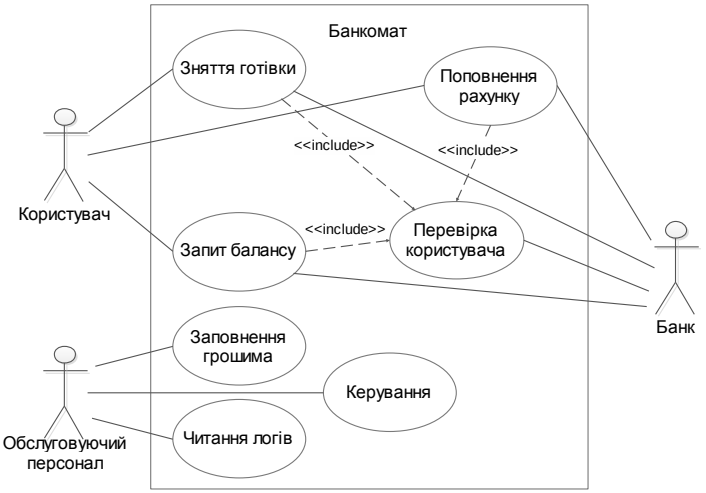
– основний потік подій і при необхідності підпотоків і альтернативний потік;

– післяумови, що визначають стан системи, після досягнення якого прецедент завершується.

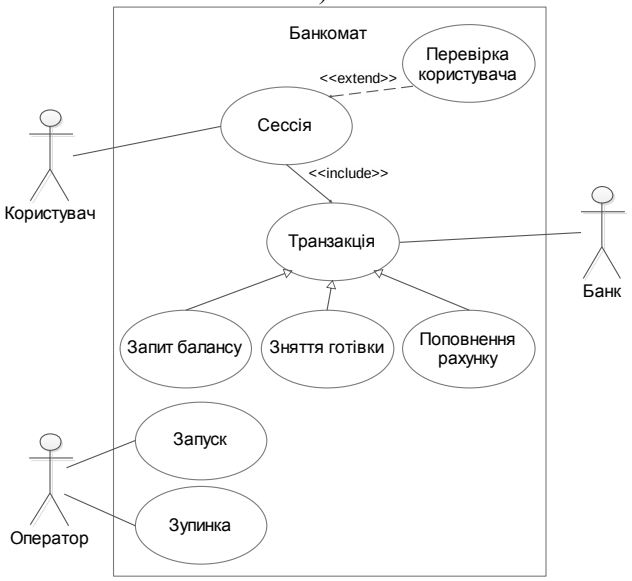
4.2.2 Приклад діаграми прецедентів для ІС «Банкомат»

Наведемо можливі приклади діаграми прецедентів для системи банкомату (рис.4.5а та рис.4.5б). Зазначимо, що одна і та ж інформаційна система може бути змодельована за допомогою діаграм прецедентів різними способами.

У табл. 4.1 наведено приклад описової специфікації прецеденту «Зняття готівки» для ІС банкомату.



а)



б)

Рисунок 4.5 – Діаграми прецедентів для системи банкомату

Таблиця 4.1 – Специфікація прецеденту "Зняття готівки"

Розділ	**Опис**
1. Короткий опис (Brief description)	Цей прецедент описує, як користувач використовує банкомат для зняття готівки зі свого рахунку (банківської картки)
2. Суб'єкти (Actors)	1. Користувач 2. Банк
3. Предумова (precondition)	1. Присутнє активне мережеве з'єднання між банкоматом і банком. 2. У банкоматі присутні грошові купюри.
4. Основний потік (Basic Flow of Events)	1. Прецедент починається, коли Користувач вставляє в банкомат свою Банківську картку. 2. Виконується прецедент «Перевірка користувача». 3. Банкомат відображає різні альтернативи, які доступні для виконання користувачем. У цьому прецеденті Користувач завжди вибирає «Зняття готівки». 4. Банкомат запитує суму для зняття. 5. Користувач вводить суму. 6. Номер картки, ПІН-код і сума відправляються до Банку у вигляді транзакції. Банк повертає відповідь, чи була транзакція успішною. 7. Банкомат видає гроші. 8. Картка повертається клієнтові. 9. Друкується чек. 10. Прецедент успішно завершений.
5. Альтернативний потік (Alternative Flows)	2а. Невірний користувач: прецедент «Перевірка користувача" не завершується успішно. 2а-1. Прецедент завершується з помилкою. 5а. Неправильна сума: користувач вводить суму, яка не може бути видана банкоматом. 5а-1. Банкомат відображає повідомлення про те, що сума повинна бути кратна мінімальної купюрі в наявності, і просить ввести суму повторно. 5а-2. Прецедент поновлюється на кроці 4. 5б. Запитана сума перевищує ліміт: Користувач вводить суму, що перевищує ліміт для видачі за раз.

Продовження таблиці 4.1

Розділ	Опис
	5б-1. Банкомат відображає попереджувальне повідомлення і просить ввести суму повторно. 5б-2. Прецедент поновлюється на кроці 4. 5в. Запрошення сума перевищує денний ліміт: Користувач вводить суму , яка перевищує ліміт для видачі готівки в день (його розмір визначається Банком і залежить від рахунку Користувача). 5в-1. Банкомат відображає попереджувальне повідомлення і просить користувача ввести суму повторно. 5в-2. Прецедент поновлюється на кроці 4. 5г. Недостатньо готівки: Користувач вводить суму, яка перевищує суму готівки, доступну в Банкомат. 5г-1. Банкомат відображає попереджувальне повідомлення і просить користувача ввести суму повторно. 5г-2. Прецедент поновлюється на кроці 4. 6а. Немає відповіді від Банку: відповідь від Банку не надходить протягом 3 секунд. 6а-1. Банкомат робить повторні спроби зв'язатися з Банком, аж до 3 разів. 6а-2. Якщо відповіді немає, то Банкомат відображає повідомлення «Зв'язок з банком відсутній - повторіть спробу пізніше». 6а-3. Банкомат повертає картку Користувачеві. 6а-4. Банкомат показує, що він не працює. 6а-5. Прецедент завершується з помилкою. 7а. Гроші не були витягнуті: гроші не були витягнуті Користувачем протягом 15 сек, тоді 7а-1. Банкомат видає попереджуючий звуковий сигнал і показує повідомлення «Заберіть, будь ласка, гроші» 7а-2. Якщо протягом наступних 15 сек Користувач не забирає гроші, Банкомат забирає гроші назад і робить запис в базі помилок.

Продовження таблиці 4.1

Розділ	Опис
	7а-3. Прецедент завершується з помилкою. 7б. Скасування Якщо до кроку 6 основного потоку Користувач вибирає Відміну, тоді 7б-1. Банкомат друкує чек, що відображає скасування операції. 7б-2. Банкомат повертає карту. 7б-3. Прецедент завершується
6 Key Scenarios	6.1 Немає відповіді від банка.
7. Постумови (Post-conditions)	7.1 Успішне завершення Користувач отримує готівку і його внутрішній лог оновлений. 7.2 Завершення з помилкою Оновлення логів роботи банкомату.
8. Спеціальні вимоги (Special Requirements)	8.1 Банкомат може видавати гроші сумами кратними 10 грн. 8.2 Максимальна сума, яка може бути знята за одну транзакцію, дорівнює 1000 грн. 8.3 Банкомат веде логування всіх завершених і незавершених транзакцій, що включає їх дату і час їх проведення.

4.3 Завдання на лабораторну роботу

4.3.1. Побудувати діаграми прецедентів для своєї предметної області.

4.3.2. Описати декілька (2-3) прецедентів.

4.4 Зміст звіту

4.4.1. Тема і мета роботи.

4.4.2. Результати роботи.

4.4.3. Висновки за результатами роботи.

Лабораторна робота № 5
Побудова діаграм діяльності

5.1 Мета роботи

Ознайомитись з методологією моделювання діяльності на основі мови UML**.**

5.2 Основні теоретичні відомості

Створення інформаційної системи – це складний процес, який можна представити як поетапний спуск від загальної концепції майбутньої ІС, через розуміння її логічної структури до найбільш детальних моделей, що описує фізичну реалізацію.

В якості графічного представлення для виділення основних функцій системи ми застосовуємо діаграму варіантів використання (use case).

Діаграма варіантів використання дає нам уявлення про те, ЩО повинна робити система. На питання ЯК ми можемо відповісти, використовуючи діаграму активності. Тобто якщо варіанти використання ставлять перед системою мету, то діаграма діяльності показує послідовність дій, необхідних для її досягнення.

Діаграми діяльності (Activity diagram), що також називаються діаграмами активності або діаграмами видів діяльності, - це по суті блок-схема, яка показує, як потік управління переходить від однієї діяльності до іншої, при цьому увага фіксується на результаті діяльності. Результат може привести до зміни стану системи або до повернення деякого значення.

Діаграма діяльності – це діаграма, на якій показано розкладання деякої діяльності на її складові частини. Під діяльністю (activity) розуміється специфікація виконуваної поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів – вкладених видів діяльності та окремих дій (action), з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого.

Діаграма діяльності відрізняється від традиційної блок-схеми

– більш високим рівнем абстракції;

– можливістю подання за допомогою діаграм діяльності управління паралельними потоками поряд з послідовним управлінням.

Основними напрямками використання діаграм діяльності є

– візуалізація особливостей реалізації операцій класів;

– відображення внутрішньосистемної точки зору на прецедент.

В останньому випадку діаграми діяльності застосовують для опису кроків, які має вжити система після того, як прецедент ініційовано.

Розробка діаграми діяльності переслідує цілі:

– деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій і прецедентів;

– виділити послідовні і паралельні потоки управління;

– підготувати детальну документацію для взаємодії розробників системи з її замовниками і проектувальниками.

Графічно діаграма діяльності представляється у формі графа діяльності, вершинами якого є стани дії чи стани діяльності (операції, дії), а дугами – переходи від одного стану дії/діяльності до іншого.

Кожна діаграма діяльності повинна мати єдиний початковий і єдиний кінцевий стани (на практиці іноді можна бачити кілька кінцевих станів на одній діаграмі, але це один і той же стан, зображений кілька разів для кращої читабельності діаграми). Саму діаграму діяльності прийнято розташовувати таким чином, щоб дії слідували зверху вниз. У цьому випадку початковий стан буде зображуватися у верхній частині діаграми, а кінцевий – в її нижній частині.

Розглянемо основні елементи діаграми діяльності.

Стан діяльності *(Activity, Process)* – це триваючий у часі неатомарний крок обчислень в автоматі. Стани діяльності можуть бути піддані подальшій декомпозиції, внаслідок чого виконувану діяльність можна представити за допомогою інших діаграм діяльності. Стани діяльності не є атомарними, тобто можуть бути перервані. Передбачається, що для їх завершення потрібно помітний час.

Стани дії *(action state)* – стан, який представляє обчислення атомарної дії, як правило – виклик операції. Стани дії не можуть бути піддані декомпозиції. Вони атомарні, тобто всередині них можуть відбуватися різні події, але виконувана в стані дії робота не може бути перервана. І нарешті, зазвичай передбачається, що тривалість одного стану дії займає невідчутно малий час. Дія може полягати у виклику іншої операції, посилці сигналу, створенні або знищенні об'єкту або в

простому обчисленні значення виразу.

Стани діяльності і стану дії мають однакове стандартне графічне позначення – прямокутник із закругленими краями. Усередині такого символу записують ім'я операції (action-expression), яке має бути унікальним в межах однієї діаграми діяльності. При необхідності можна відобразити параметри операцій за допомогою контактів (pins).

Початковий і кінцевий вузли (initial and final node) на діаграмах діяльності зображуються як зафарбований кружечок і зафарбований кружечок всередині кола, відповідно.

На рис. 5.1 наведено приклад діаграми діяльності.

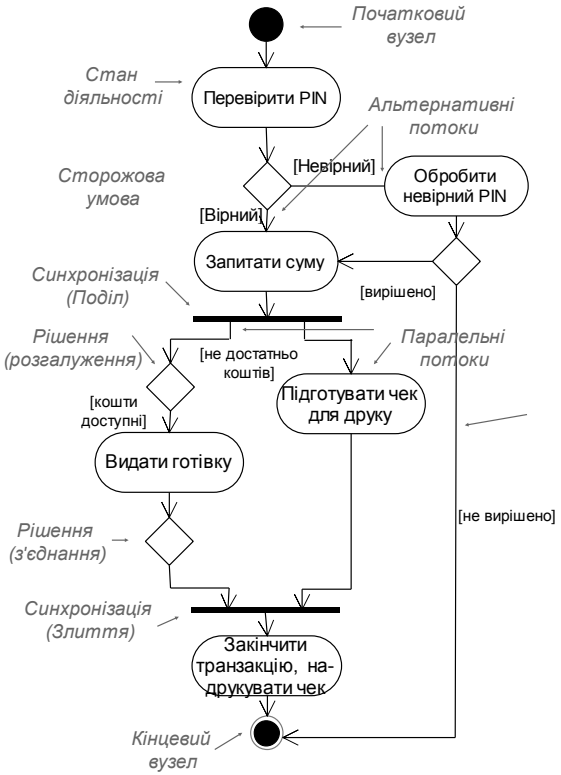


Рисунок 5.1 – Приклад діаграми діяльності

Перехід *(Transitions)* – відношення між двома станами, що показує, що об'єкт, який перебуває у першому стані, повинен

виконати деякі дії і перейти до іншого стану. Коли дія або діяльність в деякому стані завершується, потік управління відразу переходить у наступний стан дії або діяльності. Для опису цього потоку і використовуються переходи, що показують шлях з одного стану дії або діяльності в інший. В UML перехід представляється простою лінією зі стрілкою.

Розгалуження. Прості послідовні переходи зустрічаються найбільш часто, але їх одних недостатньо для моделювання будь-якого потоку управління. Як і в блок-схему, в діаграму діяльності може бути включено *розгалуження або множинний перехід зі сторожовими умовами.* Розгалуження описує різні шляхи виконання залежно від значення деякого булевського виразу. Графічно точка розгалуження представляється ромбом. В точку розгалуження може входити рівно один перехід, а виходити – два або більше. Для кожного вихідного переходу задається булевський вираз, який обчислюється тільки один раз при вході в точку розгалуження. Ні для яких двох вихідних переходів сторожові умови не повинні одночасно приймати значення "істина", інакше потік управління виявиться неоднозначним. Але ці умови повинні покривати всі можливі варіанти, інакше потік зупиниться. Для кожного з переходів має бути явно записано сторожова умова в прямих дужках.

Розділення і злиття. Прості і розгалужені послідовні переходи в діаграмах діяльності використовуються найчастіше. Проте часто виникає потреба зображення паралельних потоків, і це особливо характерно для моделювання бізнес-процесів. В UML для позначення розділення і злиття таких паралельних потоків виконання використовується синхронізаційна риска, яка малюється у вигляді жирної вертикальної або горизонтальної лінії. При цьому поділ *(concurrent fork)* має один вхідний перехід і декілька вихідних, злиття *(concurrent join),* навпаки, має кілька вхідних переходів, які повинні бути виконані перш ніж виконається вихідний.

Приклад поділу і злиття потоків приведений на рис.5.1.

Доріжки. При моделюванні потоків бізнес-процесів іноді буває корисно розбити стан діяльності на діаграмах діяльності на групи, кожна з яких представляє відділ компанії або особу, що відповідає за ту чи іншу роботу. В UML такі групи називаються доріжками (Swimlanes), оскільки візуально кожна група відокремлюється від сусідніх вертикальної рисою, як плавальні доріжки в басейні. Доріжки

– це різновид пакетів, що описують пов'язану сукупність робіт.

Кожній присутній на діаграмі доріжці присвоюється унікальне ім'я. Кожна доріжка представляє сферу відповідальності за частину всієї роботи, зображеної на діаграмі. На діаграмі діяльності, розбитій на доріжки, кожна діяльність належить рівно одній доріжці, але переходи можуть перетинати межі доріжок.

Концептуально діяльність всередині кожної доріжки може бути розглянута окремо від діяльності в сусідніх доріжках.

Об'єкти. Дії на діаграмі діяльності виконуються над тими чи іншими об'єктами. Об'єкти ініціюють виконання дій або визначають деякий їх результат.

Об'єкти відіграють певну роль в розумінні процесу діяльності – іноді виникає необхідність явно вказати їх на діаграмі. Графічне представлення об'єктів – прямокутник з ім'ям. Після імені може вказуватися характеристика стану об'єкту в прямих дужках.

Прямокутники об'єктів приєднуються до станів дії відношенням залежності пунктирною лінією зі стрілкою.

Відповідна залежність визначає стан конкретного об'єкту після виконання попередньої дії.

На діаграмі діяльності з доріжками розташування об'єкта може мати деякий додатковий сенс:

1) об'єкт розташований на границі двох доріжок – перехід до наступного стану дії в сусідній доріжці асоційований з готовністю деякого документа (об'єкт в деякому стані);

2) об'єкт цілком розташований усередині доріжки – стан цього об'єкта цілком визначається діями даної доріжки.

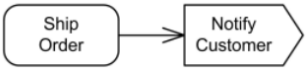
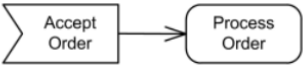
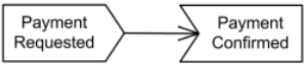
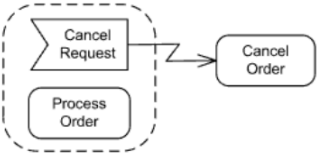
Окремі види станів

Діаграма дії також може описувати поведінку, на яку впливають зовнішні події, що відбуваються за межами даної системи чи процесу. На діаграмі це може бути показано за допомогою зображення передачі сигналу (табл. 5.1).

Приклад діаграми діяльності з використанням сигналів наведено на рисунках 5.2 та 5.3.

На рисунку 5.4 наведено приклад діаграми діяльності для системи банкомату.

Таблиця 5.1 – Передача сигналів

Подія	Графічне представлення
Подія відправки сигналу (Send signal action) – створює сигнал з вхідних даних, передаючи їх кінцевому об'єкту	
Подія обробки вхідної події (Accept event action) – очікує виникнення події. Обробляє асинхронні виклики	
Якщо подією, яка очікується, є подія типу send, використовується наступна комбінація	
Часовий сигнал (time signal) приходить як мине час і викликає подію часу (wait time action).	
Стан переривання необхідний для вказівки тієї сукупності станів, у яких можливе виникнення переривання.	

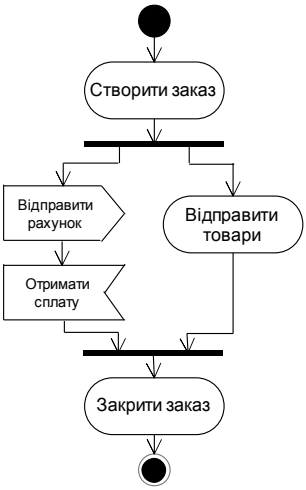


Рисунок 5.2 – Приклад використання сигналів: події відправлення та отримання сигналів

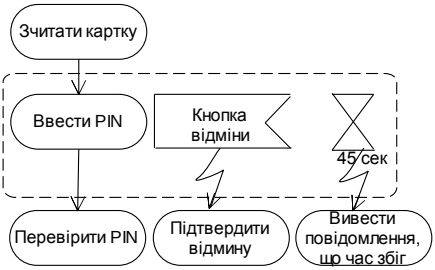


Рисунок 5.3 – Приклад використання сигналів: часовий сигнал

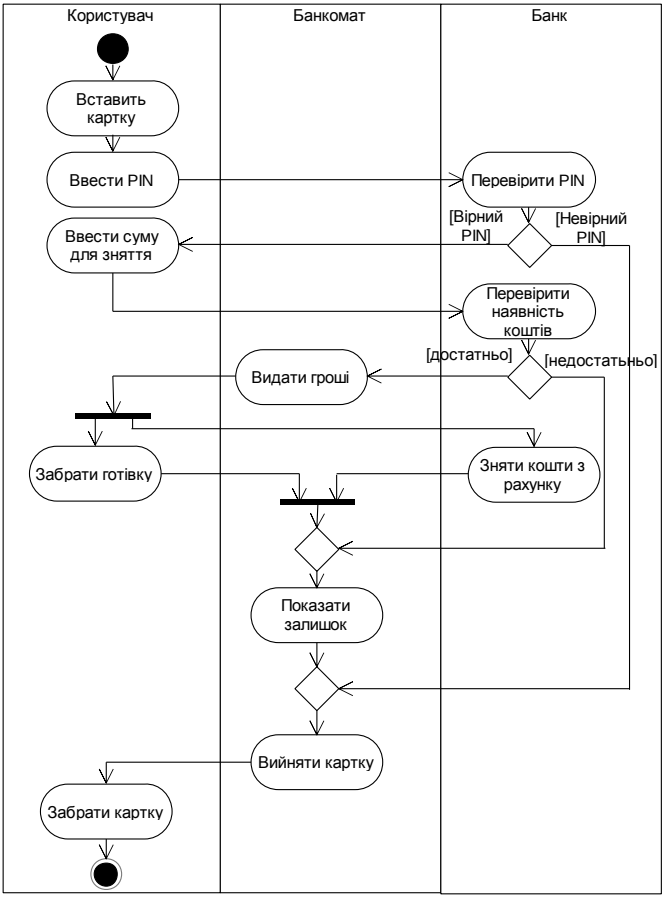


Рисунок.5.4 - Приклад діаграми діяльності для системи банкомату

5.3 Завдання на лабораторну роботу

5.3.1.Побудуйте діаграми діяльності для кожного прецеденту присутнього на діаграмі прецедентів, розробленої при виконанні лабораторної роботи № 4.

5.4 Зміст звіту

5.4.1. Тема та мета роботи.
5.4.2. Результати роботи:
5.4.4. Висновки за результатами роботи.

ЛІТЕРАТУРА

1. ГОСТ 19.201-78. Единая система программной документации. Техническое задание, требования к содержанию и оформлению. – Введ. 01.01.1980. – М.: Изд-во стандартов, 2010. – 2 с.

2. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 01.01.1990. – М.: Изд-во стандартов, 2004. – 11 с.

3. ГОСТ 25123-82. Машины вычислительные и системы обработки данных. Техническое задание. Порядок построения, изложения и оформления. – Введ. 01.01.1983. – М.: Изд-во стандартов, 1982. – 7 с.

4. IEEE 830-1998. IEEE Recommended Practice for Software Requirements Specifications. – Approved 25.06.1998. – Institute of Electrical and Electronics Engineers, Inc., 1998. – 37 p.

5. Shelly, Gary B. Systems analysis and design / Gary B. Shelly, Harry J. Rosenblatt. –9th ed. – Course Technology, 2012. – 761 p.

6. Whitten, Jeffrey L. Systems Analysis and Design methods / Jeffrey L. Whitten, Lonnie D. Bentley. – 7th ed. – McGrow-Hill, 2007. – 765 p.

7. Satzinger , John W. Systems Analysis and Design in a Changing World / John W. Satzinger, Robert B. Jackson, Stephen D. Burd. – 6th ed. – Course Technology, 2012. – 514 p.

8. Элизабет Халл, Кен Джексон, Джереми Дик. Разработка и управление требованиями. Практическое руководство пользователя. – 2005. – 229 с. – Глава 3.Системное моделирование для разработки тренований.

9. Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с.

ДОДАТОК А
ПЕРЕЛІК ТЕМ ЛАБОРАТОРНИХ РОБІТ

Перелік можливих предметних областей для проведення автоматизації:

1. Бібліотека
2. Ресторан або кафе
3. Пункт відео-прокату
4. Ветеринарна клініка
5. Каса театру або кінотеатру
6. Продаж та бронювання номерів готелю
7. Туристична агенція
8. Перукарня або салон краси
9. Реєстратура поліклініки
10. Облік продажу товарів невеликого магазину
11. Складський облік
12. Електронна енциклопедія
13. Тренажерный зал
14. Фотоцентр з фотокіосками
15. Комунальні платежі
16. Відділ кадрів
17. Інтернет-магазин
18. Система дистанційної освіти
19. Диспетчерський центр таксі
20. Прохідна
21. Каса автобусної станції з регіональних перевезень

Наведений перелік не обмежує студентів в виборі тем.

Діаграми до лабораторних робіт можуть бути побудовані в будь-якому графічному редакторі або за допомогою демонстраційних версій спеціальних CASE-засобів: Microsoft Visio, Visible Analyst , Systems architect , Edraw Max , Smart Draw тощо. Крім того, можливе використання он-лайн засобів, таких як https://www.lucidchart.com.