МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ

СХЕМОКОНСПЕКТ ЛЕКЦІЙ

з навчальної дисципліни

«Проєктування інформаційних систем»

для студентів денної та заочної форм навчання
галузі знань F Інформаційні технології
спеціальності F3 Комп’ютерні науки

Дрогобич, 2025

УДК 004.415.26(042.3)
С 92
Схемоконспект лекцій з навчальної дисципліни «Проєктування інформаційних систем» для студентів денної та заочної форм навчання галузі знань F Інформаційні технології спеціальності F3 Комп’ютерні науки / уклад. Т.В. Алтухова. – Дрогобич: ДонНТУ, 2025. – 356 с.

Навчальна дисципліна «Проєктування інформаційних систем» є актуальною і затребуваною в контексті підготовки висококваліфікованих фахівців у галузі знань F Інформаційні технології спеціальності F3 Комп’ютерні науки. Схемоконспект лекцій охоплює основні питання в сфері проєктування інформаційних систем та дозволить забезпечити студентам необхідну допомогу під час вивчення ними ключових положень базових тем курсу. Схематична форма конспекту акцентує увагу студентів на закріпленні основних фахових знань, формуванні відповідних умінь та навичок і в цілому отриманні ними кваліфікаційних компетентностей.

Укладач: Тетяна АЛТУХОВА, канд. техн. наук, доцент кафедри ПМІ ДВНЗ «ДонНТУ»

Рецензент: Сергій КОВАЛЬОВ, канд. техн. наук, доц., завідувач кафедри ЕТКІ ДВНЗ «ДонНТУ»
Відповідальний за випуск: Ярослав ДОРОГИЙ, д-р. техн. наук, проф., в.о завідувача кафедри ПМІ ДВНЗ «ДонНТУ»
Затверджено навчально-методичним відділом ДонНТУ, протокол №6 від 29.04.2025 р.
Розглянуто на засіданні кафедри ПМІ ДВНЗ «ДонНТУ» протокол №4 від 03.04.2025р.

ЗМІСТ

ВСТУП

Сучасний стан інформаційних технологій (ІТ) характеризується швидким розвитком та трансформацією багатьох галузей, тому питання проєктування інформаційних систем є на даний час досить актуальним. З огляду на це, ***метою*** викладання даної дисципліни є ***формування базових теоретичних знань та практичних навичок з проєктування інформаційних систем та забезпечення бвзової профілюючої підготовки за фахом***

В результаті ***опанування*** навчальної дисципліни «Проєктування інформаційних систем» студент повинен отримати:

Знання: задачі, функції і вимоги до ІС; види ІС; стандарти проєктування ІС та оформлення проєктної документації; системний підхід до проєктування ІС; топології та архітектури ІС; технології проєктування; методології, технології створення та супроводу ІС, реінжиніринг ІС

Уміння: аналізувати вітчизняний та закордонний досвід у сфері проєктування ІС; виявляти та аналізувати вимоги до ІС; специфікувати та документувати вимоги до ІС; проєктувати моделі даних та моделі процесів; застосувати стандарти проєктування та сучасні технології створення й супроводу ІС; самостійно оволодівати новітніми методами, засобами, інструментами проєктування ІС; формулювати перспективні та обґрунтовані ідеї щодо проєктування ІС.

При вивченні курсу «Проєктування інформаційних систем» передбачено опанування студентами 16 тем, що поділені на декілька змістовних тематичних питань. Стисле, але вичерпне подання матеріалу забезпечить студентам можливість зосередитися на основних ключових поняттях в галузі ***проєктування інформаційних систем*** в цілому сприятиме формуванню в них необхідних для кваліфікаційного рівня «Бакалавр» знань, умінь і компетентностей.

ТЕМА №1. ЗМІСТОВНІ АСПЕКТИ ПРОЄКТУВАННЯ

Тематичний план

1.1 Визначення сутності терміну «проєктування» та його мети

1.2 Основні принципи проєктування

1.3 Огляд базової термінології в межах проєктування

1.1 ВИЗНАЧЕННЯ СУТНОСТІ ТЕРМІНУ «ПРОЄКТУВАННЯ» ТА ЙОГО МЕТИ

Основна мета дисципліни «Проєктування інформаційних систем»:

- вивчення основних понять і методології проєктування;
- вивчення об’єктно-орієнтованого підходу до проєктування на базі використання мови UML;
- набуття практичних навичок проєктування інформаційних систем (ІС).

Головна мета – вивчення основ методології проєктування складних об’єктів і ІС.

Система – це упорядкована сукупність взаємозалежних і взаємодіючих елементів, які закономірно утворюють єдине ціле.

Інформаційна система (ІС) призначена для своєчасного забезпечення належних людей належною інформацією, тобто для задоволення конкретних інформаційних потреб в рамках певної предметної області, при цьому результатом функціонування інформаційних систем є інформаційна продукція.

При проєктуванні ІС – перша задається мета.

Для визначення меж ІС – визначаються значимі об'єкти, властивості, функції і закономірності

Складність систем ***характеризується*** кількістю елементів і типом зав’язків між ними.

Проєктування ***характеризується*** та відрізняється наявністю великого числа варіантів рішень, необхідністю урахування багатьох факторів, що впливають на структуру та параметри, і базується на блочно-ієрархічному підході, сутність якого полягає в декомпозиції об’єкта чи системи до декількох рівнів.


ТЕРМІН «ПРОЄКТУВАННЯ»
процес складання опису, необхідного для створення в заданих умовах ще нествореного об’єкту
процес створення проєкту, прототипу, прообразу майбутнього об’єкта, стану та способів його виготовлення
комплекс робіт

Рисунок 1.1

Проєктування (техніка) – це розробка проєктної, конструкторської та іншої технічної документації, призначеної для забезпечення процесу створення нових видів і зразків.

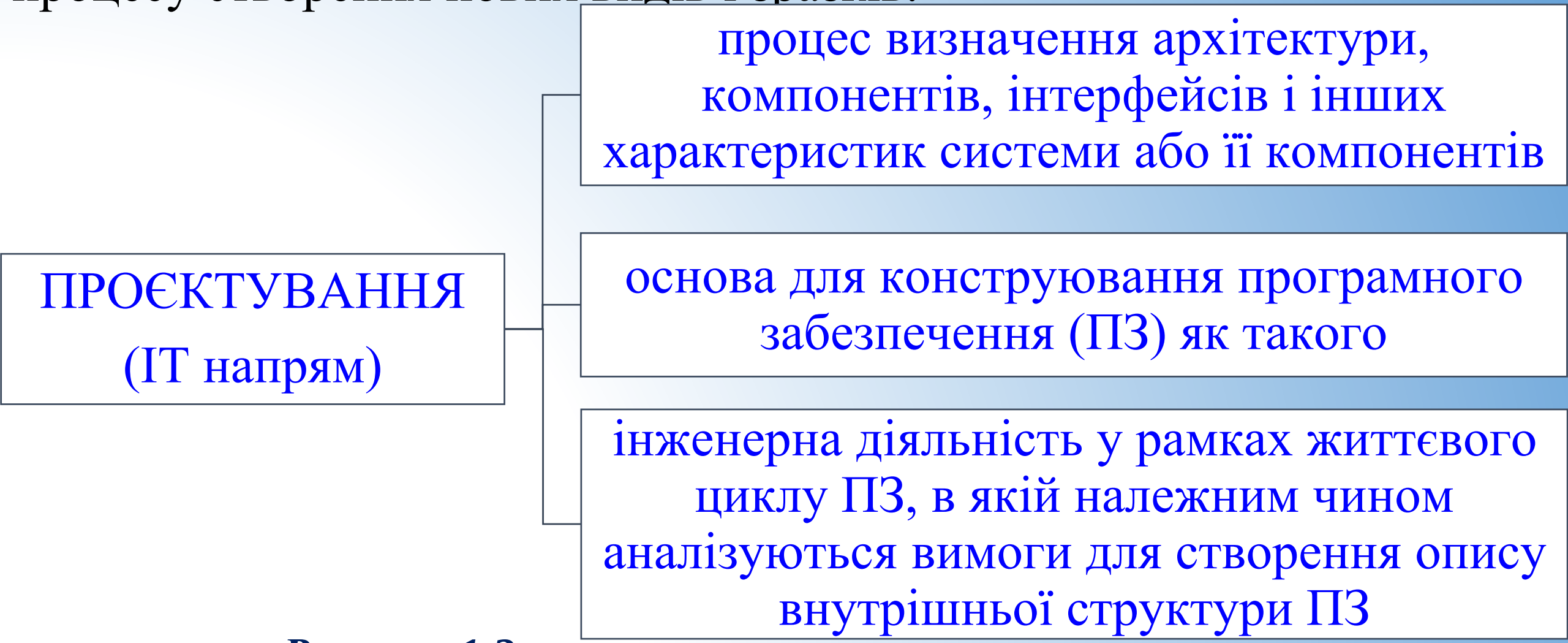


Рисунок 1.2

Мета проєктування – пошук, фіксація та документальне оформлення інформації про об'єкт проєктування, яка необхідна для його створення.

Під час проєктування необхідно:

- перетворити вимоги до програмного елемента в архітектуру, яка описує його загальну структуру та ідентифікує програмні компоненти;
- розробити та задокументувати загальний проєкт як зовнішніх щодо програмного елемента інтерфейсів, так і інтерфейсів між програмними компонентами програмного елемента;
- розробити та задокументувати загальний проєкт бази даних;
- розробити та задокументувати попередню версію користувацької документації;
- визначити та задокументувати попередні вимоги до тестування та

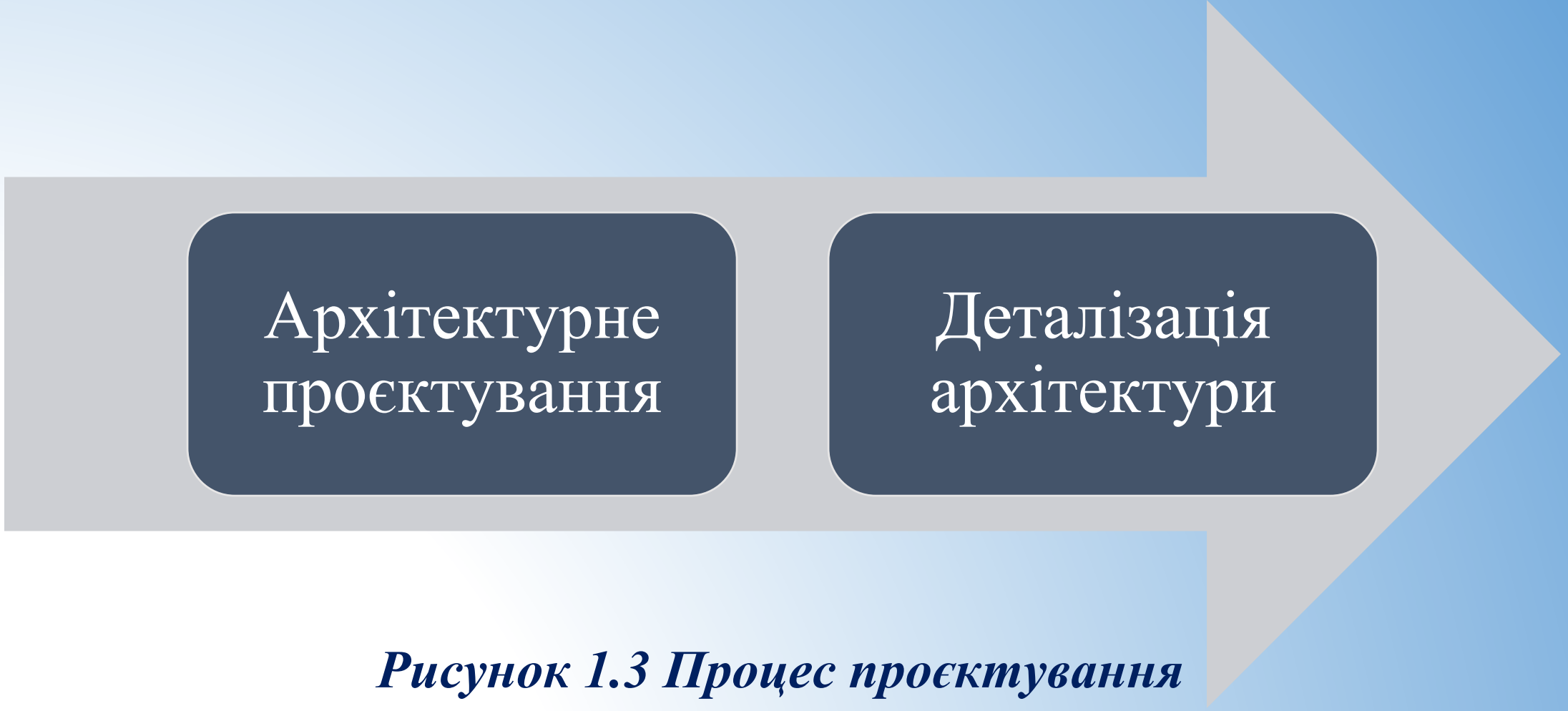


Рисунок 1.3 Процес проєктування

Результат проєктування – це набір моделей і артефактів, що містять результати рішень, прийнятих за способами реалізації вимог в програмному коді.

Особливості сучасних великомасштабних проєктів створення ПС:

1. Характеристики об'єкта впровадження:

- структурна складність і територіальна розподіленість;
- функціональна складність;
- інформаційна складність, складна технологія проходження документів;
- складна динаміка поведінки.

2. Технічні характеристики проєктів:

- різна ступінь уніфікованості проєктних рішень в рамках одного проєкту;
- висока технічна складність;
- відсутність аналогів;
- велика кількість і висока вартість успадкованих застосунків;
- велика кількість локальних об’єктів впровадження, територіально розподілене і неоднорідне середовище функціонування;
- велика кількість зовнішніх взаємодіючих систем з різними форматами обміну інформацією.

3. Організаційні характеристики проєктів:

- різні форми організації та управління проєктом;
- велика кількість учасників проєкту;
- значна тривалість життєвого циклу системи;
- високі вимоги з боку замовника до рівня технологічної зрілості організацій-розробників.

1.2 ОСНОВНІ ПРИНЦИПИ ПРОЄКТУВАННЯ

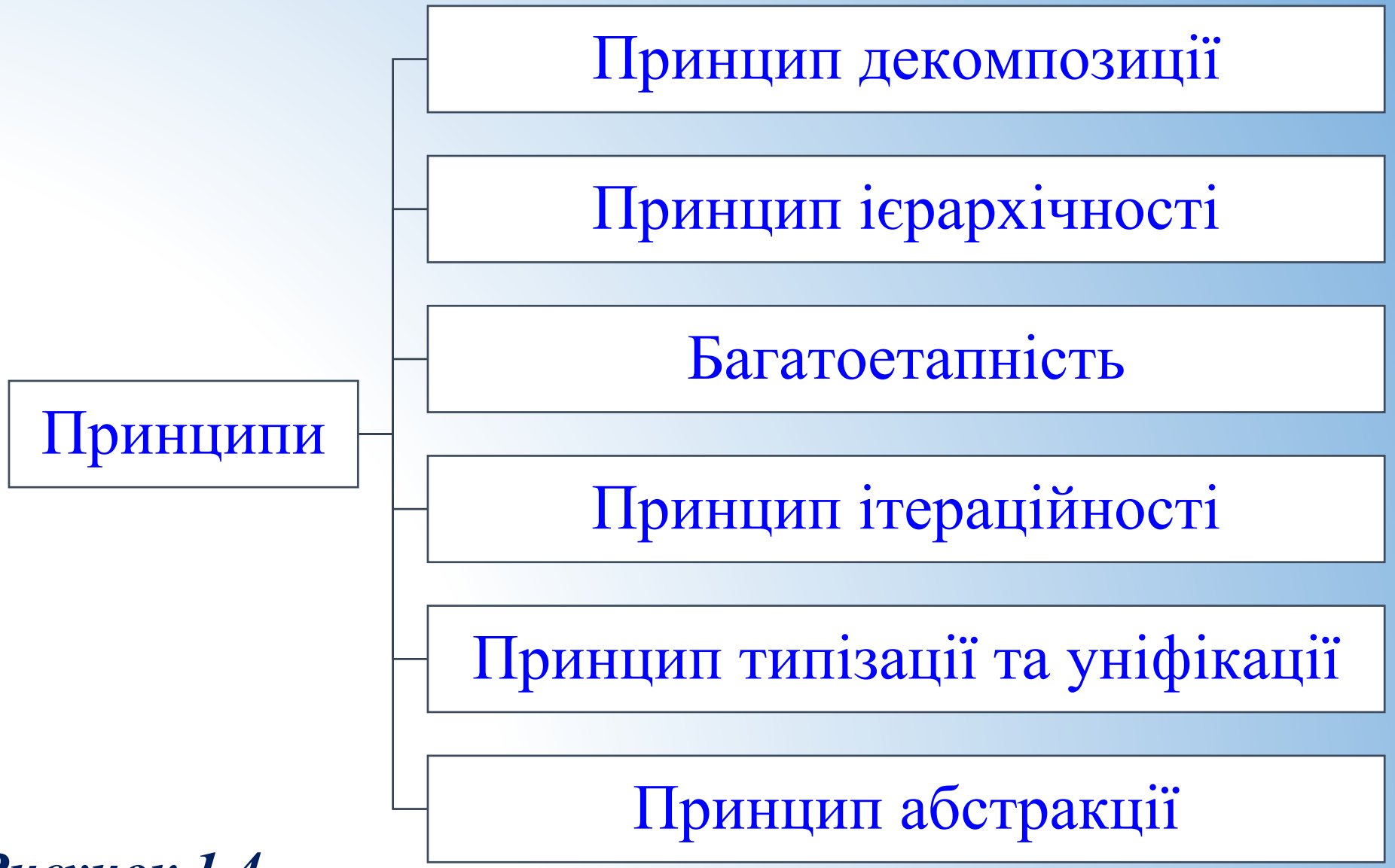


Рисунок 1.4

1.3 ОГЛЯД БАЗОВОЇ ТЕРМІНОЛОГІЇ В МЕЖАХ ПРОЄКТУВАННЯ

Проєктування – це комплекс робіт, ціллю яких є отримання опису ще неіснуючого технічного об’єкта, який достатній для реалізації і виготовлення об’єкта в заданих умовах.

Об’єкт проєктування – це виріб чи процес.

Процес – послідовна зміна станів об’єкту в часі або сукупність ряду послідовних дій, спрямованих на досягнення певного результату.

Неавтоматизоване проєктування – це проєктування, під час якого весь процес проєктування здійснює людина.

Автоматизоване проєктування – це найбільш поширене проєктування, при якому відбувається взаємодія людини та ЕОМ.

Автоматичне проєктування – це проєктування, при якому всі перетворення описів об'єкта й алгоритму його функціонування здійснюються без участі людини.

Завдання на проєктування – це первинний опис технічного об'єкту (ТО), представлений в заданій формі.

Проєктне рішення – це проміжний або кінцевий опис об'єкта проєктування, необхідний і достатній для розгляду та визначення подальшого напрямку або закінчення проєктування.

Результат проєктування – це проєктне рішення або їхня сукупність, що задовольняють заданим вимогам, необхідні для створення об'єкта проєктування.

Проєктний документ – це документ, виконаний за заданою формою, у якому представлене яке-небудь проєктне рішення, отримане при проєктуванні.

Проєкт – це сукупність проєктних документів відповідно до встановленого переліку, у якому представлений результат проєктування.

Проєктна процедура – це формалізована сукупність дій, виконання яких закінчується проєктним рішенням.

Мова проєктування – це мова, призначена для представлення та перетворення описів при проєктуванні.

Алгоритм проєктування – це сукупність розпоряджень, необхідних для виконання проєктування.

Програмний дизайн системи – це результат діяльності з проєктування, цілісний погляд на архітектуру системи.

Шаблон проєктування – це «загальне рішення загальної проблеми в заданому контексті»; визначає приватні аспекти деталей архітектури системи.

Нотація – це угода про представлення; візуальне подання/представлення.

Програмне забезпечення проєктування – комп'ютерні програми, необхідні для здійснення процесу проєктування.

Система автоматизованого проєктування або ***автоматизована система проєктування:***

- організаційно-технічна система, що здійснює автоматизоване проєктування об’єктів;
- автоматизована система, призначена для автоматизації технологічного процесу проєктування, результатом якого є комплект проєктно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об’єкта проєктування

Операційна система автоматизованого/автоматичного проєктування – це частина програмного забезпечення автоматизованого/автоматичного проєктування, призначена для управління проєктуванням.

Модуль – це частина задачі, що має самостійне значення та володіє відносною незалежністю.

Компонент – це різновид, складова частина чогось.

Підсистема – це набір елементів, що представляють автономну усередині системи галузь.

Архітектура системи – це опис підсистем і компонент програмної системи, а також зв’язків між ними.

Архітектурне подання – приватні аспекти програмної архітектури, що розглядають специфічні властивості програмної системи.

Архітектурний стиль – метамодель проєктування макроархітектури; набір обмежень, що визначають сімейство архітектур.

Функціонально-орієнтоване/структурне проєктування – це один з класичних методів проєктування, в якому декомпозиція сфокусована на ідентифікації основних програмних функцій і, потім, детальної розробки та уточнення цих функцій «зверху-вниз».

Об'єктно-орієнтоване проєктування – безліч методів проєктування, які базуються на концепції об'єктів.

Проєктування на основі структур даних – різновид проєктування, в якому фокус сконцентрований на структурах даних, якими управляє система.

Компонентне проєктування – різновид проєктування, покликаний вирішити задачу використання, розробки та інтеграції компонентів з метою підвищення повторного використання активів.

ТЕМА №2. ВСТУПНІ ПОЛОЖЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

2.1 Базові поняття ІТ та ІС. Визначення мети створення ІС

2.2 Огляд етапів розвитку ІС

2.3 Визначення принципів створення ІС

2.4 Класифікація ІС

2.1 БАЗОВІ ПОНЯТТЯ ІТ ТА ІС. ВИЗНАЧЕННЯ МЕТИ СТВОРЕННЯ ІС

Комп'ютерні технології – методи застосування обчислювальної техніки і комп'ютерного програмного забезпечення під час роботи з інформацією.

Інформаційна технологія (ІТ) – це комплекс взаємопов'язаних наукових й інженерних дисциплін, що вивчають ОТ, способи взаємодії її з людьми та виробничим обладнанням, способи ефективної організації праці людей, а також пов'язані з усім цим проблеми.

ІТ – це вивчення, проєктування, розроблення, впровадження, підтримка або управління комп'ютерними інформаційними системами.

ІТ – це цілеспрямована організована сукупність інформаційних процесів з використанням засобів обчислювальної техніки, що забезпечують високу швидкість обробки даних, пошуку інформації, розосередження даних й доступ до джерел інформації, незалежно від місця їхнього розташування.

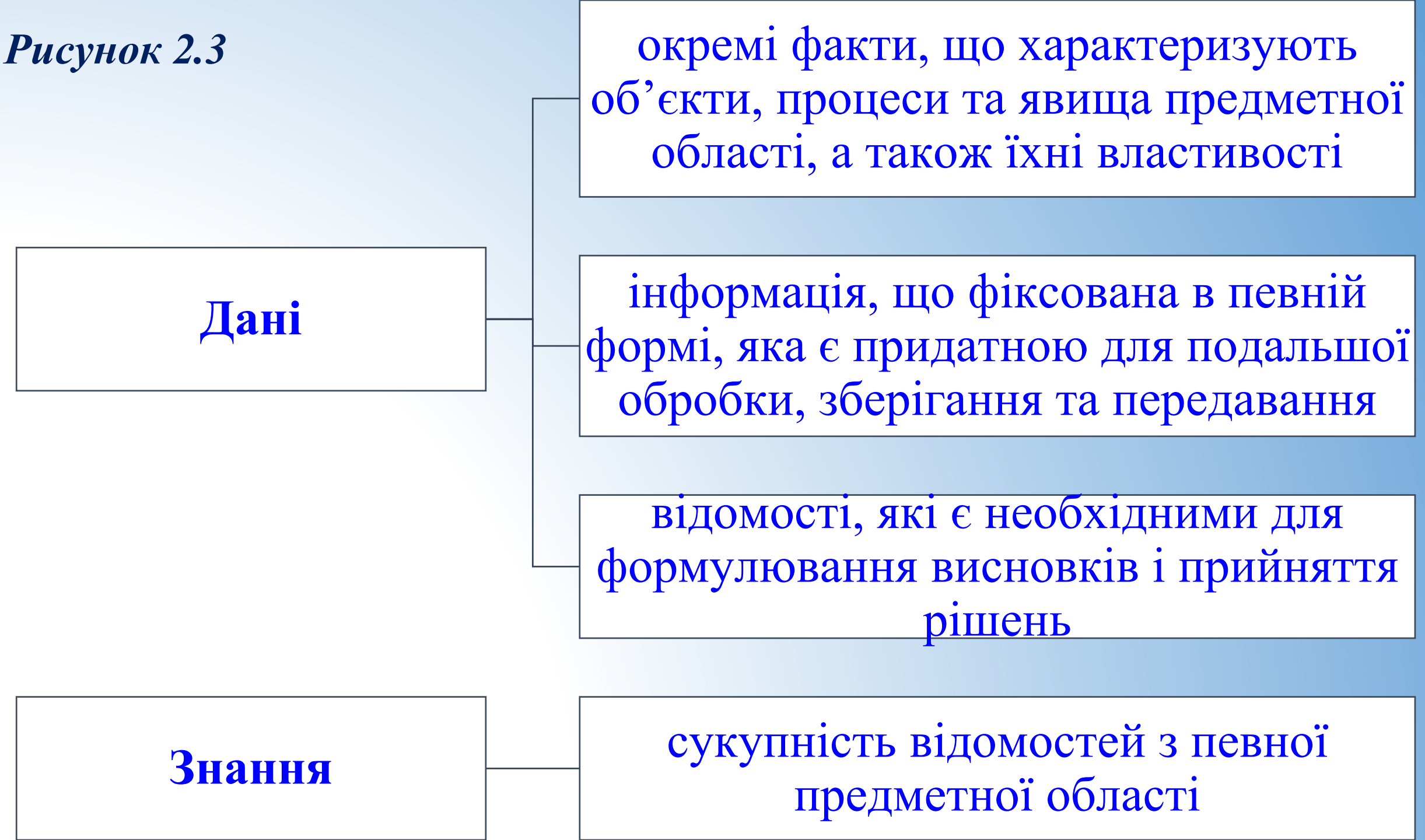
Рисунок 2.1

Інформаційно-комунікаційні технології (ІКТ) – це всі технічні засоби обробки та обміну інформацією в поєднанні з відповідним математичним та програмним забезпеченням.

Рисунок 2.2

Інформація в ІС може бути ***структурованою*** або ***неструктурованою*** та умовно поділяється на ***дані*** та на ***знання***.

Рисунок 2.3


Дані
окремі факти, що характеризують об’єкти, процеси та явища предметної області, а також їхні властивості
інформація, що фіксована в певній формі, яка є придатною для подальшої обробки, зберігання та передавання
відомості, які є необхідними для формулювання висновків і прийняття рішень
Знання
сукупність відомостей з певної предметної області

Інформаційні ресурси (ІР):

- інформація, що зафіксована на матеріальних носіях і зберігається в ІС;
- окремі документи й окремі масиви документів в інформаційних системах, які організовані для багаторазового використання та вирішення проблем користувача.

Інформаційний продукт (ІП) – документований інформаційний ресурс, підготовлений відповідно до потреб користувачів і поданий у формі товару.

Система – це сукупність взаємозв'язаних між собою елементів, підлеглих єдиній меті, щоб могла реалізовуватися функція системи.

Елемент системи – це частина системи, що має визначене функціональне призначення.

Структура системи – це сукупність внутрішніх стійких зв'язків між елементами системи, що визначає її основні властивості.

Цілісність системи – це зведення властивостей системи до суми властивостей, що її формують, і залежність властивостей кожного елемента від його місця і функцій усередині системи.

Проста система – це система, що складається з обмеженої кількості елементів і не має розгалуженої структури.

Складна система – це система з розгалуженою структурою і значною кількістю взаємозалежних елементів.

Стан системи – це зафіксовані значення характеристик системи, важливі для цілей дослідження.

Процес – це набір станів системи, що відповідає впорядкованій неперервній або дискретній зміні деякого параметра, що визначає характеристики чи властивості системи.

IT-система/інформаційна система/IT system – це сукупність ресурсів інформаційних технологій.

Корпоративна інформаційна система (КІС) – сукупність спеціалізованого ПЗ і обчислювальної апаратної платформи, на якій встановлено та налаштовано ПЗ.

Комп'ютерна інформаційна система – це формальна інформаційна система, заснована на комп'ютерній технології.

Мета створення ІС у гранично короткі терміни створити систему обробки даних, яка має наступні задані споживчі якості:

- функціональна повнота;
- своєчасність;
- функціональна надійність;
- адаптивна надійність;
- економічна ефективність ІС.

Призначення ІС: отримання (введення або збір), зберігання, пошук, передача та обробка даних, інформації.

2.2 ОГЛЯД ЕТАПІВ РОЗВИТКУ ІС

Етапи розвитку ІС

- **Перший етап** – ІС першого покоління виникли на початку 60-х р. ХХго ст. при необхідності автоматизації управління підприємством на базі великих ЕОМ і централізованого оброблення інформації
- **Другий етап** (70-80-і р. ХХ ст.) характерний розробленням програмних продуктів відповідно до концепцій планування потреби в матеріалах і планування ресурсів підприємства
- **Третій етап** (початок 90-х р.) характерний розробкою програмних продуктів відповідно до концепції ERP
- **Четвертий етап** (початок третього тисячоліття) характерний глобальною комп'ютеризацією суспільства

Рисунок 2.4

Особливості ІС четвертого покоління:

- максимальне використання потенціалу ПК і середовища розподіленої обробки даних;
- модульна побудова системи;
- економія ресурсів системи за рахунок централізації зберігання та обробки даних на вищих рівнях системи;
- наявність ефективних централізованих засобів мережевого системного адміністрування.

2.3 ВИЗНАЧЕННЯ ПРИНЦИПІВ СТВОРЕННЯ ІС

Створення ІС передбачає:

1. *частковий чи повний перегляд методів і засобів функціонування ІС об'єкта управління;*
2. *виконання наступних завдань:*
   - виявлення його суттєвих характеристик;
   - створення математичних і фізичних моделей досліджуваної системи та її елементів;
   - встановлення умов взаємодії людини та комплексу технічних засобів;
   - детальна розробка окремих проєктних рішень;
   - аналіз проєктних рішень, практична апробація та впровадження.

Прийняття рішення про необхідність створення ІС

Рисунок 2.5

Принципи створення ІС поділяють на дві частини:

1) ***загальні принципи:*** мають універсальний характер і визначають методологічний підхід до створення будь-яких об’єктів (принципи: науковості, нормативності, неперервності, розвитку, ефективності, послідовності, від загального до часткового, системності, комплексності, використання типових і керівних матеріалів);
2) ***часткові принципи:*** систему управління потрібно розглядати як людино-машинну; передбачається чіткий поділ системи на складові, забезпечення сумісності й зв’язку між усіма видами забезпечення; забезпечення єдності обліку, типізація, уніфікація та стандартизація.


Основні принципи створення ІС, які враховуються в першу чергу
Принцип системності
Принцип розвитку (відкритості)
Принцип сумісності
Принцип стандартизації (уніфікації)
Принцип ефективності

Рисунок 2.6

Створення ІС

Рисунок 2.7

2.4 КЛАСИФІКАЦІЯ ІС

Рисунок 2.8


Класифікація ІС
За характером обробки даних
Фнформаційно-пошукові/довідкові
Інформаційно-вирішальні
За використанням вихідної інформації
ІС, що управляють
ІС, що надають поради та рекомендації
За масштабом
однокористувацькі
групові
корпоративні/ інтегровані
глобальні

Рисунок 2.9


Класифікація ІС
За рівнем управління
ІС оперативного рівня
ІС фахівців
ІС рівня менеджменту
Стратегічні ІС
За архітектурою
З архітектурою «файл-сервер»
З архітектурою «клієнт-сервер»
З багаторівневою архітектурою
на базі технології Internet/Intranet

Рисунок 2.10


Класифікація ІС
За рівнем діяльності
державні
регіональні/територіальні
галузеві
ІС об’єднання
управління технологічними процесами
За ступенем централізації та інтеграції обробки даних
централізовані
децентралізовані
колективного використання або комбіновані
За призначанням
адміністративні
виробничі
соціальні
тарнспортні і інш.

Рисунок

ТЕМА №3. ВИМОГИ ДО ІНФОРМАЦІЙНИХ СИСТЕМ ТА ЇХ ФУНКЦІЇ

Тематичний план

3.1 Визначення вимог до ІС

3.2 Огляд функцій ІС

3.3 Регламент функціонування ІС

3.1 ВИЗНАЧЕННЯ ВИМОГ ДО ІС

Рисунок 3.1

Головна вимога до ІС: забезпечення підвищення ефективності, що приводити до отримання корисних техніко-економічних, соціальних чи інших результатів.

Причини складності розробки вимог до ІС полягають у необхідності їх вірного виокремлення та формулювання.

Групи вимог:

- вимоги, визначені державними стандартами, методичними матеріалами галузі замовника;
- вимоги, які відбивають специфіку об’єкта управління.

Категорії представлення вимог до ІС:

- ***вимоги замовника*** – документують бажання і потреби замовника та пишуться мовою, зрозумілою замовнику;
- ***вимоги розробника*** – документуються вимоги в спеціальній, структурованій формі; вони деталізовані стосовно до первинних вимог.

Збір або формування вимог – це робота із створення первинних вимог.
Аналіз вимог – це робота із створення детальних вимог.

Види вимог:

- ***функціональні вимоги***, які описують поведінку системи і сервіси/функції;
- ***нефункціональні вимоги***, які стосуються системи в цілому.

Властивості вимог:

- ясність;
- погодженість;
- повнота.

Групи базових вимог до ІС:

- вимоги до ІС в цілому;
- вимоги до функцій ІС;
- вимоги до підготовленості персоналу;
- вимоги до видів забезпечення;
- вимоги до безпеки ІС.

Вимоги до ІС

Основні	Системність
	Відкритість
	Стандартизація/уніфікація
	Здійснення узгоджених між собою процесів
Спеціальні	Повнота інформації
	Ієрархічність
	Семантична єдність
	Переносність елементів системи
	Комплексна безпека
Додаткові	Гнучкість
	Надійність
	Ефективність
	Безпека

1. *Вимоги до системи в цілому:*

1.1. Вимоги до структурних характеристик і режимів функціонування системи:

- склад основних функцій;
- об’єктна структура системи;
- вимоги до засобів і способів обміну інформацією між об’єктними підсистемами в разі їх територіальної роз’єднаності;
- вимоги до інтегрованості з суміжними системами або вже реалізованими елементами створюваної системи, з якими повинна бути забезпечена можливість взаємодії;
- вимоги до режимів функціонування системи.

1.2. Вимоги до показників призначення (до найважливіших характеристик системи).

1.3. Вимоги до надійності:

- перелік відмов системи або її частин, за якими слід висувати вимоги щодо надійності;
- склад і кількісні значення показників надійності за типами відмов для системи або її елементів;
- вимоги до методів оцінки та контролю надійності на різних етапах створення системи.

1.4. Вимоги до якості даних:

- показники достовірності даних і їх кількісні значення;
- ситуації, при яких повинна бути забезпечена схоронність даних;
- можливі способи несанкціонованого доступу до даних, від яких система повинна бути захищена.

1.5. Вимоги з стандартизації та уніфікації: використані стандарти при створенні системи документообігу, класифікатори, які використовуються, вимоги щодо застосування типових програмних і технічних засобів при створенні системи.

1.6. Вимоги до розвитку системи: можливості модифікації, включення нових функцій, відкритості, масштабованості.

2. Вимоги до функцій (задач), що виконуються системою:

- переліки задач до кожної функціональної підсистеми з їх розподілом за рівнями системи;
- вимоги до якості реалізації кожної функції;
- форми подання вхідної та вихідної інформації;
- тимчасовий регламент;
- вимоги до якості результатів

3. Вимоги до видів забезпечення:

- вимоги до ***інформаційного забезпечення*** можуть включати в себе вимоги до якості даних, складу і способу організації даних, їх сумісності із суміжними системами, використання класифікаторів та уніфікованих документів, методів контролю, зберігання, обновлення і відновлення даних;
- до складу вимог до ***програмного забезпечення*** можуть входити вимоги до якості програмних засобів, до інтерфейсів, мов програмування, що використовуються, до операційної системи;
- до складу вимог до ***технічного забезпечення*** можуть входити вимоги до функціональних, конструктивних, експлуатаційних характеристик окремих видів апаратних засобів.


Способи подання вимог до ІС
Список необхідних елементів
Перелік можливих елементів
Формування вимог якісного типу
Кількісні показники

Рисунок 3.2

Рисунок 3.3 Стадії створення ІС

Управління вимогами – це вид комунікаційних робіт із взаємодії замовника і виконавця з метою виявлення вимог на всіх етапах життєвого циклу ІС.

3.2 ОГЛЯД ФУНКЦІЙ ІС

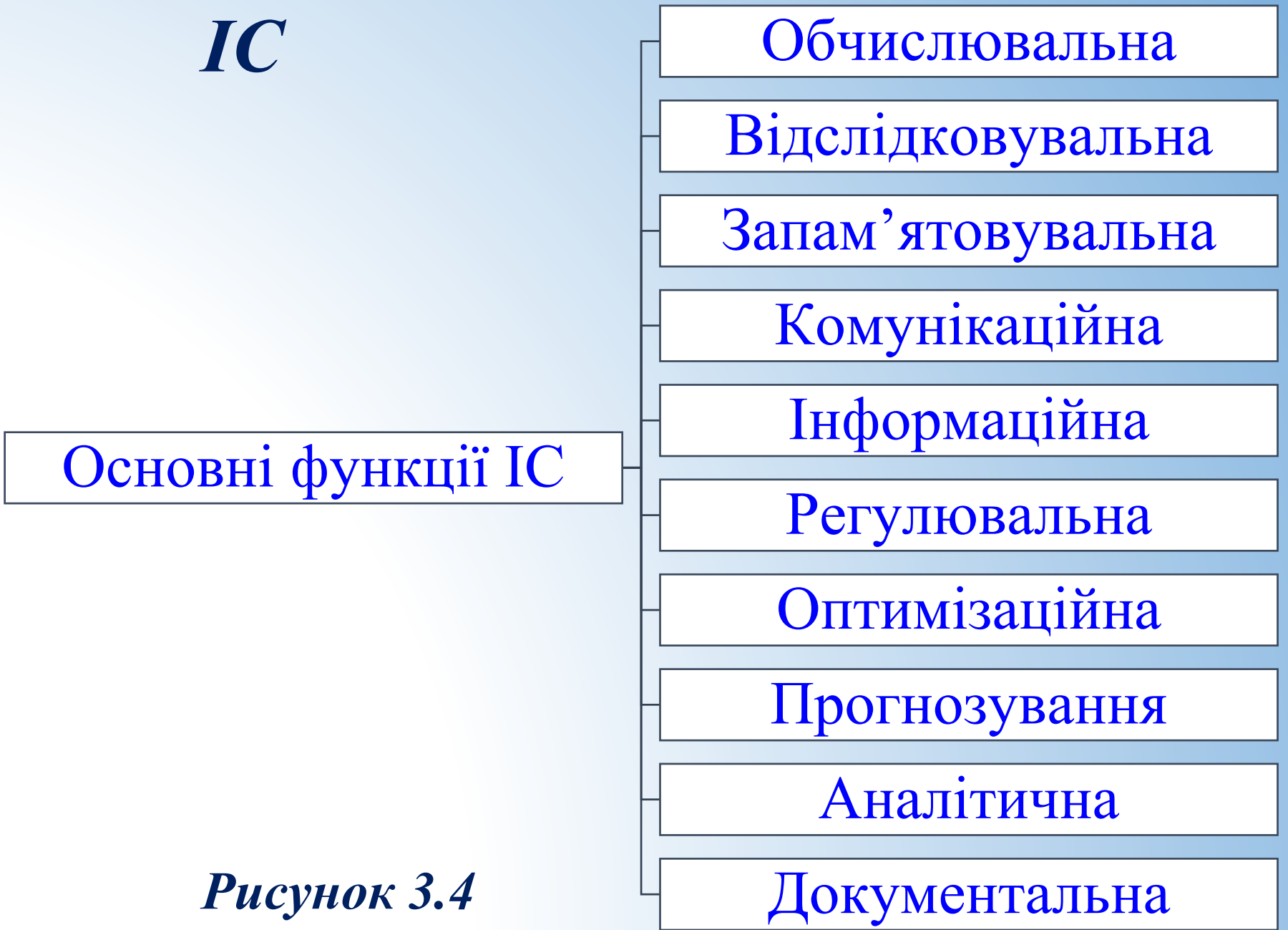


Рисунок 3.4

ІС повинні виконувати ряд функцій:

1. збір та реєстрація інформаційних ресурсів

Очищення даних – необхідна стадія попередньої обробки даних і підготовки їх до завантаження в систему.

Конвертація даних при введенні в систему використовується для перетворення даних з одного формату в інший.

2. зберігання інформаційних ресурсів

3. управління збереженими даними в системах БД

4. обробка інформаційних ресурсів

5. надання інформаційних ресурсів користувачам

Дві категорії користувачів інформаційних систем: кінцеві користувачі та прикладні програми.

6. інші функції

Ряд функцій покладається на персонал системи і на її ПЗ:

- управління розподіленими інформаційними ресурсами;
- захист фізичної цілісності інформаційних ресурсів і їх відновлення при руйнування;
- забезпечення інформаційної безпеки в системі;
- управління метаданими;
- адміністрування інформаційними ресурсами;
- забезпечення адаптації системи до змін вимог до неї і до змін в предметній області.

3.3 РЕГЛАМЕНТ ФУНКЦІОНУВАННЯ ІС

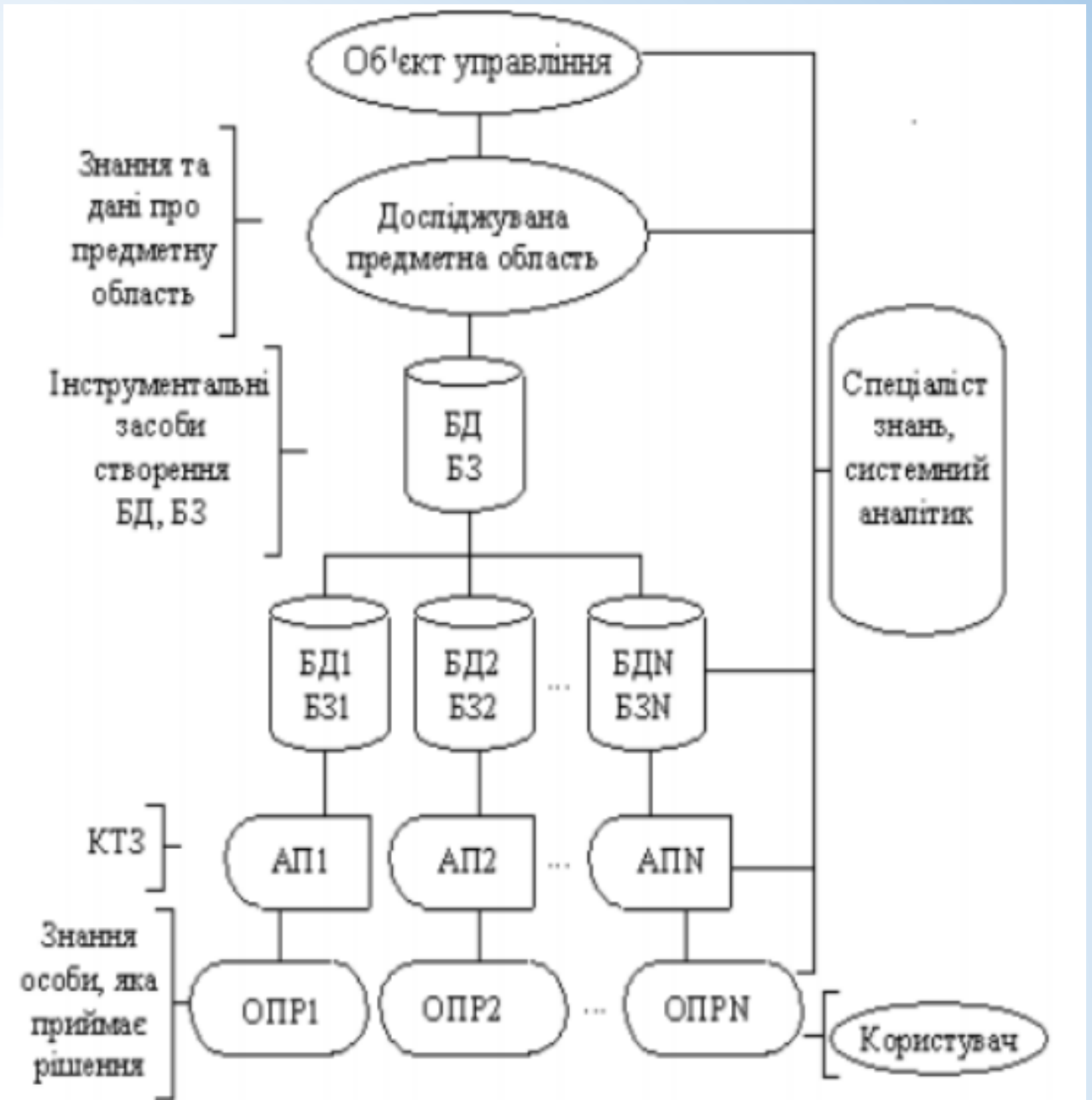


Рисунок 3.5 Схема взаємозв'язку при функціонуванні ІС

ТЕМА №4. СТРУКТУРНІ ОСОБЛИВОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ЇХ ЗАБЕЗПЕЧЕННЯ

Тематичний план

4.1 Аналіз видів структур ІС та основних їх компонентів

4.2 Огляд різновидів забезпечення ІС. Функціональний підхід до структури ІС.

4.1 АНАЛІЗ ВИДІВ СТРУКТУР ІС ТА ОСНОВНИХ ЇХ КОМПОНЕНТІВ

Структура системи – це представлення системи у вигляді груп елементів із зазначенням зв’язків між ними.

Структура системи за типами зв’язків:

- послідовне
- паралельне з’єднання елементів
- зворотний зв’язок

Ієрархія – це структура з наявністю підпорядкованості, тобто нерівноправних зв’язків між елементами.

Рисунок 4.1

Компонент – це частина ІС, яку після декомпозиції можна розглядати як самостійне ціле.

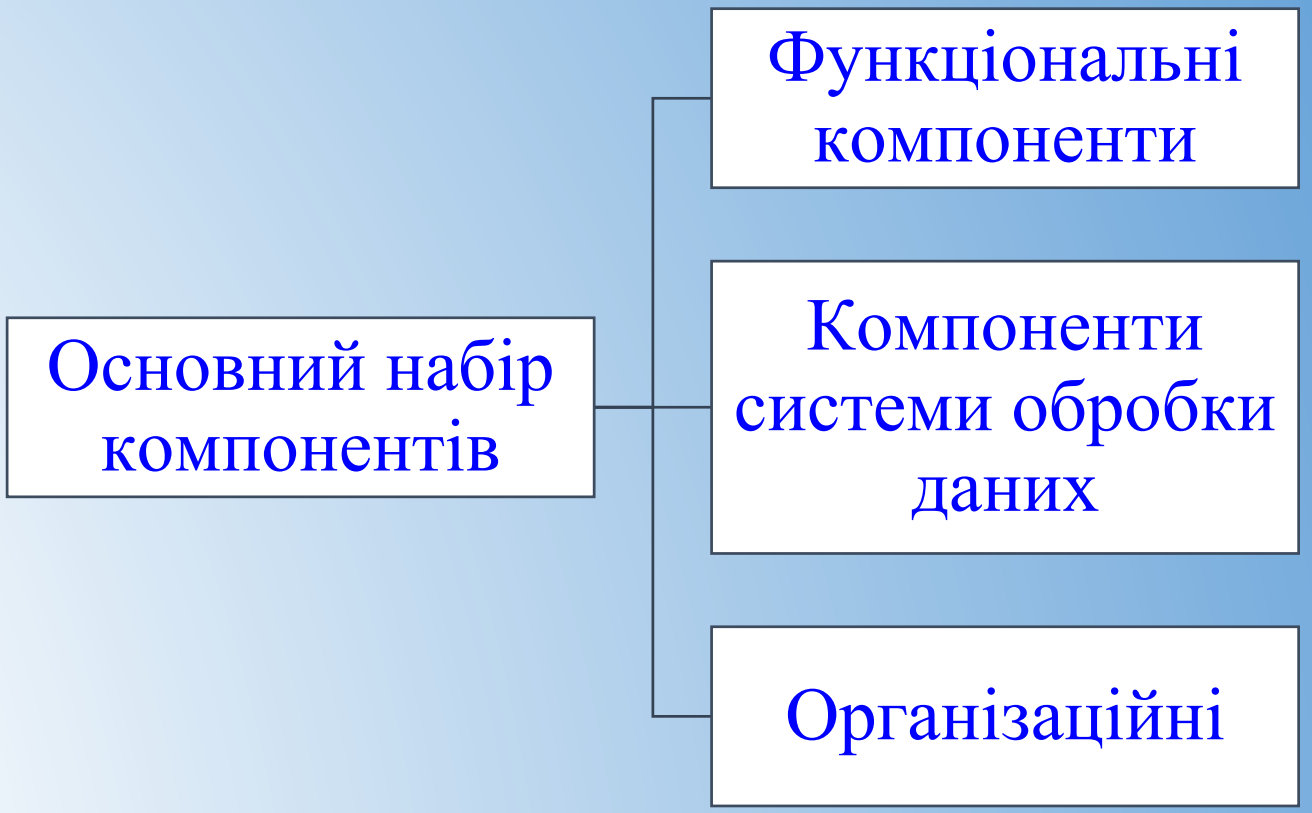


Рисунок 4.2

Типові функціональні компоненти ІС

Позначення	Найменування	Характеристика
PS	Presentation Services (засоби представлення)	Обслуговує введення даних і запитів від користувача та відображає те, що повідомляє йому компонент логіки представлення (PL) з використанням відповідної програмної підтримки.
PL	Presentation Logic (логіка представлення)	Управляє взаємодією між користувачем і системою; обробляє дії користувача під час вибору команди в меню, при виборі пункту зі списку тощо.
BL	Business Logic (прикладна логіка)	Реалізує набір правил для прийняття рішень, обчислень і операцій, які повинен виконати програмний додаток.

Позначення	Найменування	Характеристика
DL	Data Logic (логіка управління даними)	Операції з БД за допомогою мови SQL, які потрібно виконати для реалізації прикладної логіки управління даними.
DS	Data Services (операції з БД)	Підтримує дії СУБД, що реалізують логіку управління даними (маніпулювання даними, визначення даних, фіксацію або відкат транзакцій тощо).
FS	File Services (файлові операції)	Реалізує дискові операції читання і запису даних для СУБД та ін. компонентів. Зазвичай є функціями операційної системи (ОС).

4.2 ОГЛЯД РІЗНОВИДІВ ЗАБЕЗПЕЧЕННЯ ІС. ФУНКЦІОНАЛЬНИЙ ПІДХІД ДО СТРУКТУРИ ІС

Рисунок 4.3

Правове забезпечення включає:

- статус ІС у конкретній сфері управління;
- правове положення про компетенцію ланок ІС, організацію її діяльності;
- порядок створення і використання інформації в ІС;
- порядок одержання і використання обчислювальних і технічних засобів в ІС;
- порядок створення і використання математичного та програмного забезпечення;
- організацію процесу управління в ІС;
- права, обов’язки і відповідальність персоналу ІС;
- правове регулювання процесів створення ІС.

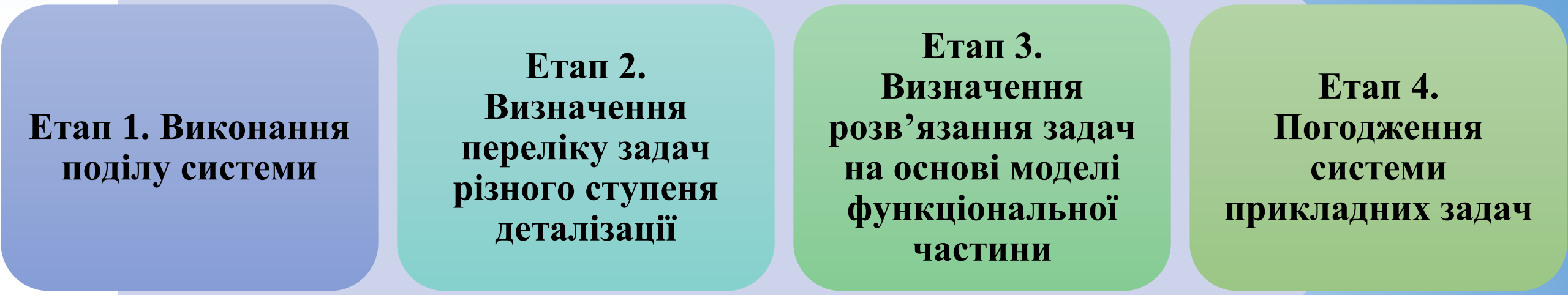


Рисунок 4.4 Етапи визначення структури і поділу ІС

Задача ІС – функція чи частина функції ІС, що являє собою формалізовану сукупність автоматичних дій, виконання яких приводить до результату заданого виду.

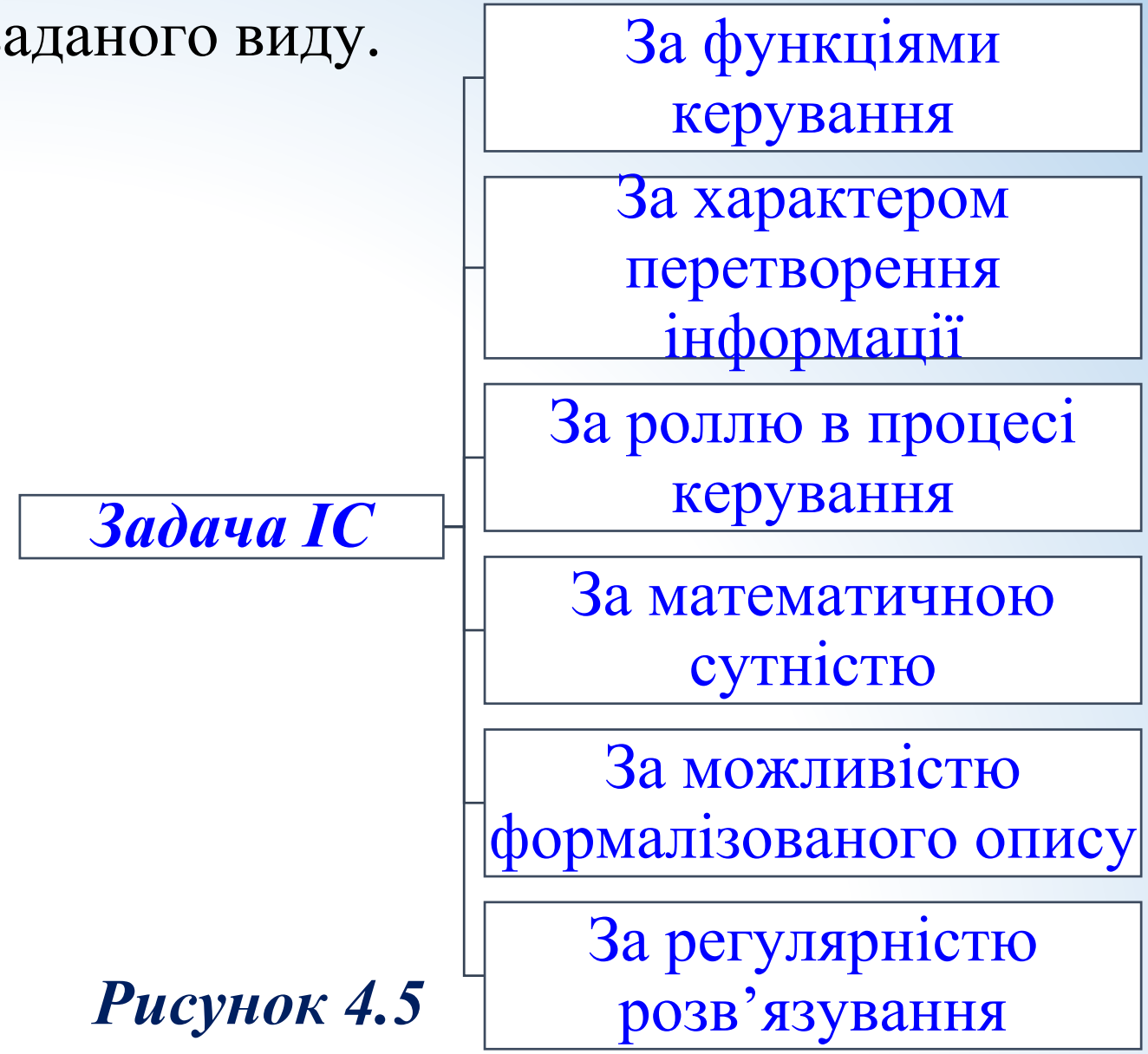


Рисунок 4.5

Макропроєктування – це моделювання мети, призначення, побудови, функціонування, розвитку та інших несистемних атрибутів ІС і її складових частин.

Мікропроєктування – це перетворення перелічених моделей у кінцеві проєктні рішення, що впроваджуються на об'єкті.

ТЕМА №5. АРХІТЕКТУРА ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

5.1 Визначення сутності поняття «архітектура інформаційної системи»

5.2 Типова класифікація архітектур інформаційних систем

5.3 Аналіз архітектурного підходу до проєктування інформаційних систем

5.4 Огляд платформеної архітектури інформаційних систем

5.1 ВИЗНАЧЕННЯ СУТНОСТІ ПОНЯТТЯ «АРХІТЕКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ»

Рисунок 5.1

Термін «архітектура» застосовувався у проєктуванні та при побудові різних споруд та визначав їх структуру, взаємозв'язки між складовими частинами, базові принципи організації і подальшого розвитку.

Архітектура системи (стандарт ANSI/IEEE 1471-2000) – це опис організації системи в термінах компонентів, їх взаємозв'язків між собою і з довкіллям, принципи управління їх розробкою і розвитком.

Архітектура є ***багатовимірною.***

Архітектура ПЗ передбачає ***різні подання, які служать різним цілям:***

- зображенню функціональних можливостей системи;
- відображенню логічної організації системи;
- опису фізичної структури програмних компонентів в середовищі реалізації;
- відображенню структури потоків управління та аспектів паралельної роботи;
- опису фізичного розміщення програмних компонентів на базовій платформі.

Зображення архітектури – це спрощений опис системи з конкретної точки зору.

Архітектурно-значущий елемент – це елемент, який має значний вплив на структуру системи та її продуктивність, надійність і можливість розвитку.

Розробка моделі архітектури системи ПЗ необхідна на стадії, що передує його реалізації або оновленню.

Процедура вибору архітектури для проєктованої ІС зводиться до визначення вартості володіння нею.

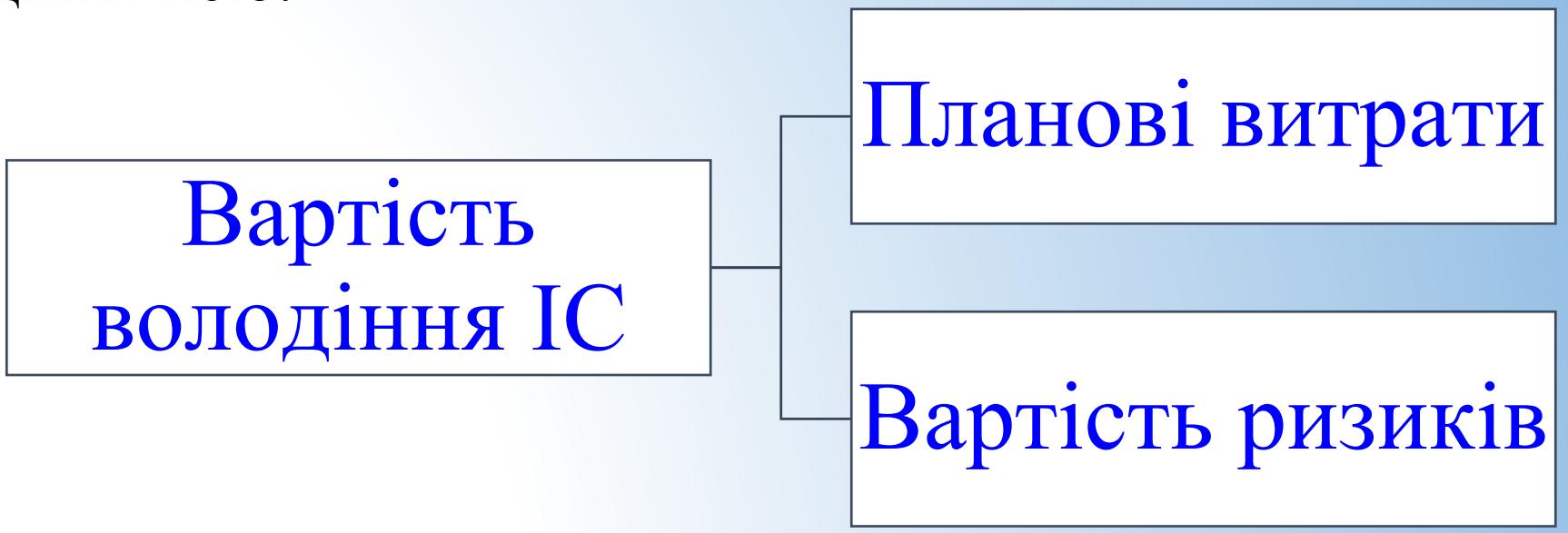


Рисунок 5.2

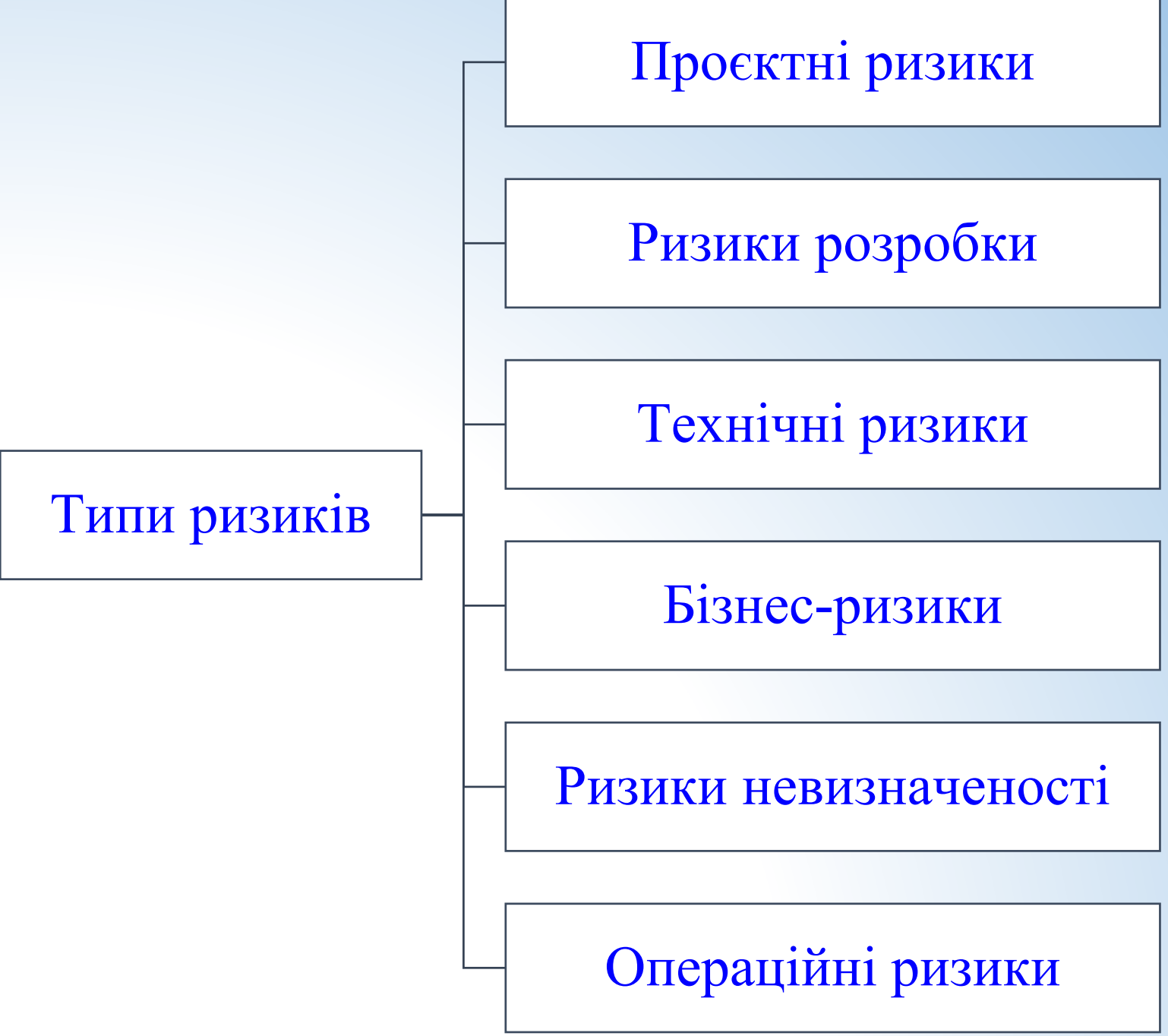


Рисунок 5.3

Концепція архітектури ІС повинна формуватися на етапі техніко-економічного обґрунтування і вибиратися такою, щоб вартість володіння нею була мінімальною.

5.2 ТИПОВА КЛАСИФІКАЦІЯ АРХІТЕКТУР ІНФОРМАЦІЙНИХ СИСТЕМ

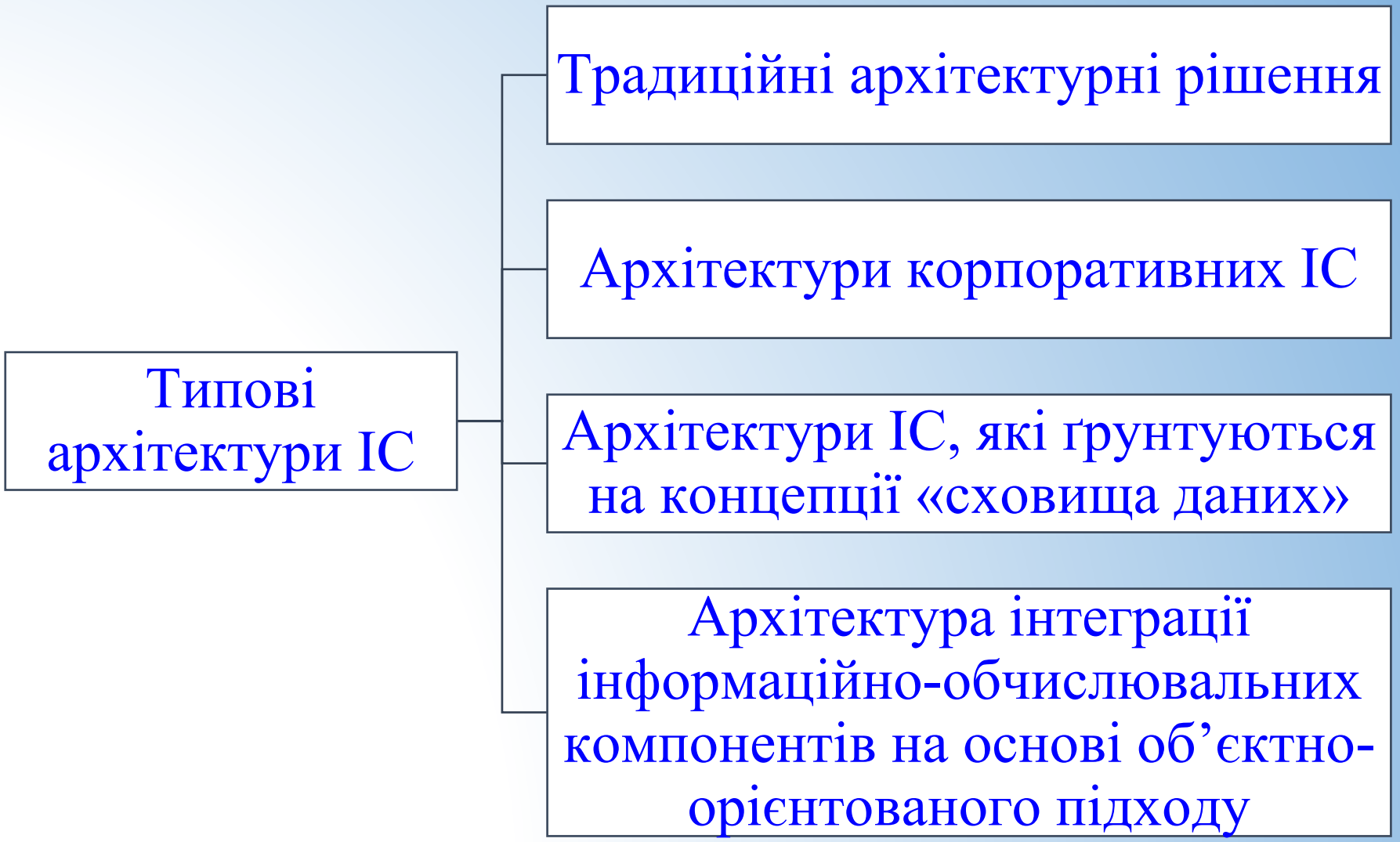


Рисунок 5.4

Індустрія розробки автоматизованих ІС управління зародилася в ***1950 – 1960-х*** роках і до кінця XX ст. набула певної форми.

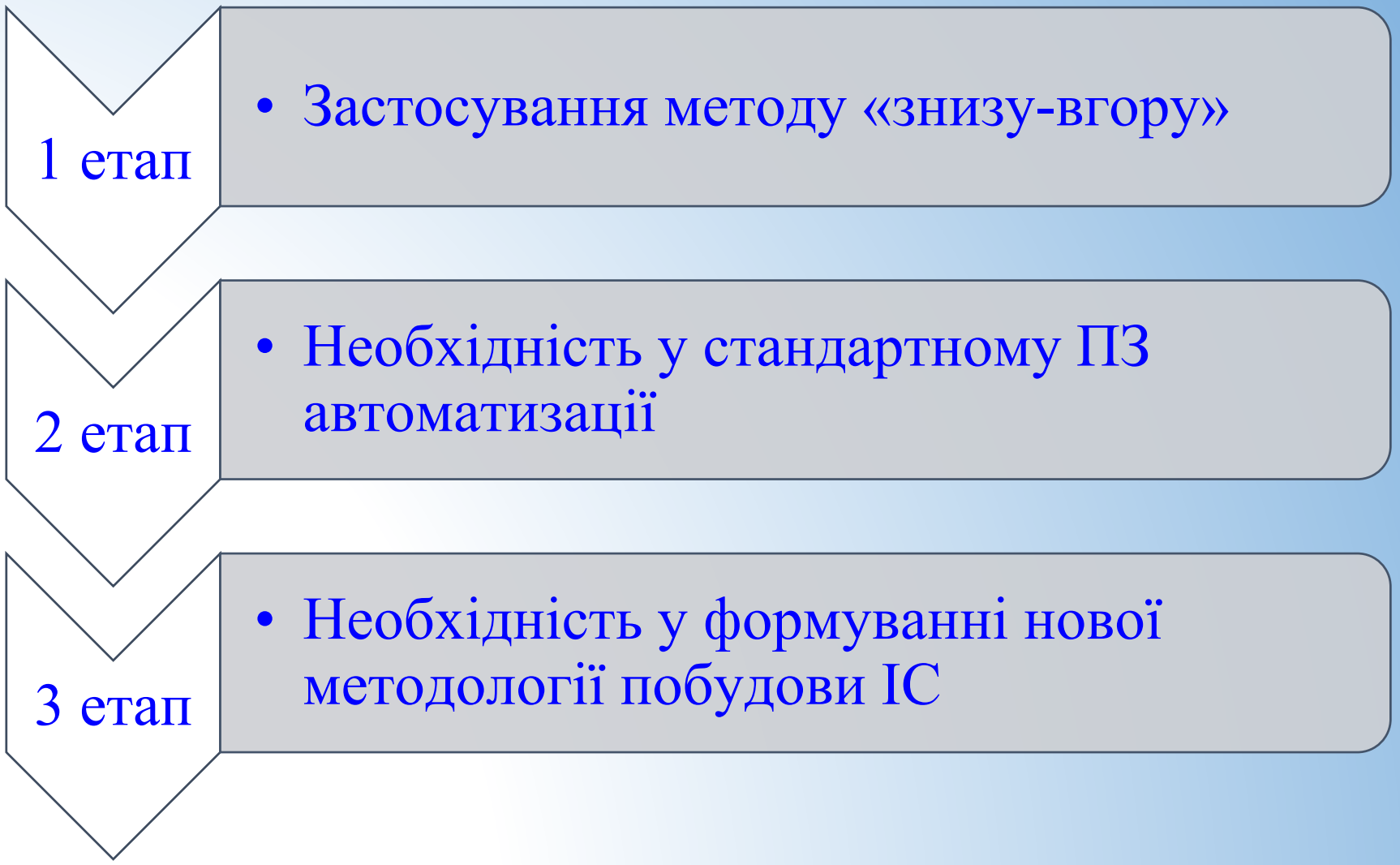


Рисунок 5.5

Основні завдання, вирішенню яких має сприяти методологія проєктування корпоративних ІС:

- забезпечувати створення корпоративних ІС, які відповідають цілям і задачам організації, вимогам до автоматизації ділових процесів замовника;
- гарантувати створення системи із заданою якістю та в задані терміни, в рамках встановленого бюджету проєкту;
- підтримувати зручну дисципліну супроводу, модифікації і нарощування системи;
- забезпечувати спадкоємність розробки.

Впровадження методології повинно призводити до зниження складності процесу створення ІС за рахунок:

- повного і точного опису цього процесу;
- застосування сучасних методів і технологій створення ІС на всьому життєвому циклі ІС.


Бізнес-стратегії та бізнес-процеси
Засоби реалізації та інтеграції бізнес-застосунків
Сховища даних і СУБД
Програмні застосунки
Апаратні засоби
5
4
3
2
1
Бізнес-архітектура
IT-архітектура
Архітектура даних
Програмна архітектура
Технічна архітектура

Рисунок 5.6

5.3 АНАЛІЗ АРХІТЕКТУРНОГО ПІДХОДУ ДО ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

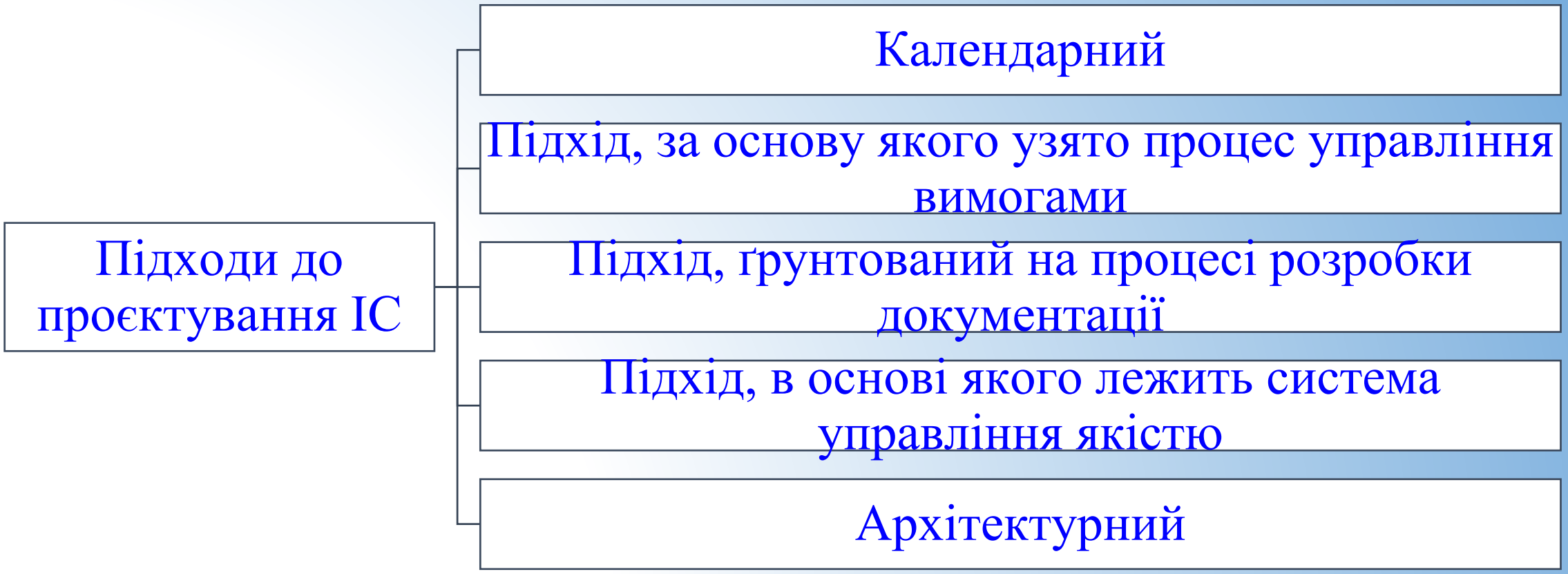


Рисунок 5.7

Архітектурний підхід до проєктування ІС:

- найбільш зрілий;
- ключовий аспект – це створення фреймворка, адаптацію якого під потреби конкретної системи легко здійснити;
- завдання проєктування розбивається на дві підзадачі:
  - ✓ розробка багаторазово використовуваного каркаса;
  - ✓ створення системи на його основі;
- підзадачі можуть вирішуватися різними групами фахівців;
- при використанні каркасів з’являється можливість швидко змінювати функціональність системи за рахунок ітераційного характеру процесу проєктування.

5.4 ОГЛЯД ПЛАТФОРМЕНОЇ АРХІТЕКТУРИ ІНФОРМАЦІЙНИХ СИСТЕМ

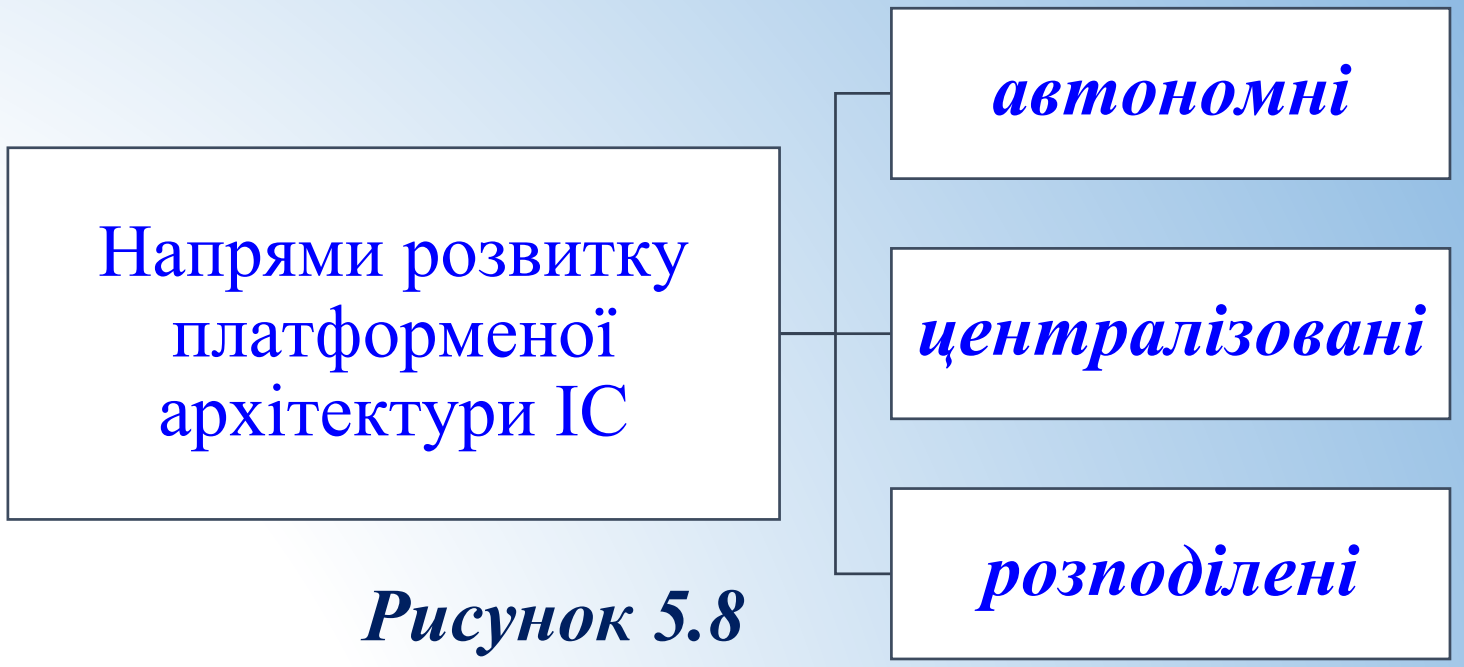


Рисунок 5.8

Автономна архітектура передбачає наявність усіх функціональних компонентів системи на одному фізичному пристрої і не повинна мати зв’язків із зовнішнім середовищем (системні утиліти, текстові редактори та корпоративні програми).

Централізована архітектура передбачає виконання усіх необхідних завдань на спеціально відведеному вузлі, потужності якого вистачає, щоб задовольнити потреби усіх користувачів.

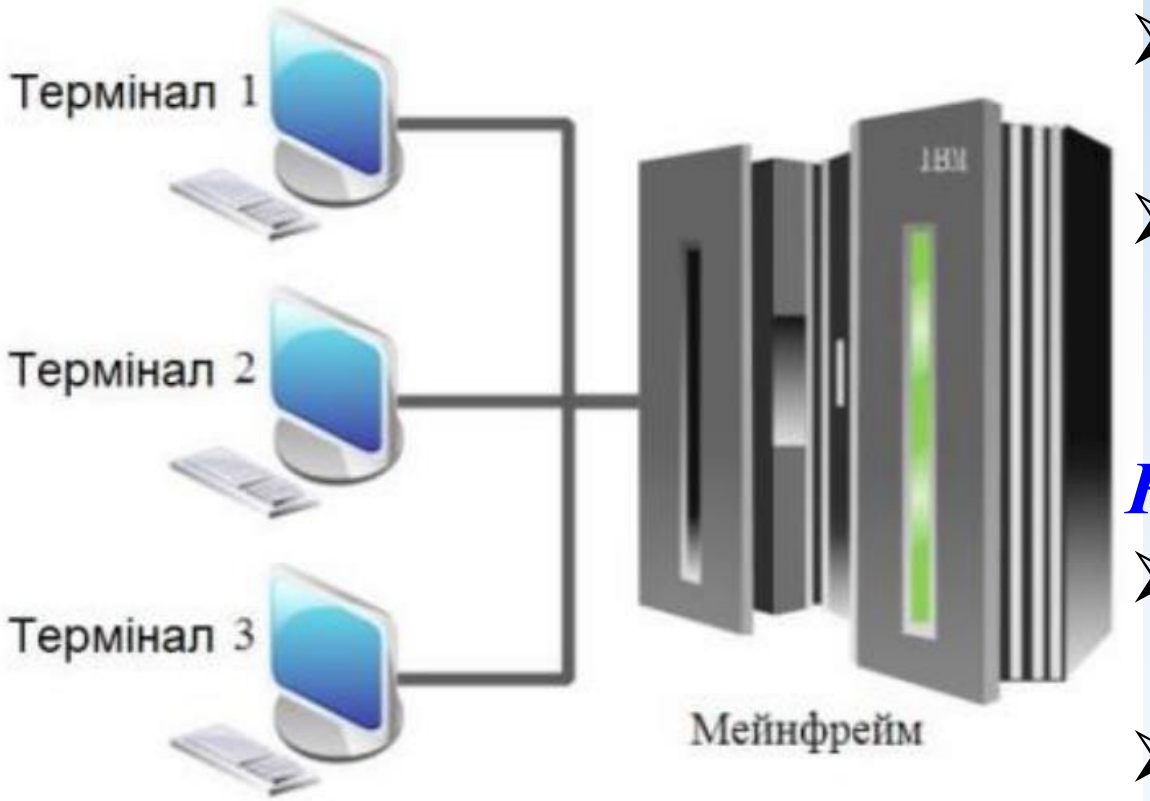


Рисунок 5.9

Переваги:

- відсутність необхідності адміністрування робочих місць;
- легкість обслуговування та експлуатації системи, оскільки усі ресурси зосереджено в одному місці

Недоліки:

- функціонування усієї системи повністю залежить від головного вузла;
- всі ресурси та програмні засоби є колективними і не можуть бути змінені під потреби конкретних користувачів

Розподілена архітектура – функціональні компоненти ІС розподіляються за наявними вузлами залежно від поставленнях цілей і завдань.

Основні характеристики систем розподіленої архітектури:

- спільне використання ресурсів
- відкритість
- паралельність
- масштабованість
- відмовостійкість

Недоліки:

- структурна складність;
- складність у забезпеченні достатнього рівня безпеки;
- велика кількість зусиль на управління системою;
- непередбачувана реакція на зміни.

Архітектура «файл-сервер» передбачає наявність виділеного мережевого ресурсу для зберігання даних («файлового серверу»).

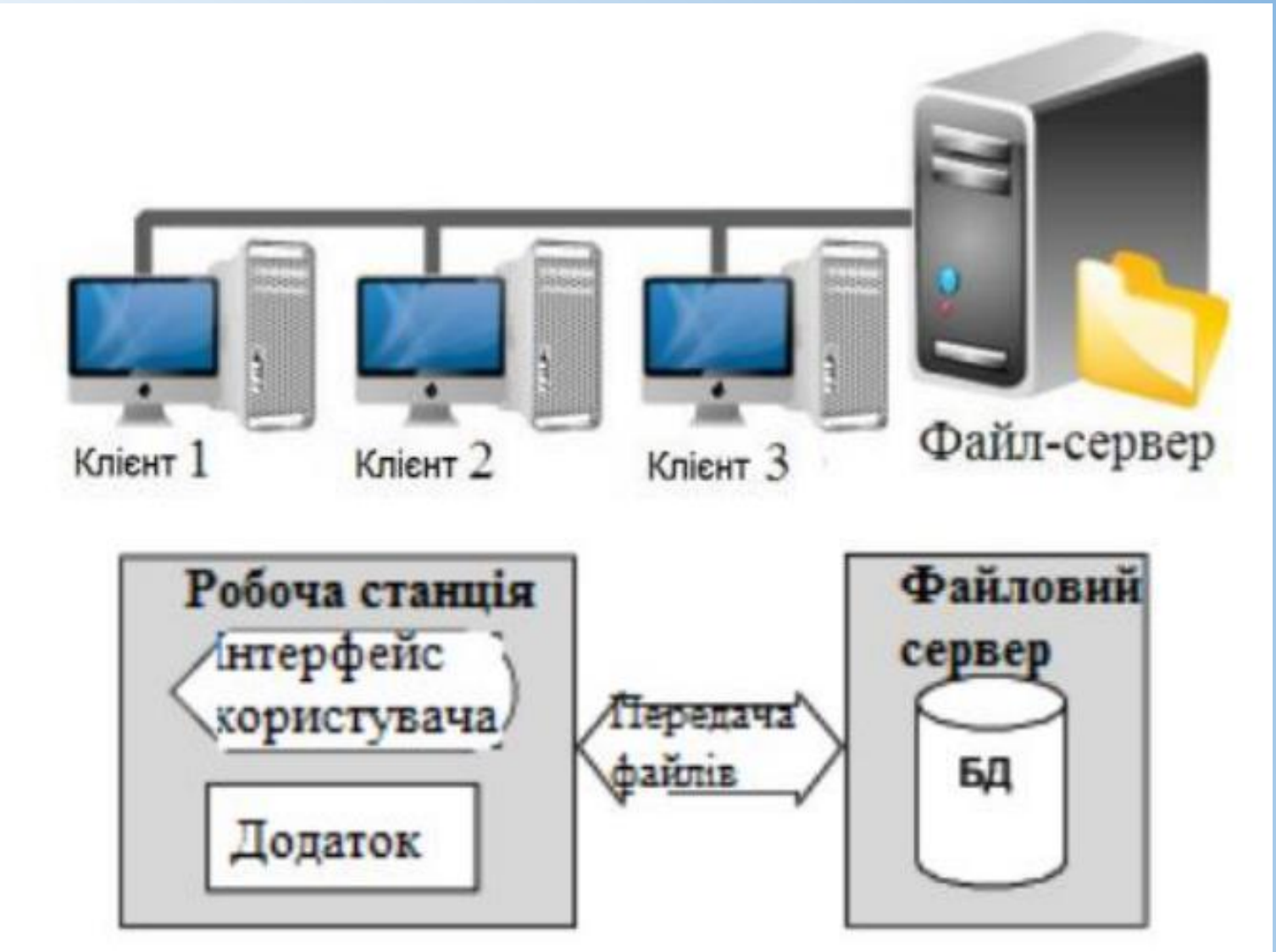


Рисунок 5.10

Архітектура «клієнт-сервер» – є мережевою інфраструктурою, в якій сервери є постачальниками певних сервісів, а комп'ютери клієнтів виступають їх споживачами (дворівнева (рис. 5.11) та багаторівнева (рис. 5.12)).

Рисунок 5.11

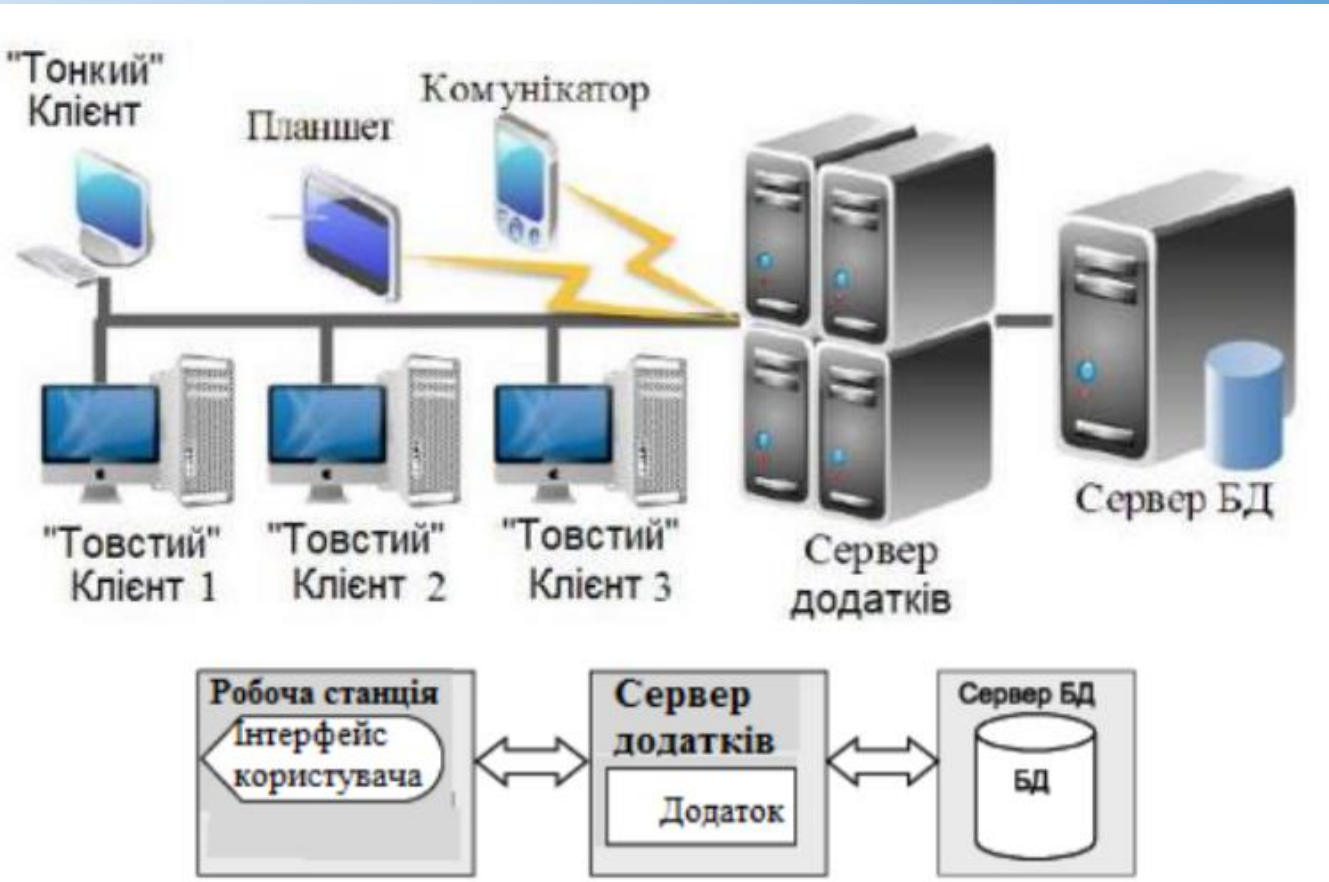


Рисунок 5.12

Архітектура Web-застосунків або архітектура Web-сервісів – передбачає надання деякого сервісу, доступного в мережі Internet, через спеціальне застосування.

Основою для надання цих послуг служать такі відкриті стандарти і протоколи:

- SOAP (Simple Object Access Protocol)
- WSDL (Web Service Description Language)
- UDDI (Universal Description, Discovery and Integration)

Технології для побудови розподіленої архітектури Web-сервісу:

- EJB (Enterprise JavaBeans);
- DCOM (Distributed Component Object Model);
- CORBA (The Common Object Request Broker Architecture).

ТЕМА №6. АНАЛІЗ ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

6.1 Визначення основних фаз проєктування інформаційних систем

6.2 Аналіз процесів життєвого циклу інформаційних систем

6.3 Основна структура життєвого циклу інформаційних систем

6.4 Огляд моделей життєвого циклу інформаційних систем. Визначення їх особливостей та переваг і недоліків

6.1 ВИЗНАЧЕННЯ ОСНОВНИХ ФАЗ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Рисунок 6.1

Типові помилки, до яких вдаються розробники на початкових стадіях проєкту:

- помилки у визначенні інтересів замовника;
- концентрація на другорядних інтересах;
- хибна інтерпретація початкового завдання;
- невірне або недостатнє розуміння деталей;
- неповнота функціональних специфікацій;
- вади у визначенні необхідних ресурсів і термінів виконання робіт;
- недостатня перевірка на узгодженість етапів, відсутність контролю з боку замовника

6.2 АНАЛІЗ ПРОЦЕСІВ ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНИХ СИСТЕМ

Життєвий цикл – це безперервний процес, що починається з моменту рішення про створення ІС і закінчується після вилучення її з експлуатації.

У відповідності до міжнародного стандарту ISO/IEC 12207 ***структура життєвого циклу має три групи процесів:***

- головні процеси
- допоміжні процеси
- організаційні процеси


Основні процеси ЖЦ ІС
Розробка – поєднує всі роботи щодо створення інформаційного та програмного забезпечення та його компонентів відповідно до вимог техзавдання
Експлуатація
Супроводження (технічне обслуговування)
Головні роботи (експлуатація, локалізація проблем та усунення причин, модифікація ПЗ, підготовка пропозицій щодо удосконалення ПЗ, розвиток та подальша модернізація)
Підготовчі роботи (конфігурація БД і робочих місць користувачів, забезпечення експлуатаційною документацією, навчання персоналу)

Рисунок 6.2

Допоміжні процеси здійснюють процес ***управління конфігурацією***, який підтримує головні етапи життєвого циклу ІС, зокрема – процеси розробки та супроводження.

Організаційні процеси передбачають управління проєктом, яке пов'язано з:

- плануванням і організацією робіт
- створенням колективу проєктувальників
- контролем за термінами та якістю реалізації всіх етапів розробки ІС

Технічне й організаційне забезпечення проєкту передбачає:

- вибір методів та інструментальних засобів для реалізації проєкту;
- визначення методів щодо опису всіх проміжних станів розробки
- розробку методів і засобів для випробувань ПЗ
- навчання та підвищення кваліфікації персоналу

Верифікація – процес визначення відповідності поточного стану розробки вимогам до цього етапу.

Перевірка – це процес, спрямований на визначення відповідності параметрів ІС вихідним вимогам.

Тестування – це процес, спрямований на визначення розбіжностей між дійсними та очікуваними результатами проєкту та оцінкою відповідності характеристик ІС вихідним вимогам.

6.3 ОСНОВНА СТРУКТУРА ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНИХ СИСТЕМ

Рисунок 6.3

Фактори впливу на трудомісткість стадій створення ІС:

- складність та специфіка процесу, що автоматизується;
- наявність відповідних розробок;
- ступінь автоматизації проєктних робіт на кожній із стадій;
- кваліфікація виконавців;
- готовність об’єкта до впровадження ІС;
- метод проєктування.

6.4 ОГЛЯД МОДЕЛЕЙ ЖИТТЄВОГО ЦИКЛУ ІНФОРМАЦІЙНИХ СИСТЕМ. ВИЗНАЧЕННЯ ЇХ ОСОБЛИВОСТЕЙ ТА ПЕРЕВАГ І НЕДОЛІКІВ

Модель життєвого циклу ІС – це певна структура, що визначає послідовність та взаємозв'язки між процесами, що передбачені у межах ЖЦ ІС.

Стандарт ISO/IEC 12207 описує структури процесів, без деталізації методів і дій для вирішення завдань, що входять до процесів ЖЦ ІС.

Складові компоненти моделі життєвого циклу ІС:

- стадії;
- основні результати виконання робіт на кожній стадії;
- ключові події.

Найбільш поширені ***моделі життєвого циклу ІС***:

- каскадна (класична або водоспадна) модель;
- ітераційна модель;
- спіральна модель

Каскадна (класична, водоспадна) модель ЖЦ ІС:

- запропонована в 1970 р. Уинстоном Ройсом;
- передбачає послідовне виконання всіх етапів проєкту в строго фіксованому порядку;
- перехід на наступний етап здійснюється після повного закінчення робіт попереднього етапу;
- кожний етап завершується оформленням повного комплекту робочої документації

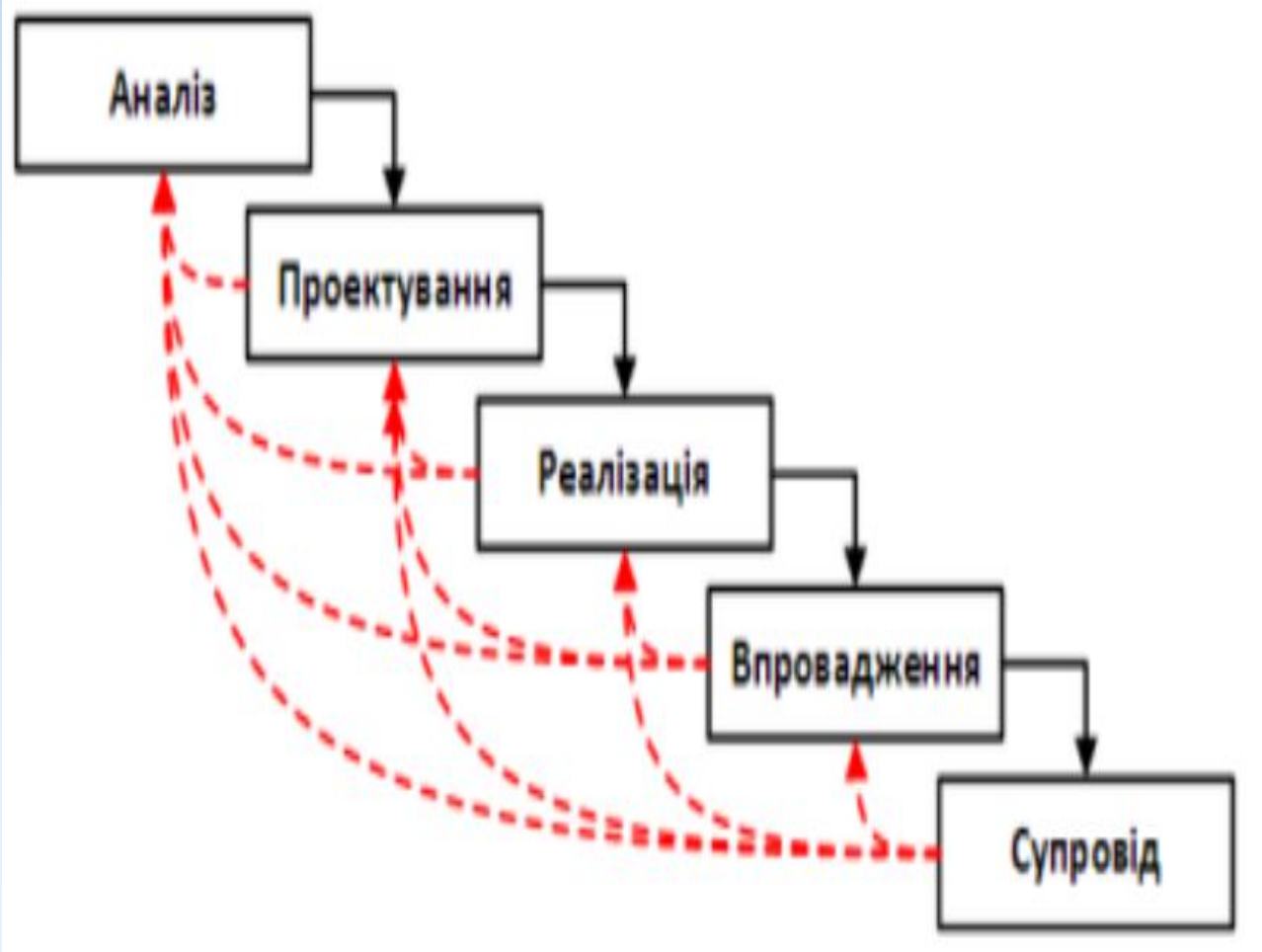


Рисунок 6.4

Ітераційна модель ЖЦ ІС (поетапна модель з проміжним контролем):

- розробка ІС ведеться ітераціями з циклами зворотного зв’язку між етапами;
- наявні міжетапні коригування, що дозволяють враховувати реально існуючий взаємовплив результатів розробки на різних етапах;
- час життя кожного з етапів розтягується на весь період розробки;
- трудомісткість робіт і тимчасові витрати істотно скорочуються в порівнянні з водоспадною моделлю ЖЦ ІС.

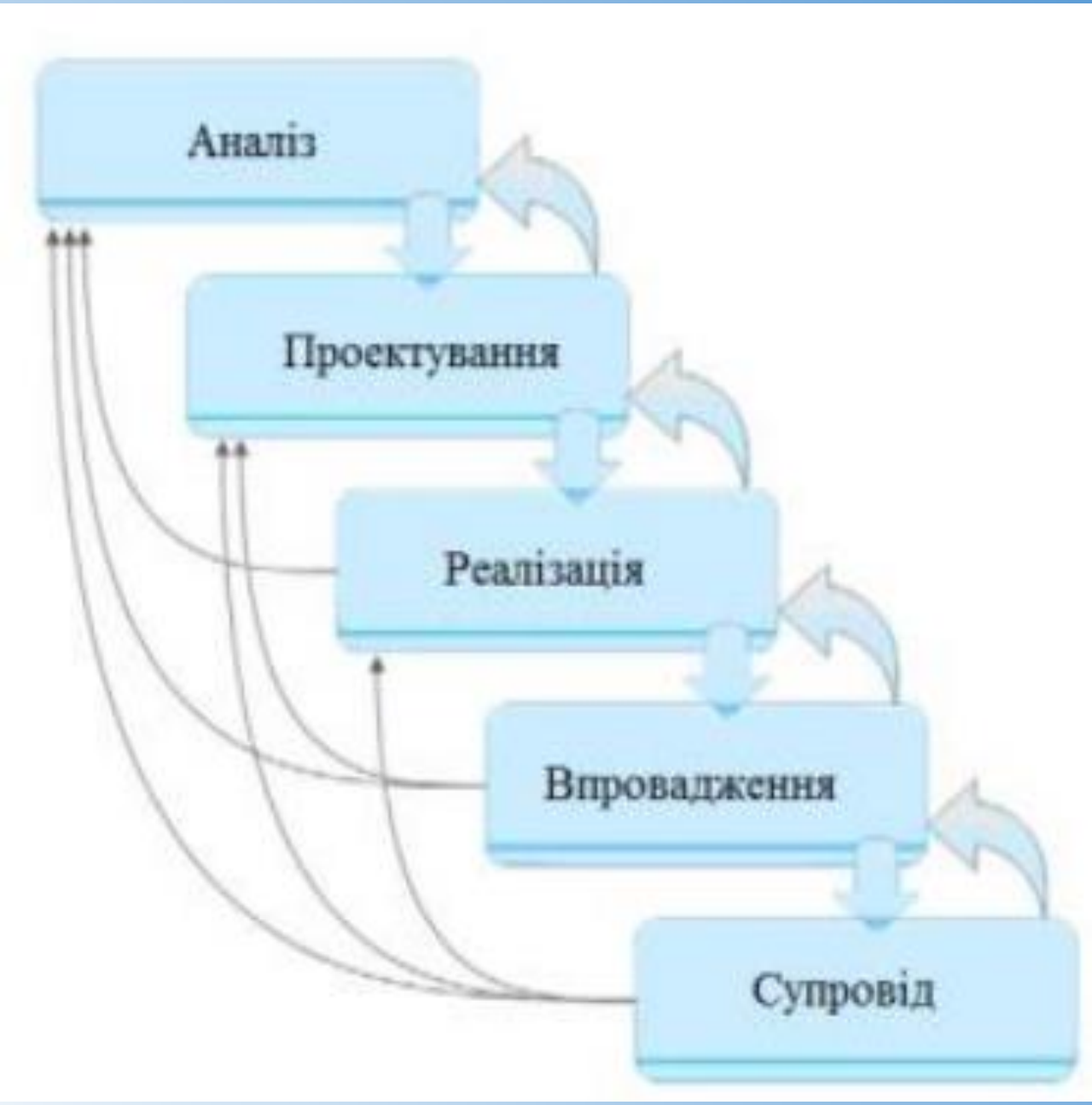


Рисунок 6.5

Спіральна модель ЖЦ ІС:

- запропонована Баррі Боем в 1988 р.;
- розроблена для подолання основних проблем каскадної моделі
- передбачає ітераційний процес розробки ІС;
- кожна ітерація є завершеним циклом розробки, що призводить до випуску діючої версії виробу.

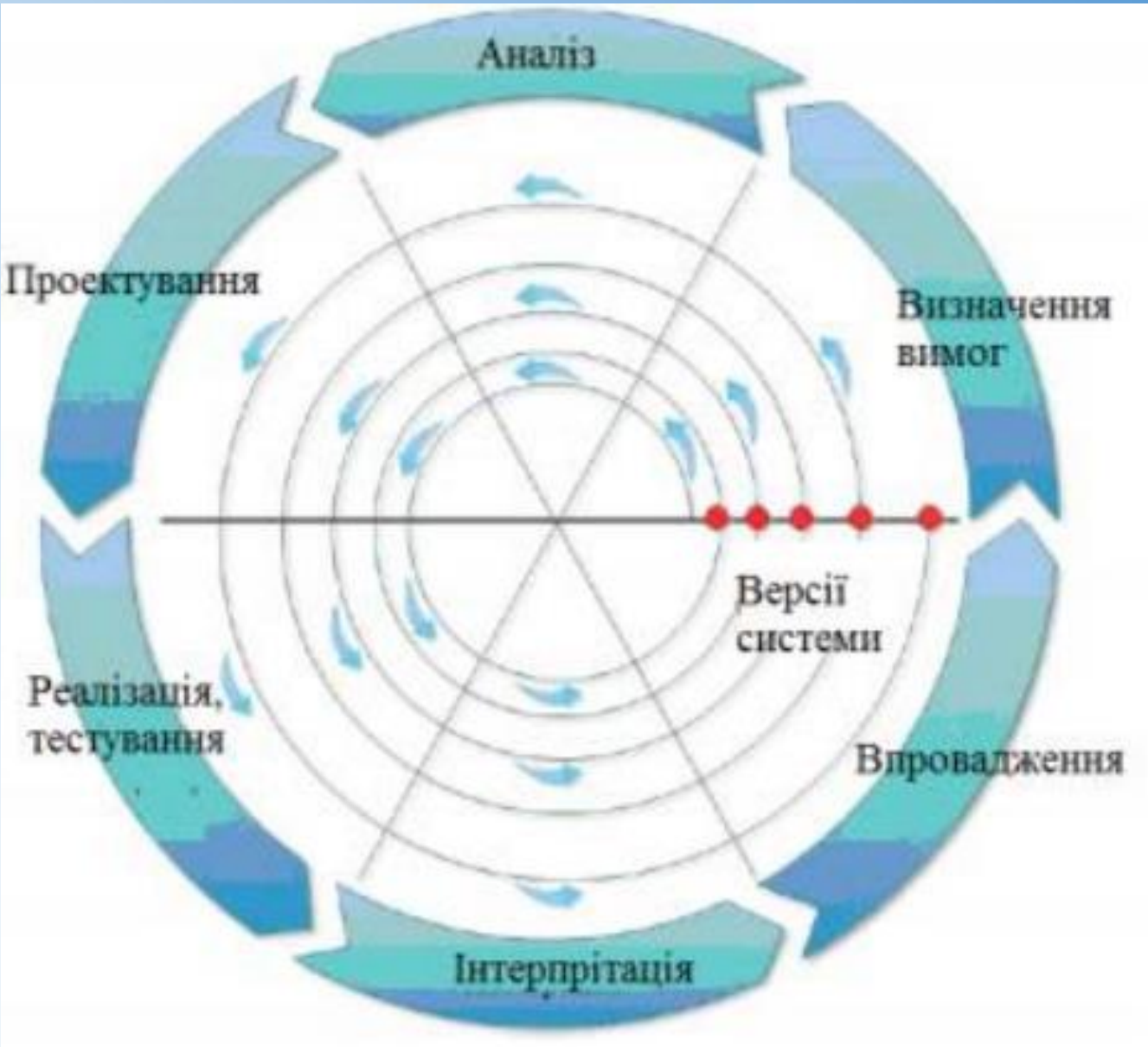


Рисунок 6.6

ТЕМА №7. ОГЛЯД ТЕХНОЛОГІЇ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

7.1 Визначення напряму проєктування інформаційних систем

7.2 Огляд основних елементів технології проєктування інформаційних систем

7.3 Застосування структурного підходу до проєктування інформаційних систем

7.1 ВИЗНАЧЕННЯ НАПРЯМУ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Потреба у створенні ІС обумовлюється необхідністю автоматизації чи модернізації існуючих інформаційних процесів або необхідністю корінної реорганізації в діяльності організації.

Потреба у створенні ІС вказує на:

- досягнення мети
- момент часу доцільного здійснення розробки
- витрати під час проєктування

Мета проєктування ІС – створити проєкт ІС, який становить технічна документація з докладним описом усіх проєктних рішень щодо створення та експлуатації ІС.

Процес проєктування ІС – це процес ухвалення проєктно-конструкторських рішень, спрямованих на одержання опису системи, що задовольняє вимоги замовника.

Проєкт ІС – це проєктно-конструкторська і технологічна документація, в якій наведений опис проєктних рішень щодо створення й експлуатації ІС у конкретному програмно-технічному середовищі.

Проєктування ІС – процес перетворення вхідної інформації про об’єкт проєктування, про методи проєктування і про досвід проєктування об’єктів аналогічного призначення в проєкт ІС відповідно до державних стандартів.

Об’єкт проєктування ІС – окремі елементи або комплекси їх функціональних і забезпечувальних частин.

Функціональні елементи – це завдання, комплекси завдань і функції управління.

Забезпечувальні частини ІС – об’єктами проєктування виступають елементи і комплекси інформаційного, програмного та технічного забезпечення системи.

Предметна область проєктування ІС – система організаційно-економічного управління діяльністю підприємства.

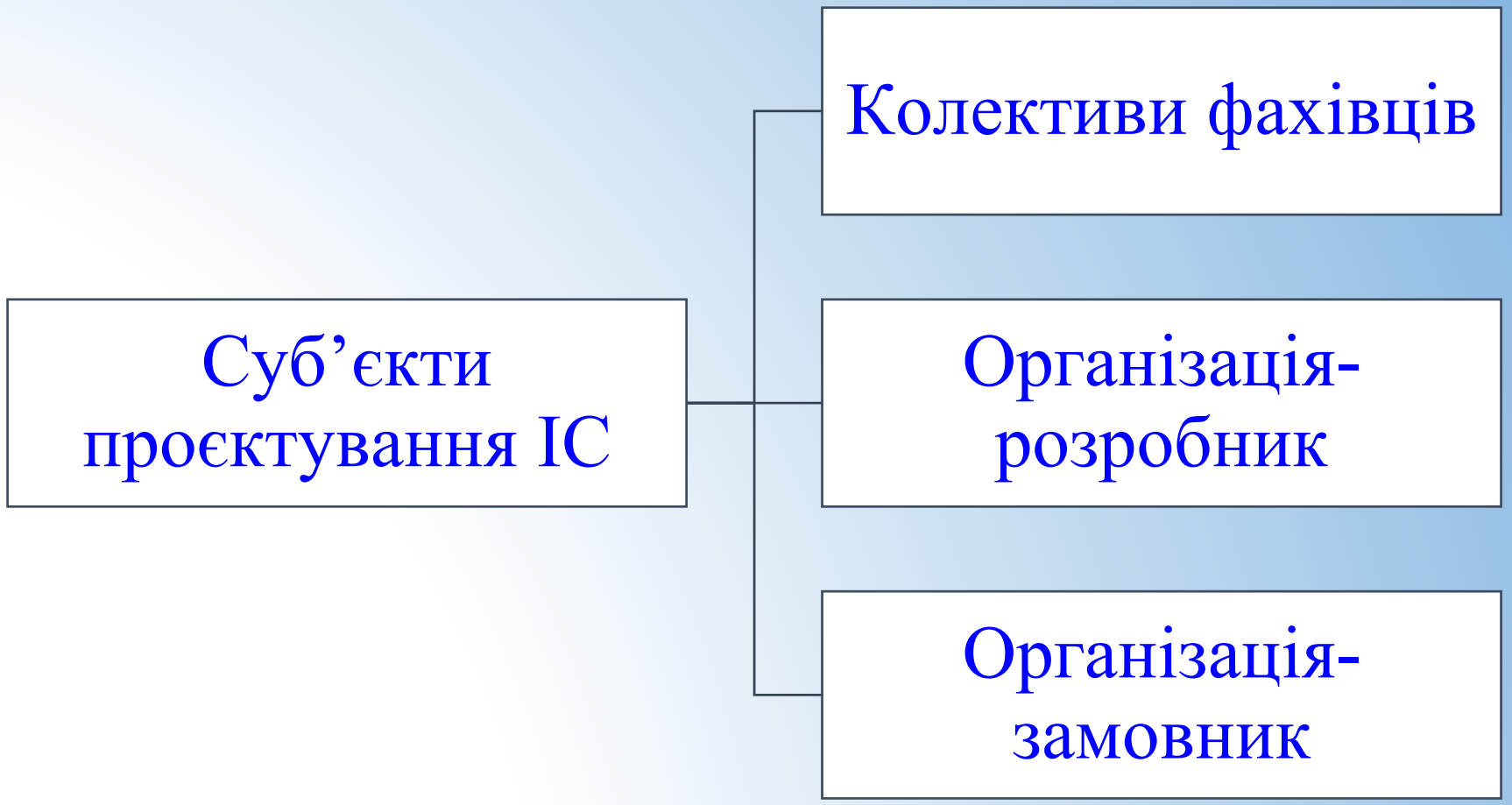


Рисунок 7.1

7.2 ОГЛЯД ОСНОВНИХ ЕЛЕМЕНТІВ ТЕХНОЛОГІЇ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Технологія проєктування ІС – це сукупність методів і засобів проєктування ІС, використовуваних організаційних прийомів і технічних засобів, орієнтованих на створення чи модернізацію проєкту ІС.

Рисунок 7.2

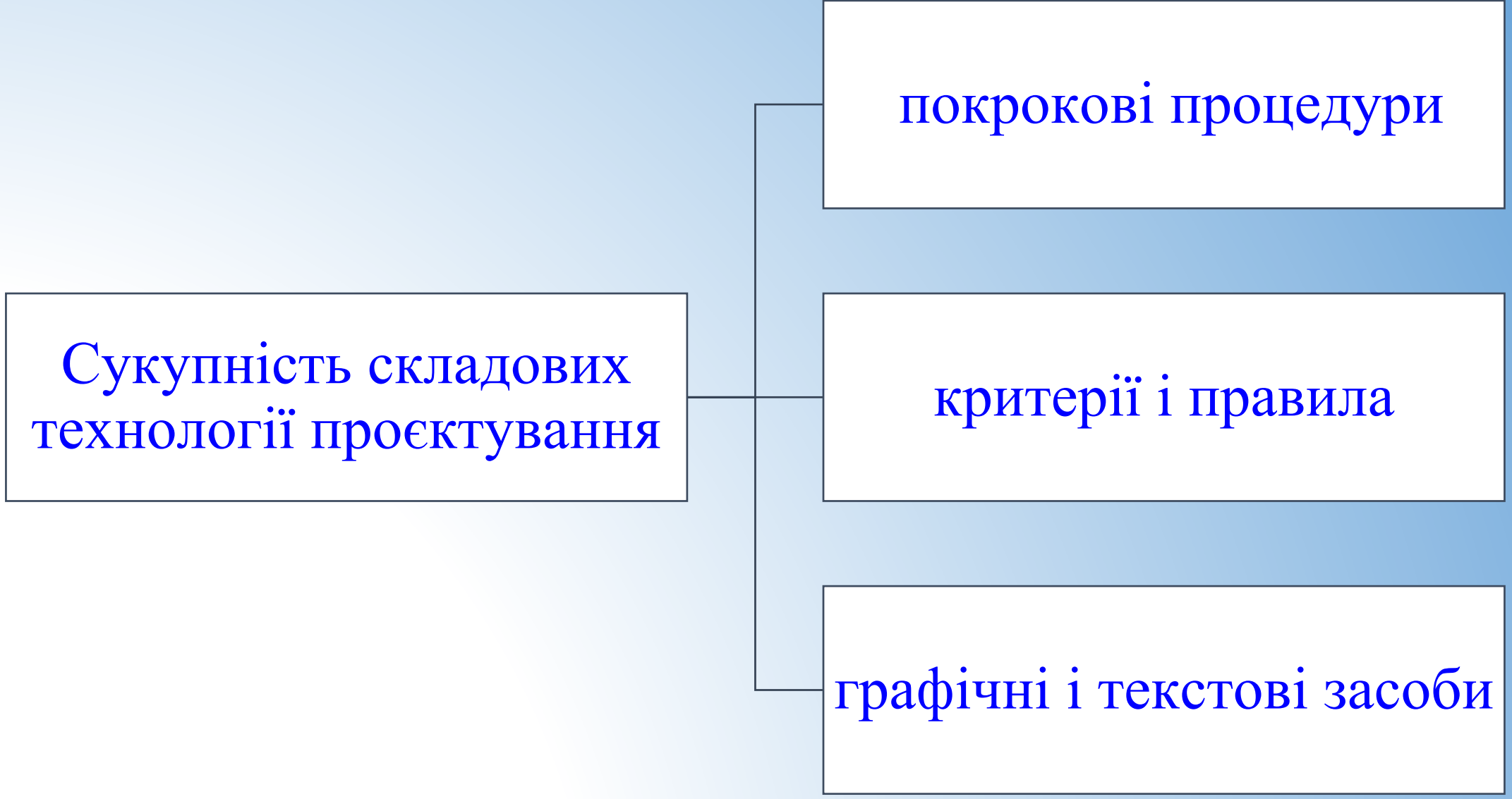

Сукупність складових технології проєктування
покрокові процедури
критерії і правила
графічні і текстові засоби

Рисунок 7.3

Основні види технологічних операцій:

- проєктувальні технологічні операції
- оцінні технологічні операції

Технологічні інструкції:

- основна складова технології проєктування
- складаються з опису послідовності технологічних операцій проєктування, умов, залежно від яких виконується та або інша технологічна операція, та опису самих операцій

Предмет технології проєктування – відображення взаємозалежних процесів проєктування на всіх стадіях життєвого циклу ІС.

Основні вимоги до обраної технології проєктування:

- проєкт повинен відповідати вимогам замовника
- відбивати всі етапи циклу життя проєкту
- мінімальні трудові та вартісні витрати на проєктування й супровід проєкту
- основою зв'язку між проєктуванням і супроводом проєкту
- зростанню продуктивності праці проєктувальника
- надійність процесу проєктування й експлуатації проєкту
- простому веденню проєктної документації
- можливість виконання великих проєктів у вигляді підсистем
- можливість ведення робіт з проєктування окремих підсистем невеликими групами
- мінімальний час отримання працездатної ІС
- можливість управління конфігурацією проєкту, ведення версій проєкту і його складових, можливість автоматичного випуску проєктної документації і синхронізацію її версій з версіями проєкту
- незалежність виконуваних проєктних рішень від засобів реалізації ІС
- підтримана комплексом так званих CASE-засобів

Застосування технології проєктування неможливе без вироблення ряду стандартів, правил і угод, яких повинні дотримуватися усі учасники проєкту:

- ***стандартів проєктування***
  - набір необхідних діаграм (моделей)
  - правила фіксації проєктних рішень на моделях
  - вимоги до конфігурації робочих місць розробників
  - механізм забезпечення спільної роботи над проєктом
- ***стандартів оформлення проєктної документації***
  - комплектність, склад і структуру документації
  - вимоги до оформлення документації
  - правила підготовки, розгляду, узгодження і затвердження документації
  - вимоги до настройки видавничої системи
  - вимоги до настройки CASE-засобів
- ***стандартів призначеного для користувача інтерфейсу***
  - правила оформлення екранів, склад і розташування вікон та елементів управління
  - правила використання клавіатури й мишки
  - правила оформлення текстів допомоги
  - перелік стандартних повідомлень
  - правила обробки реакції користувача

Основа технології створення ІС – це технологічний процес:

- діяльність колективу спеціалістів, спрямовану на розробку проєкту ІС,
- дії, їх послідовність, виконавців, засоби та ресурси

Технологічний процес поділяється на окремі ***стадії, етапи чи складові частини***.

Технологічні операції (модулі) – самостійні частини технологічного процесу, в кожній з яких визначено вхід, вихід, перетворювач, ресурси та засоби.

Входом і виходом можуть бути документи, параметри, дані чи знання про технологічну операцію чи об’єкт.

Перетворювач – це методика, формалізований алгоритм чи машинний алгоритм перетворення входу технологічної операції в її вихід (ручні, людино-машинні та автоматичні).

Ресурси – це нормоване значення трудових, матеріальних, фінансових і технічних ресурсів.

7.3 ЗАСТОСУВАННЯ СТРУКТУРНОГО ПІДХОДУ ДО ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Структурний аналіз:

- виявлення структури як відносно стійкої сукупності відносин
- визнання методологічного переважного значення відносин над елементами в системі
- часткове відвернення від розвитку об'єктів.

Мета структурного аналізу – перетворення загальних, розпливчатих знань про предметну область в точні моделі, що описують різні підсистеми модельованої організації.

Сутність структурного підходу до розробки ІС полягає в її декомпозиції на функції, що автоматизуються.

Структурний аналіз передбачає дослідження системи за допомогою її графічного модельного подання, яке починається із загального огляду і подальшої деталізації.

В методології структурного аналізу традиційно використовують моделі, що показують:

- Функції
- Дані
- Організаційні структури
- Матеріальні та інформаційні потоки

Для побудови моделей у методологіях структурного аналізу застосовуються ***діаграми***:

- діаграми функціональних специфікацій, діаграми потоків даних, діаграми переходів станів у нотаціях (SADT (Structured Analysis and Design Technique); Йордана (Yourdon); Гейна-Сарсона (Gane-Sarson); SAG (Software AG));

- діаграми моделей даних «сутність-зв’язок» у нотаціях (Чена (Chen); Беркера; SADT; SAG);

- діаграми структури програмного додатку в нотаціях (Константайна (Constantine); Джексона (Jackson); Варньє-Орра (Warnier SAG).

Інструментальні засоби структурного аналізу і проєктування:

- BFD (Business Function Diagrams) – діаграми бізнес-функцій
- DFD (Data Flow Diagrams) – діаграми потоків даних
- ERD (Entity-Relationship Diagrams) – діаграми «сутність-зв’язок»
- STD (State Transition Diagrams) – діаграми переходів станів
- SSD (System Structure Diagrams) – діаграми структури ПД
- FDD (Functional Decomposition Diagrams) – діаграми функціональної декомпозиції
- IDEF0 – методологія функціонального моделювання
- IDEF1 – методологія аналізу і вивчення взаємозв’язків між інформаційними потоками
- IDEF1X – методологія інформаційного моделювання
- IDEF4 – методологія об’єктно-орієнтованого проєктування для підтримки проєктів
- IDEF5 – методологія, що забезпечує наочне подання даних

У структурному підході з метою проєктування модулів використовується ***техніка структурних карт***:

- структурні карти Константайна, призначені для опису відношень між модулями
- структурні карти Джексона – для опису внутрішньої структури модулів.

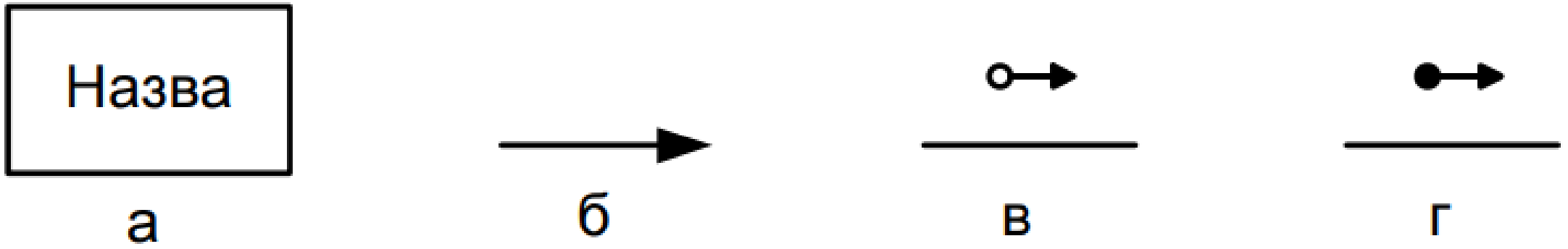


Рисунок 7.4 Основні компоненти структурних карт Константайна

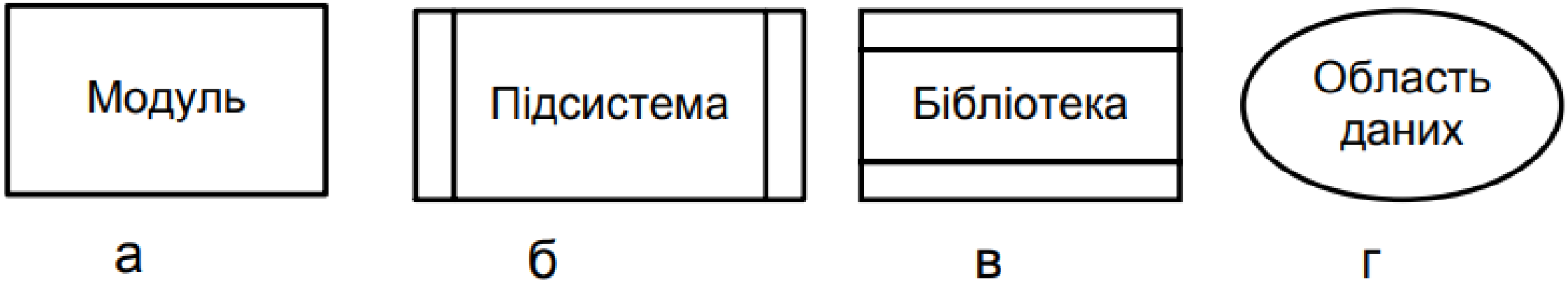


Рисунок 7.5 Типи модулів

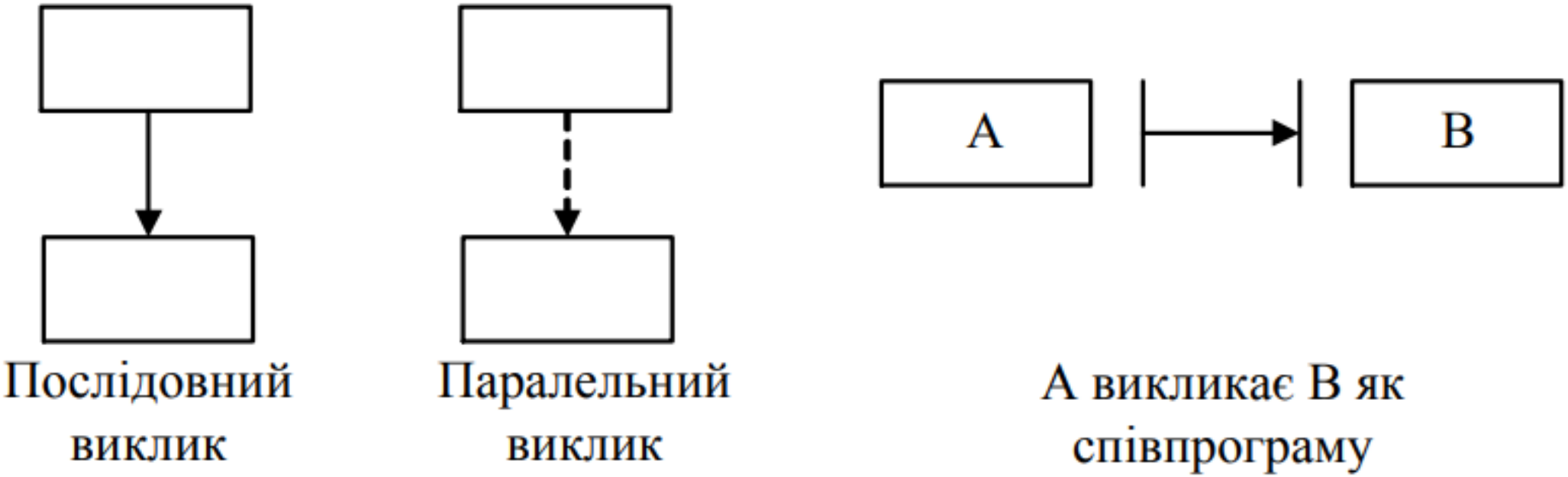
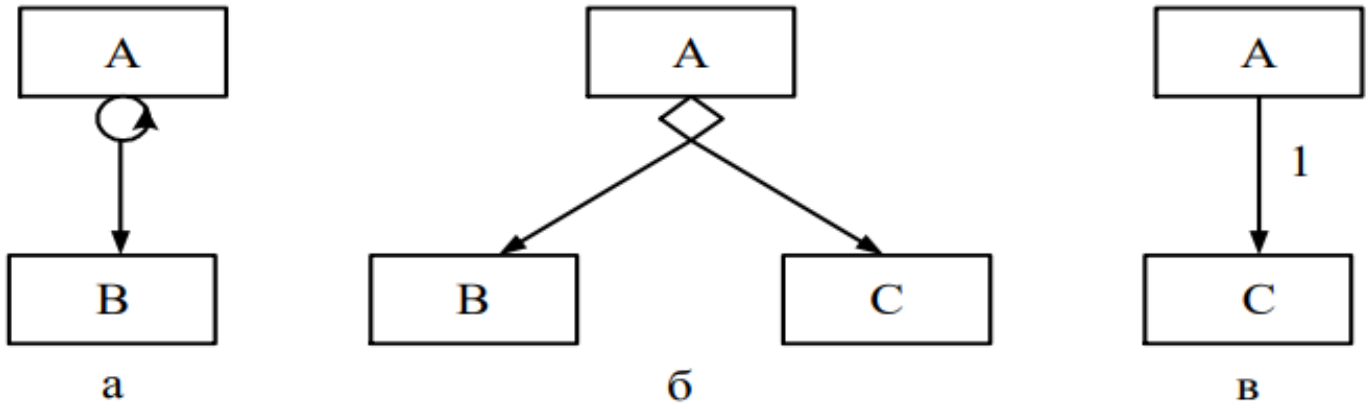


Рисунок 7.6 Типи викликів модулів



а – циклічний; б – умовний; в – одноразовий

Рисунок 7.7 Умовні і циклічні виклики модулів

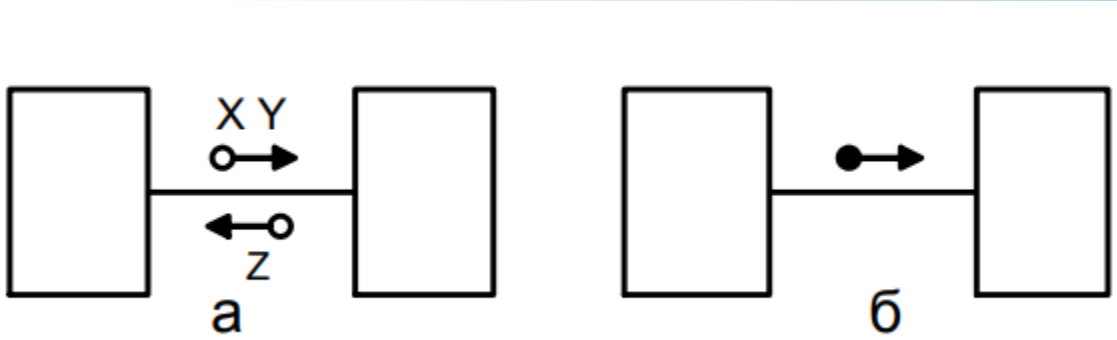


Рисунок 7.8 Зв'язки: а – за даними, б – за керуванням

Приклад. Розробити структурну карту Константайна для задачі сортування одновимірного масиву за допомогою алгоритмів бульбашки, прямого вибору і Шелла.

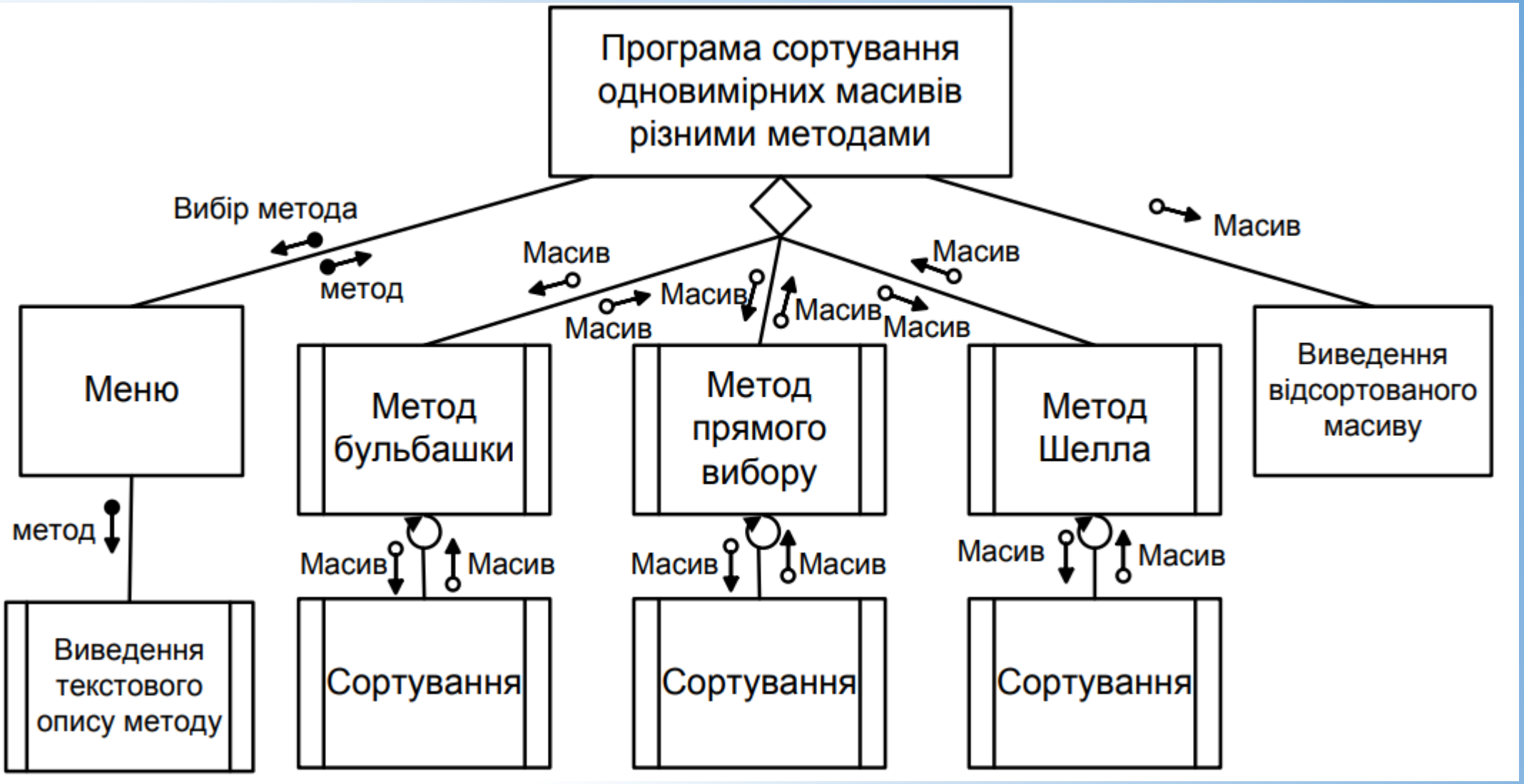
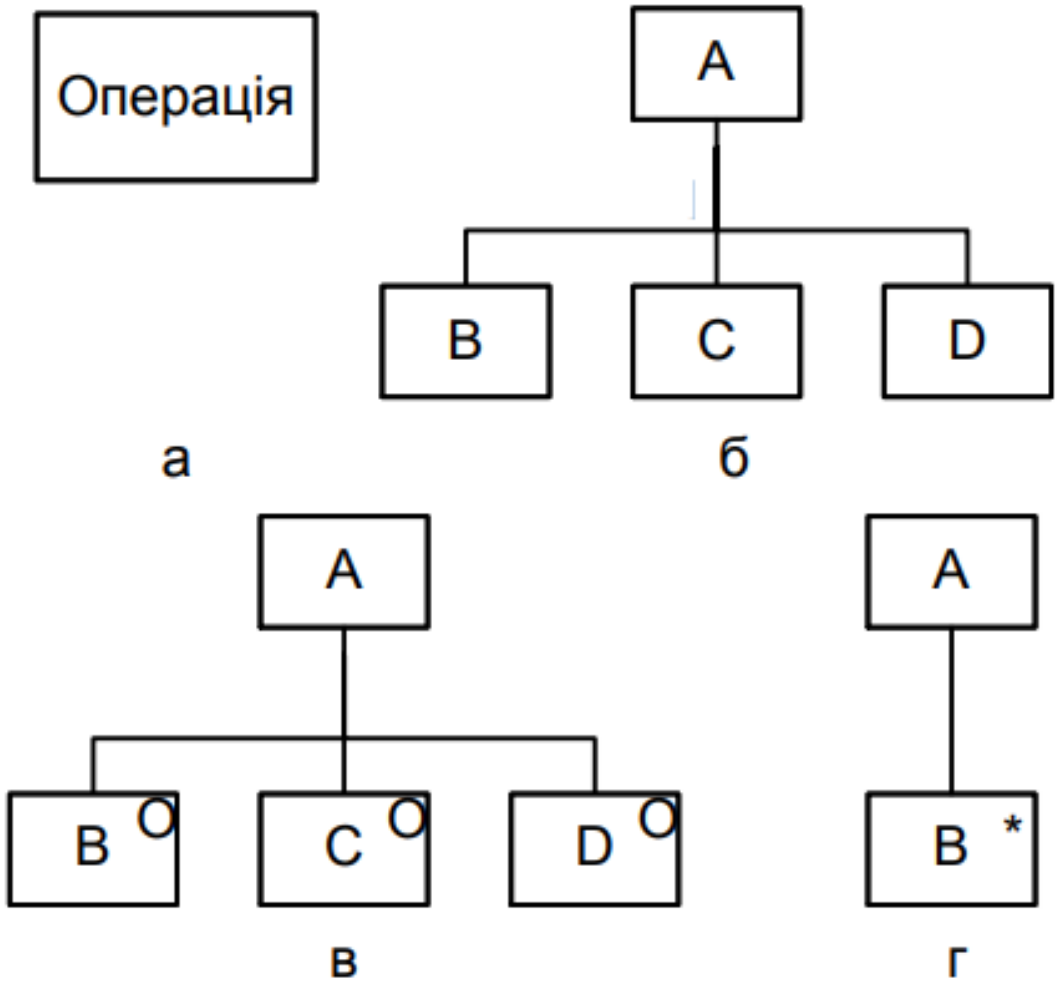


Рисунок 7.9 Приклад структурної карти Константайна



Приклад. У менеджера торгівельної фірми є файл, що містить записи про принтери з наступними полями: фірма-виробник, марка, швидкість друку, вартість, кількість одиниць на складі.

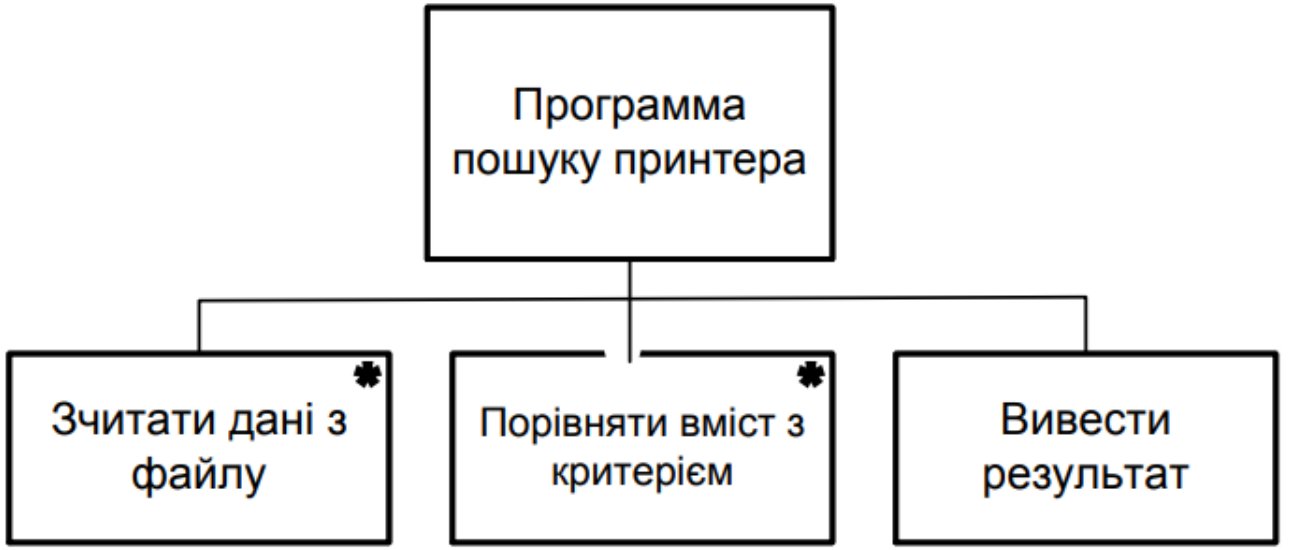


Рисунок 7.10 Елементи структурних діаграм Джексона

Рисунок 7.11 Структурна карта Джексона

Об'єктно-орієнтовані методології засновані на об'єктній декомпозиції предметної області.

Предметна область подається у вигляді сукупності об'єктів, що взаємодіють між собою за допомогою передачі повідомлень.

Об'єкт характеризується своїм станом та набором операцій для перевірки і зміни цього стану.
Об'єкт є представником певного класу однотипних об'єктів, що визначає його загальні властивості.

Об'єкти і класи організовуються з дотриманням ***принципів***:

- Принцип інкапсуляції
- Принцип спадкування
- Принцип поліморфізму

Об’єктно-орієнтований підхід ***полягає*** в поданні системи, що моделюється, у вигляді сукупності класів і об’єктів предметної сфери.

Кінцевим результатом процесу об’єктно-орієнтованого проєктування повинна стати множина класів об’єктів із приєднаними методами обробки атрибутів.

Система об’єктно-орієнтованих моделей послідовно розвертається в напрямку від статичної моделі загального подання функціонування ІС до моделі динамічної взаємодії об’єктів, на основі якої можуть бути згенеровані класи об’єктів у конкретному програмно-технічному середовищі.

Зміст операцій, що виконуються у випадку об'єктного підходу до проєктування

Рисунок 7.12

ТЕМА №8. АНАЛІЗ МЕТОДІВ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

8.1 Методи проєктування ІС і їх особливості

8.1 МЕТОДИ ПРОЄКТУВАННЯ ІС І ЇХ ОСОБЛИВОСТІ

Методи проєктування ІС – це різні способи створення ІС, що підтримуються відповідними засобами проєктування.

Ознаки класифікації методів проєктування ІС:

- за ступенем використання засобів автоматизації (ручний, комп'ютерний)
- за ступенем використання типових проєктних рішень (оригінальний, типовий)
- за ступенем використання адаптивності проєктних рішень до передбачуваних змін (реконструкція, параметризація, реструктуризація моделі).

Рисунок 8.1

Класи ТПР (класифікація ТПР заснована на рівні декомпозиції системи):

- елементні ТПР
- підсистемні ТПР
- об’єктні ТПР

Кожне ТПР передбачає наявність функціональних елементів, документації з детальним описом ТПР і процедур налаштування відповідно до вимог,

Підходи для реалізації типового проєктування:

- параметрично-орієнтований підхід до проєктування;
- модельно-орієнтований підхід до проєктування.

Параметрично-орієнтоване проєктування включає в себе етапи:

1. визначення критеріїв оцінки придатності пакетів прикладних програм (ППП) для вирішення поставлених завдань;
2. аналіз та оцінка доступних ППП по сформульованим критеріям;
3. вибір і закупівля найбільш підходящого пакета;
4. налаштування параметрів (дороблення) закупленого ППП.

Групи критеріїв оцінювання ППП: призначення і можливості пакета; відмінні ознаки і властивості пакета; вимоги до технічних і програмних засобів; документація пакета; фактори фінансового порядку; особливості установки пакета; особливості експлуатації пакета; допомога постачальника з впровадження і підтримки пакета; оцінка якості пакету і досвід його використання; перспективи розвитку пакета.

Модельно-орієнтоване проєктування:

- полягає в адаптації складу і характеристик типової ІС відповідно до моделі об’єкта автоматизації;
- технологія проєктування повинна забезпечувати єдині засоби для роботи як із моделлю типової ІС, так із моделлю конкретного підприємства.

Модельно-орієнтоване проєктування ІС передбачає, перш за все, побудову моделі об’єкта автоматизації з використанням спеціального програмного інструментарію.

Базова модель ІС в репозиторії містить опис бізнес-функцій, бізнеспроцесів, бізнес-об’єктів, бізнес-правил, організаційної структури, які підтримуються програмними модулями типової ІС.

Типові моделі описують конфігурації ІС для певних галузей або типів виробництва.

Модель конкретного підприємства будується або шляхом вибору фрагментів основної або типової моделі у відповідності зі специфічними особливостями підприємства, або шляхом автоматизованої адаптації цих моделей в результаті експертного опитування.

Бізнес-правила визначають умови коректності спільного застосування різних компонентів ІС і використовуються для підтримки цілісності створюваної системи.

Модель бізнес-функцій є ієрархічною декомпозицію функціональної діяльності підприємства.

Модель бізнес-процесів відображає виконання робіт для функцій самого нижнього рівня моделі бізнес-функцій.

Моделі бізнес-об'єктів використовують для інтеграції додатків, що підтримують виконання різних бізнес-процесів.

Модель організаційної структури підприємства є традиційною ієрархічною структурою підпорядкування підрозділів і персоналу.

Операції реалізації типового проєкту:

- встановлення глобальних параметрів системи;
- завдання структури об’єкта автоматизації;
- визначення структури основних даних;
- задання переліку реалізованих функцій і процесів;
- опис інтерфейсів;
- опис звітів;
- налаштування авторизації доступу;
- налаштування системи архівування.

Поєднання різних ознак класифікації методів проєктування обумовлює характер використовуваної технології проєктування (ТП) ІС, серед яких виділяються ***два основні класи:***

- канонічна технологія;
- індустріальна технологія.

У процесі проєктування ІС на всіх стадіях та етапах застосовується метод декомпозиції ***за двома напрямами***:

- декомпозиція даних
- декомпозиція процесів

Оригінальне (індивідуальне) проєктування – передбачає, що всі види проєктних робіт орієнтовано на створення індивідуальних проєктів для конкретних підприємств з урахуванням їхніх специфічних особливостей.

Типове проєктування передбачає застосування елементного, підсистемного, об'єктного методів проєктування.

Елементний метод проєктування передбачає декомпозицію, що здійснюється на рівні задач і окремих проєктних рішень на основі інформаційного, програмного, математичного і технічного забезпечення.

Підсистемний метод проєктування передбачає декомпозицію, що виконується на рівні підсистем, які виступають типовими елементами.

Об'єктне проєктування передбачає створення типового проєкту ІС для узагальненого об'єкта, виділеного з групи об'єктів як еталон, при цьому група однотипних об'єктів може бути невеликою.

Автоматизоване проєктування – це створення проєктів ІС на основі САПР, що ґрунтуються на глобальній інформаційній моделі ОД.

Організаційні методи охоплюють питання, які стосуються послідовності створення проєкту, добору спеціалістів на кожному етапі, забезпечення якісного документування проєкту, контролю проєктування, організації колективів розробників ІС, інформування учасників проєктування про стан розроблення проєкту, забезпечення виконання програмних та інформаційних інтерфейсів.

Модульний метод проєктування:

- пов’язаний зі створенням програмного та інформаційного забезпечення з множини відносно незалежних модулів
- модулі мають інформаційні взаємозв’язки
- дає змогу звести проєктування до оптимізуючого синтезу функціонально незалежних окремих частин

Переваги оптимального модульного синтезу:

- спрощуються розроблення і налагодження ПЗ
- спрощується подальша модифікація системи
- поліпшуються керуючі програми
- забезпечується можливість застосування технічних засобів
- поліпшується використання можливостей програмістів

Структурний метод передбачає наявність програм, що динамічно налагоджуються на структури масивів інформаційного фонду системи, при цьому опис масивів слід формалізувати, а їх збереження і підтримка в адекватному стані мають бути організовані в системі інформаційного фонду.

Метод «на основі математичної моделі» передбачає для розв'язання задачі вибір та розроблення економіко-математичної моделі, що включає створення алгоритму розв'язання і складання прикладної програми.

Метод неперервності розвитку системи полягає в тому, що після створення ІС у процесі її функціонування з'являються нові, змінюються діючі задачі управління, виникає необхідність внесення змін у систему.

ТЕМА №9. ЗАСОБИ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

9.1 Сутність та основні властивості засобів проєктування ІС

9.2 Аналіз комплексу узгоджених засобів проєктування ІС

9.3 Огляд класифікації засобів проєктування інформаційних систем

9.1 СУТНІСТЬ ТА ОСНОВНІ ВЛАСТИВОСТІ ЗАСОБІВ ПРОЄКТУВАННЯ ІС

Вимоги до засобів проєктування:

- повинні бути інваріантними до об’єкта проєктування;
- повинні охоплювати в сукупності всі етапи життєвого циклу ІС;
- повинні бути технічно, програмно й інформаційно сумісними;
- повинні бути простими в освоєнні та застосуванні;
- повинні бути економічно доцільними.

Засоби проєктування інформаційних систем – це комплекс інструментальних засобів, що забезпечують в рамках обраної методології проєктування підтримку повного життєвого циклу ІС.


Стратегічне планування
Аналіз
Проєктування
Реалізація
Впровадження
Експлуатація
Засоби проєктування ІС

Рисунок 9.1

Фактори, від яких залежить стратегія вибору ЗП ІС:

- характеристики моделюваної предметної області;
- цілі, потреби та обмеження майбутнього проєкту ІС;
- використовувана методологія проєктування.

Особливості сучасних складних ІС і проєктів:

- складність предметної області;
- наявність сукупності тісно взаємодіючих компонентів;
- ієрархічна структура взаємозв'язків компонентів;
- ієрархічна сукупність критеріїв якості функціонування компонентів і ІС;
- відсутність прямих аналогів;
- необхідність досить тривалого співіснування старих додатків і розроблених нових БД і додатків;

- наявність потреби як в традиційних додатках, так і в додатках аналітичної обробки;
- підтримка одночасної роботи локальних мереж і територіально віддалених користувачів;
- функціонування в неоднорідному операційному середовищі на декількох обчислювальних платформах;
- роз'єднаність і різнорідність окремих мікроколективів розробників;
- істотна тимчасова протяжність проєкту;
- досить часто застосовуються універсальні проєктні рішення;
- спроби використовувати готові рішення і ПЗ, що працюють в інших умовах;
- використання відкритих стандартів, принципів модульності.

9.2 АНАЛІЗ КОМПЛЕКСУ УЗГОДЖЕНИХ ЗАСОБІВ ПРОЄКТУВАННЯ ІС

Узгодженість засобів забезпечується ***наявністю інтерфейсів для прямої взаємодії і підтримкою загальноприйнятих стандартів відкритих систем.***

Комплекс узгоджених ЗП ІС ***дозволяє***:

- будувати моделі;
- формувати вимоги до ІС;
- швидко переходити від моделей вимог до ІС до проєкту додатків і БД.

Комплекс узгоджених ЗП ІС ***забезпечує***:

- підтримку швидкої ітеративної розробки додатків;
- тестування додатків;
- інтеграцію в систему.

Методологія та набір інструментальних засобів включають:

- підтримку колективної розробки з можливістю паралельного та розподіленого виконання різних робіт;
- можливість переходу до наступного етапу, не чекаючи повного завершення попереднього;
- застосування методів контролю якості та постійний контроль отриманих результатів;
- підтримку ітеративного характеру розробки;
- можливість швидкого внесення змін у вимоги в процесі розробки;
- управління конфігурацією.

9.3 ОГЛЯД КЛАСИФІКАЦІЇ ЗАСОБІВ ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

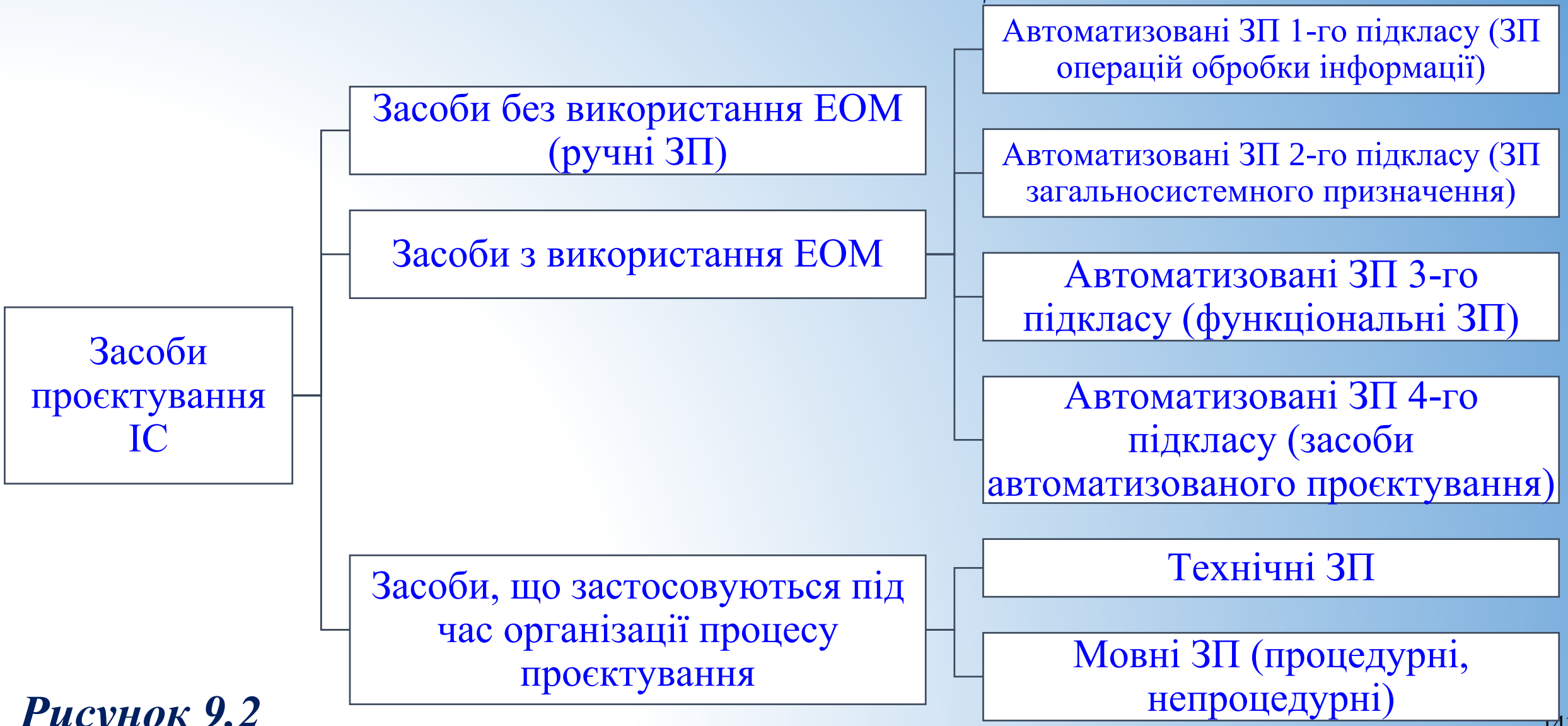


Рисунок 9.2

Мовні засоби високого рівня непроцедурного типу застосовують як формальні засоби для забезпечення однозначності й можливості аналізу ІС.

Апарат теорії фреймів – апарат, що є найефективнішим для відображення семантики первинних інформаційних сукупностей показників.

Фрейм – це структура даних для подання знань у конкретній предметній сфері.

При розробці ПЗ велике значення має вибір мови, оскільки від неї значною мірою залежить багато характеристик створюваної ІС:

- успішність і швидкість впровадження;
- простота експлуатації та проєктування програми;
- ефективність функціонування складного програмного комплексу.

Різновиди програмних засобів:

- локальні;
- комплексні.

Локальні програмні засоби – застосовуються для автоматизації окремих проєктних робіт і можуть використовуватися під час проєктування незалежно один від одного.

Локальні програмні засоби: генератори програм; автономні ППП; системи програмування (транслятори, інтерпретатори, генератори ППП, макрогенератори); СУБД; системи телеобробки; інструментальні засоби проєктування; окремі елементи комплексних засобів (система програмування МОД); засоби ОС обчислювального комплексу (системне ПЗ).

ТЕМА №10. ЗАСТОСУВАННЯ UML ПІД ЧАС ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

10.1 Сутність мови UML в процесі ООП ІС. Особливості та переваги застосування

10.2 Історія розвитку мови UML

10.3 Особливості та переваги застосування UML

10.4 Об'єктно-орієнтоване проєктування з використанням UML. Основи UML та елементи нотації

10.5 UML-модель і її елементи

10.6 Огляд діаграм та їх застосування

10.1 СУТНІСТЬ МОВИ UML В ПРОЦЕСІ ООП ІС

Сутність об'єктно-орієнтованого підходу до аналізу та проєктування систем полягає в декомпозиції системи на класи, які відповідають однотипним об'єктам предметної області, і побудові з них ієрархії у вигляді орієнтованого графа з використанням відношень композиції і успадкування.

Базові складові ООП:

- уніфікований процес;
- уніфікована мова моделювання;
- шаблони проєктування.

Уніфікований процес – це процес розробки програмного забезпечення (ПЗ), який забезпечує упорядкований підхід до розподілу завдань і обов'язків в організації-розробника.

UML – мова (система позначень) для визначення, візуалізації і конструювання моделей/проєктів системи у вигляді діаграм і документів на основі об'єктно-орієнтованого підходу.

Уніфікована мова моделювання – це графічна мова для візуалізації, специфікації, конструювання та документування систем, у яких велика роль належить програмному забезпеченню.

Шаблон – це іменована пара «проблема/рішення», що містить готове узагальнене рішення типової проблеми.

UML (англ. Unified Modeling Language):

- уніфікована мова моделювання, що використовується у парадигмі об'єктно-орієнтованого проєктування та програмування систем;
- мова широкого профілю;
- відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи;
- створена для визначення, візуалізації, проєктування та документування в основному програмних систем;
- не є мовою програмування.

Призначення UML: дозволяє розробникам ПЗ досягти угоди у графічних позначеннях для представлення загальних понять та більше сконцентруватися на проєктуванні й архітектурі.

Як формальна штучна мова UML має:

- синтаксис;
- семантика;
- Прагматика.

UML мова ***графічна***, а не текстова; UML ***мова моделювання***, а не програмування.

Мова UML необхідна:

- керівникам проєктів, які керують розподілом завдань і контролем за проєктом;
- проєктувальникам ІС, які розробляють технічні завдання для програмістів;
- бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії;
- програмістам, які реалізують модулі ІС.

10.2 ІСТОРІЯ РОЗВИТКУ МОВИ UML

UML розвивається ***з другої половини 1990-х років*** і бере свій початок з об'єктно-орієнтованих методів програмування, розроблених наприкінці 1980-х і на початку 1990-х років.

Жовтень 1994 р.: Граді Буч і Джеймс Румбах з Rational Software Corporation почали роботу з уніфікації своїх методів об’єктно-орієнтованого аналізу програм. Мета – об’єднання переваг всіх відомих об’єктно-орієнтованих методів і повна уніфікація результатів своєї роботи.

Жовтень 1995 р.: опублікований проєкт т.зв. уніфікованого методу (Unified Method).

Версії мови UML:

- *1.0 – січень 1997 р.*
- *1.1 – листопад 1997 р.*
- *1.3 – березень 2000 р.*
- *1.4 – вересень 2001 р.*
- *1.4.2 – липень 2004 р.*
- *1.5 – березень 2003 р.*
- *2.0 липень – 2005 р.*
- *2.1 – формально не прийнята*
- *2.1.1 – серпень 2007 р.*
- *2.1.2 – листопад 2007 р.*
- *2.2 – лютий 2009 р.*
- *2.3 – травень 2010 р.*
- *2.4 бета – 2 березень 2011 р.*
- *2.5 – червень 2015 р.*
- *2.5.1 – грудень 2017 р.*

10.3 ОСОБЛИВОСТІ ТА ПЕРЕВАГИ ЗАСТОСУВАННЯ UML

UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему у такому вигляді, щоб її можна було легко перевести в програмний код.

Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування.

Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення.

Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи або точці зору на систему.

За допомогою мови ***UML*** можна ***розробити детальний проєкт створюваної системи***, що відображає:

- концептуальні елементи;
- конкретні особливості реалізації.

Система об’єктно-орієнтованих моделей відповідно до нотацій UML ***включає*** наступні ***канонічні діаграми*** для формування проєкту системи:

- діаграма варіантів (випадків) використання;
- діаграма класів;
- діаграма послідовності;
- діаграма взаємодії (співпраці);
- діаграма станів;
- діаграма діяльності;
- діаграма компонентів;
- діаграма впровадження;
- діаграма взаємозв’язку сутностей;
- діаграма пакетів;
- діаграма розміщення.

10.4 ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЄКТУВАННЯ З ВИКОРИСТАННЯМ UML. ОСНОВИ UML ТА ЕЛЕМЕНТИ НОТАЦІЇ

Мова UML ***призначена*** для опису, візуалізації та документування об'єктно-орієнтованих систем у процесі їх розробки.

Розробка моделі додатку за допомогою мови UML:

- ***побудова діаграм використання*** (Use Case Diagram);
- розробка ***діаграми взаємодії «користувач-система»***, при цьому виявляються необхідні об'єкти додатку, будуються діаграми класів, формується компонентна структура програмного забезпечення;

Діаграми взаємодії об'єктів (Interaction Diagrams) відносяться до діаграм процесів, що відображають поведінковий аспект моделювання.

У ***діаграмах послідовностей*** (Sequence Diagram), які також називаються діаграмами сценаріїв, відбивається послідовність подій, що полягають у діях одного об'єкта на деякий інший об'єкт.

У ***діаграмах кооперації*** (Collaboration Diagram) об'єкти, представлені прямокутниками, які зв'язані між собою лініями, що зображають повідомлення (потік управління).

За допомогою ***діаграми станів*** моделюється послідовність подій, що відбуваються в системі.

Діаграми діяльності (Activity Diagram) близькі за своєю семантикою до діаграм станів.

У UML використовуються також діаграми компонентів і розгортання, які застосовуються для моделювання фізичної організації інформаційної системи.
У діаграмах розгортання показують розподіл класів за апаратними засобами.

Нотація – правила запису (малювання) моделей.

Типи елементів нотації:

- фігура (shape);
- лінія (line);
- значок (icon);
- текст (text);
- рамка (frame).

Канонічна нотація – будь-яка модель може бути описана монохромними рисунками з текстовими поясненнями, при цьому рисунки повинні залишатися зрозумілими після друку на чорно-білому принтері.

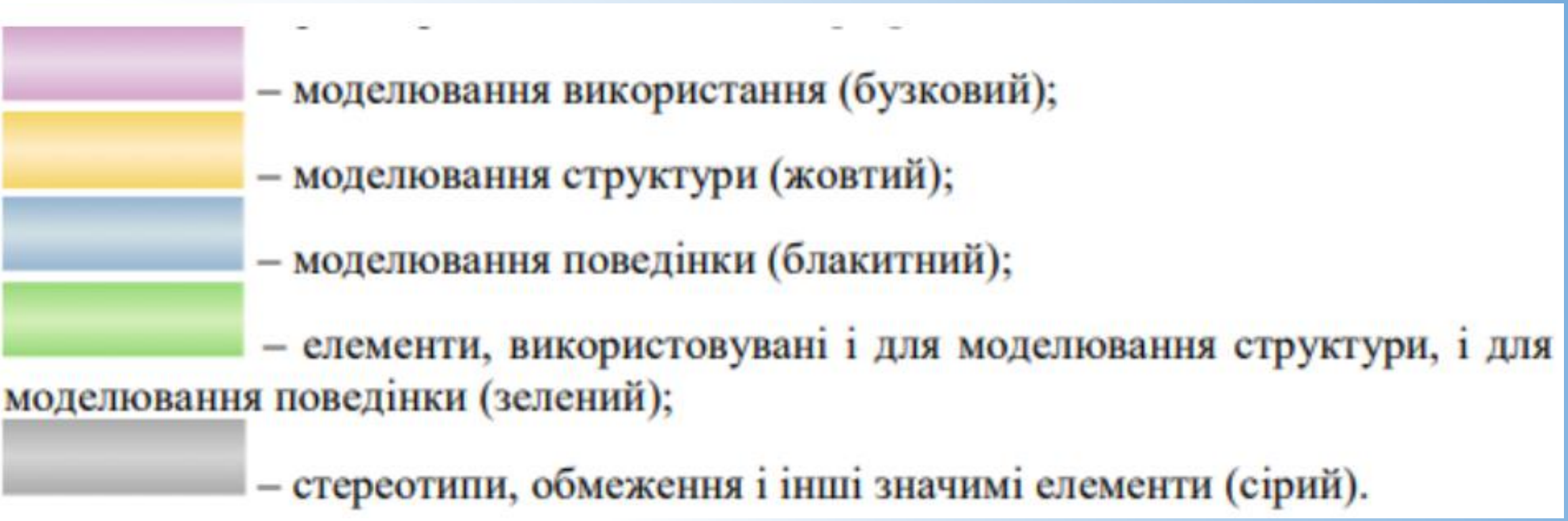


Рисунок 10.1 Палітра кольорів для представлення моделі

10.5 UML-МОДЕЛЬ І ЇЇ ЕЛЕМЕНТИ

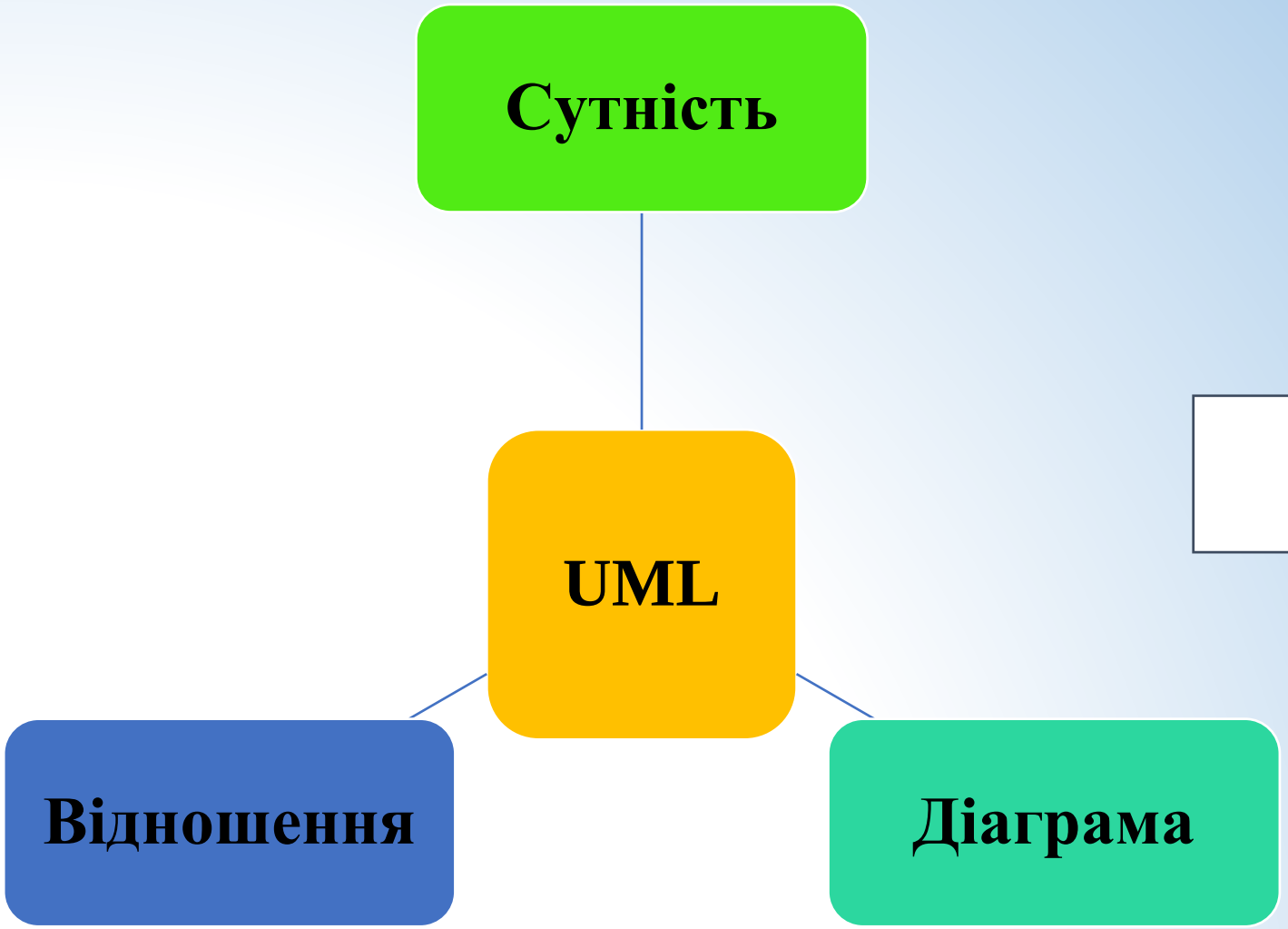


Рисунок 10.2

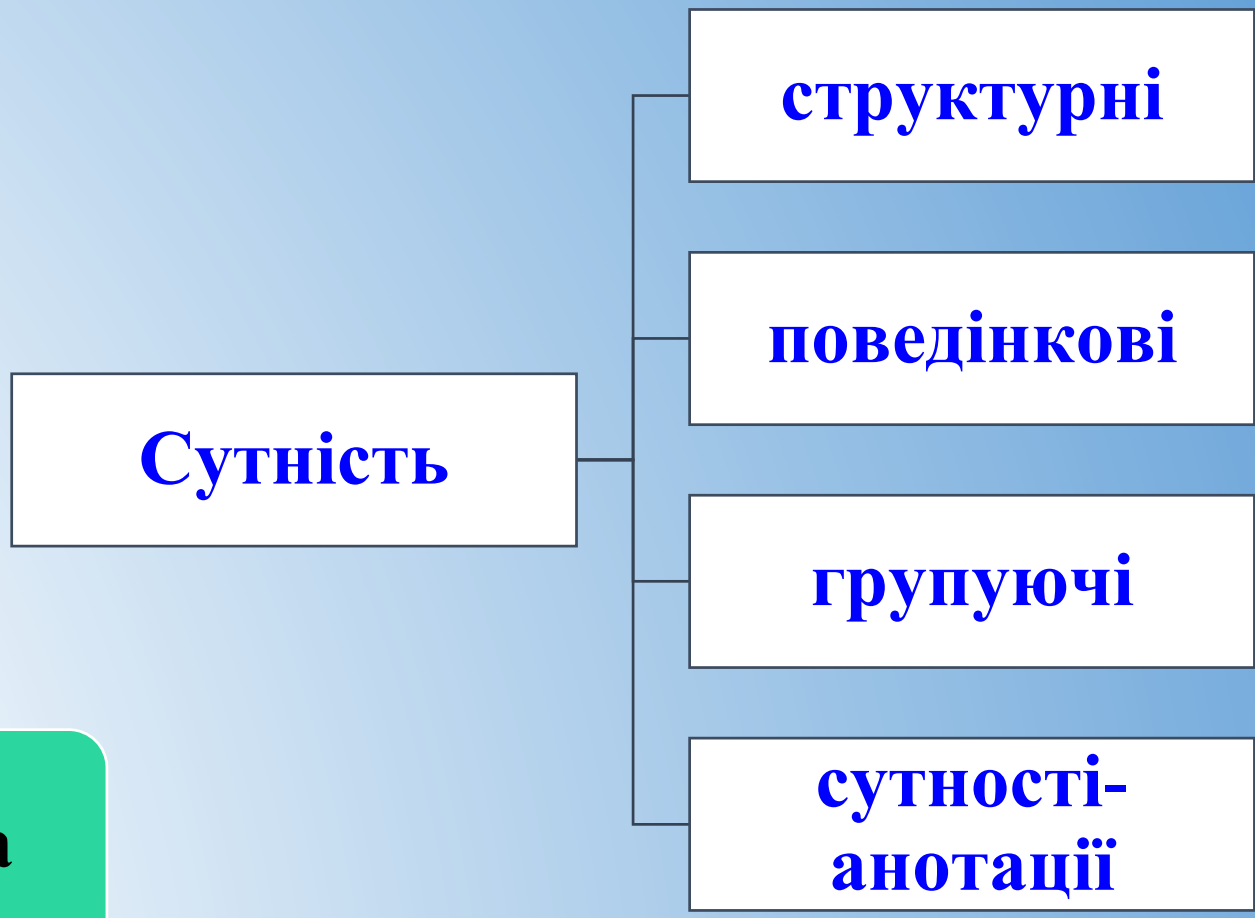
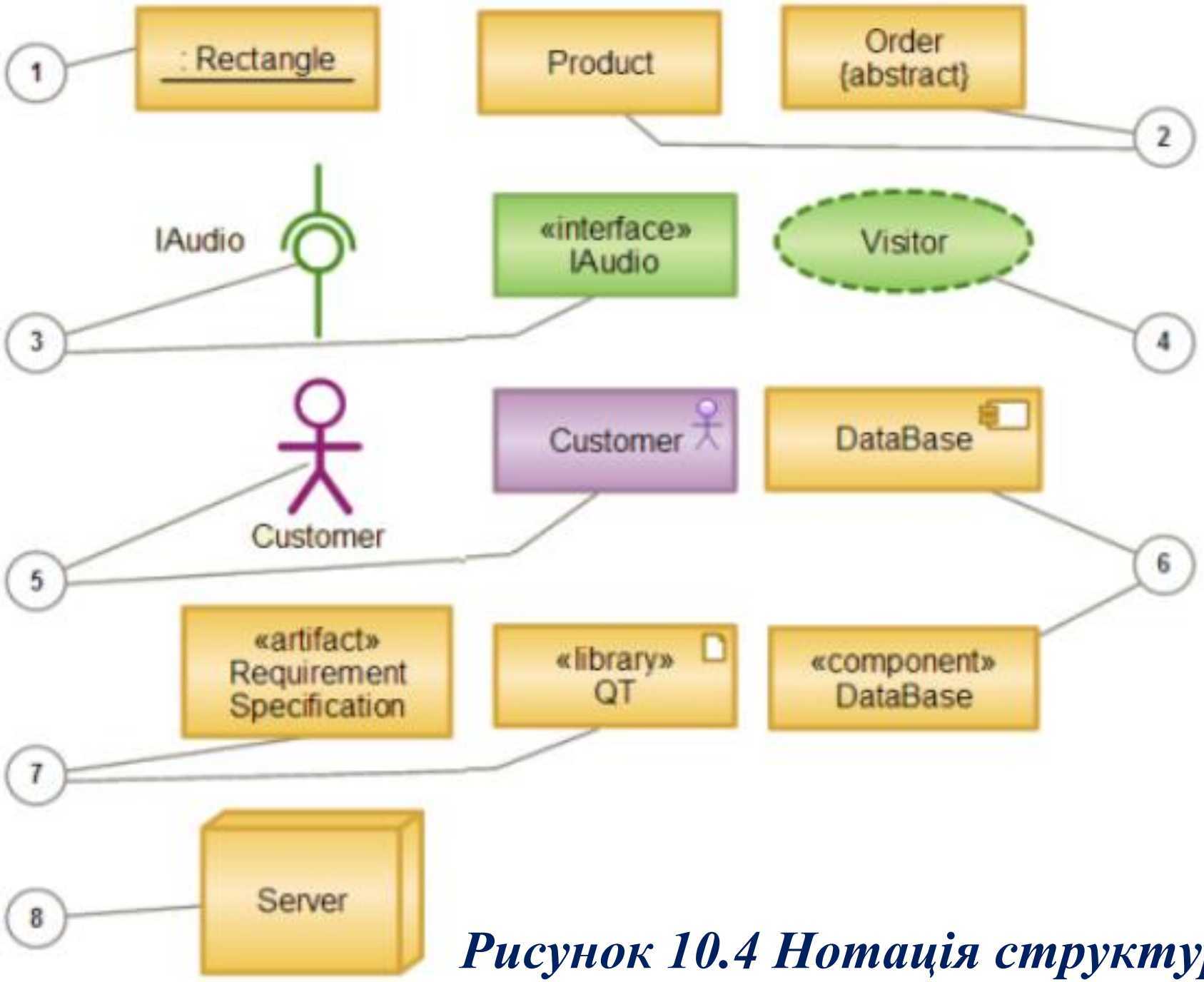
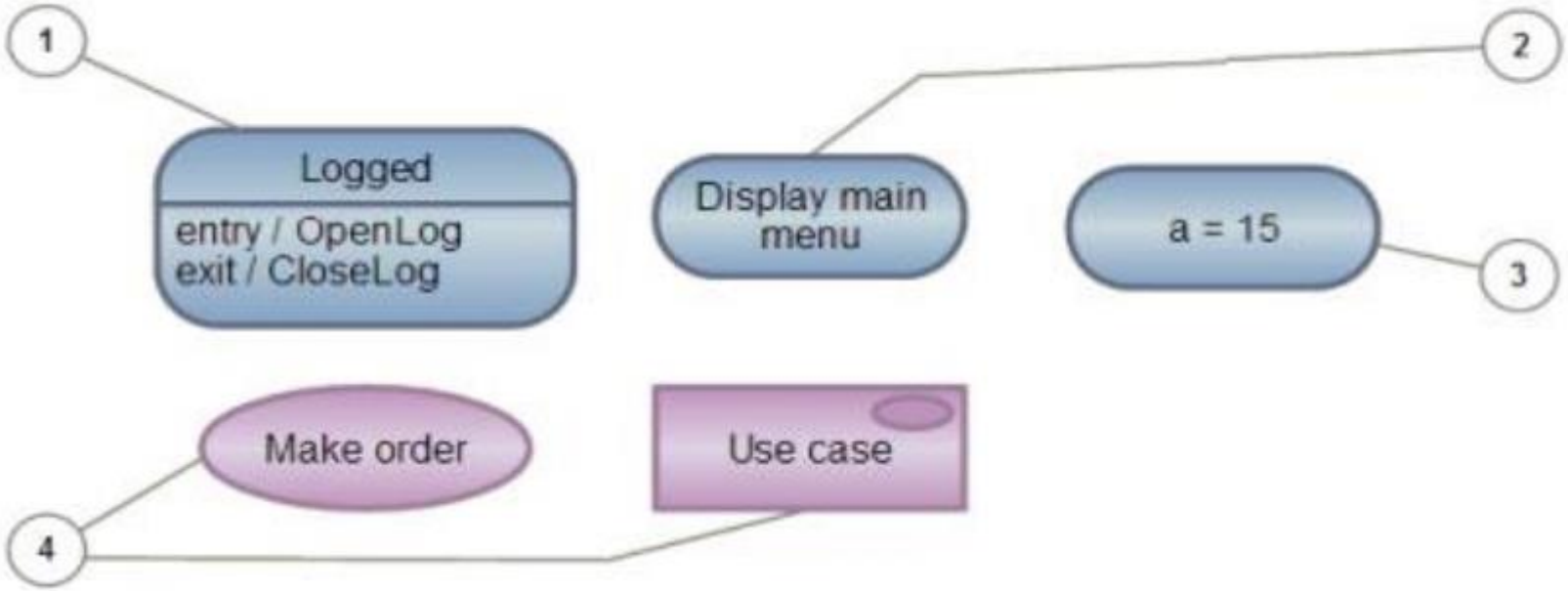


Рисунок 10.3



1 – Об'єкт (object)
2 – Клас (class)
3 – Інтерфейс (interface)
4 – Кооперація (collaboration)
5 – Дійова особа (actor)
6 – Компонент (component)
7 – Артефакт (artifact)
8 – Вузол (node)

Рисунок 10.4 Нотація структурних сутностей



1 – Стан (state)
2 – Діяльність (activity)
3 – Дія (action)
4 – Варіант використання (use case)

Рисунок 10.5 Нотація поведінкових сутностей

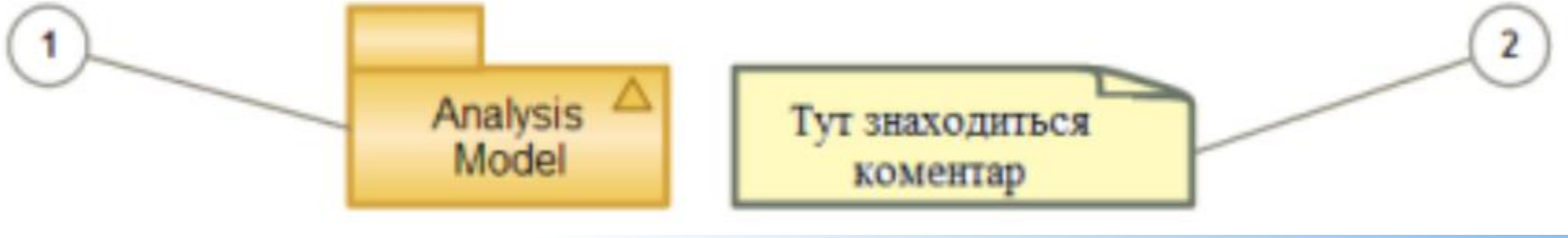


Рисунок 10.6 Групуюча нотація (1) та коментар (2)

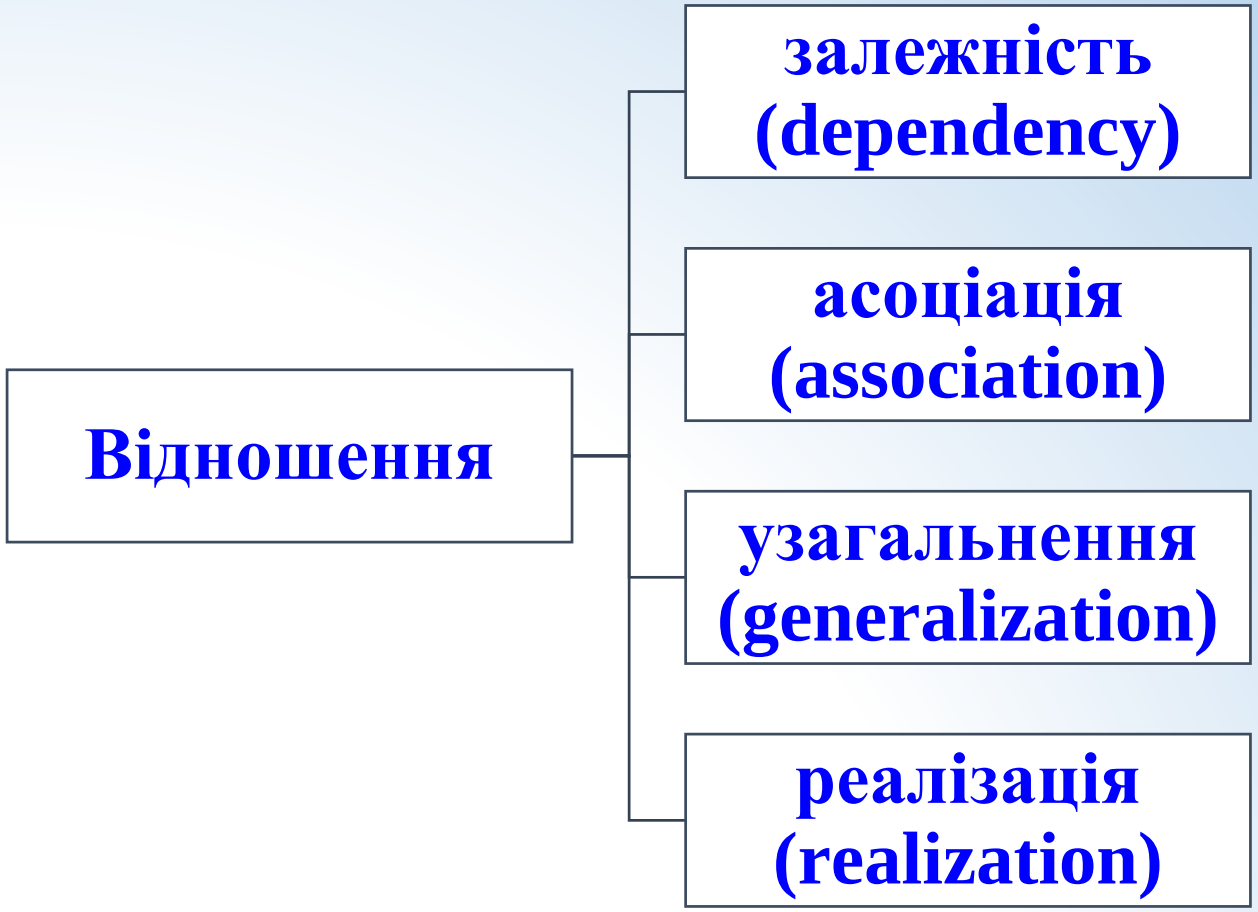


Рисунок 10.7

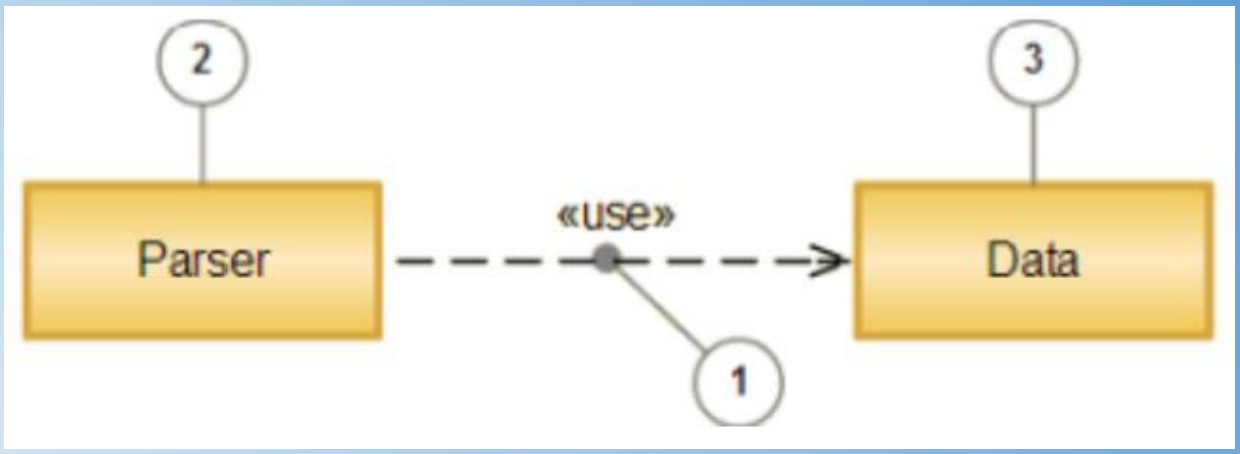


Рисунок 10.8 Відношення залежності

1 – Залежність
2 – Залежна сутність
3 – Незалежна сутність

1 - Асоціація

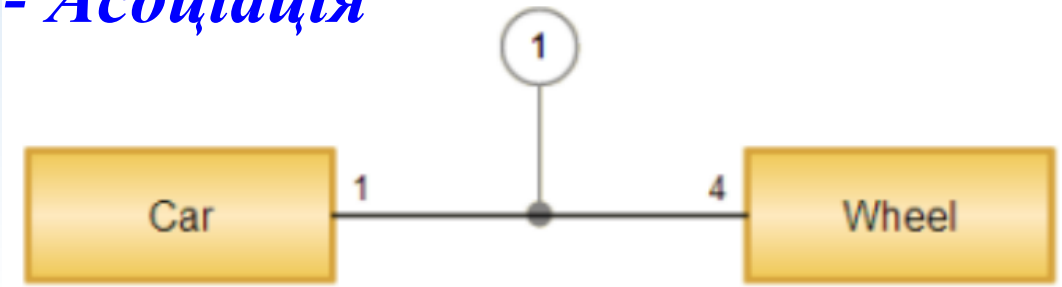


Рисунок 10.9 Відношення асоціації

1 – Узагальнення
2 – Підклас
3 – Суперклас

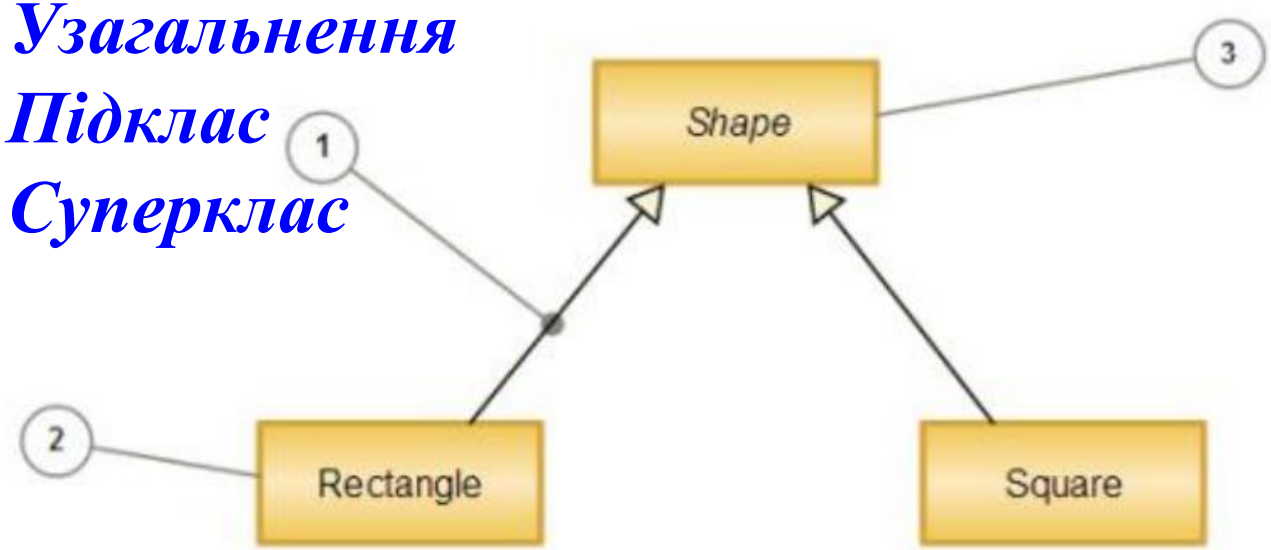


Рисунок 10.10 Відношення узагальнення

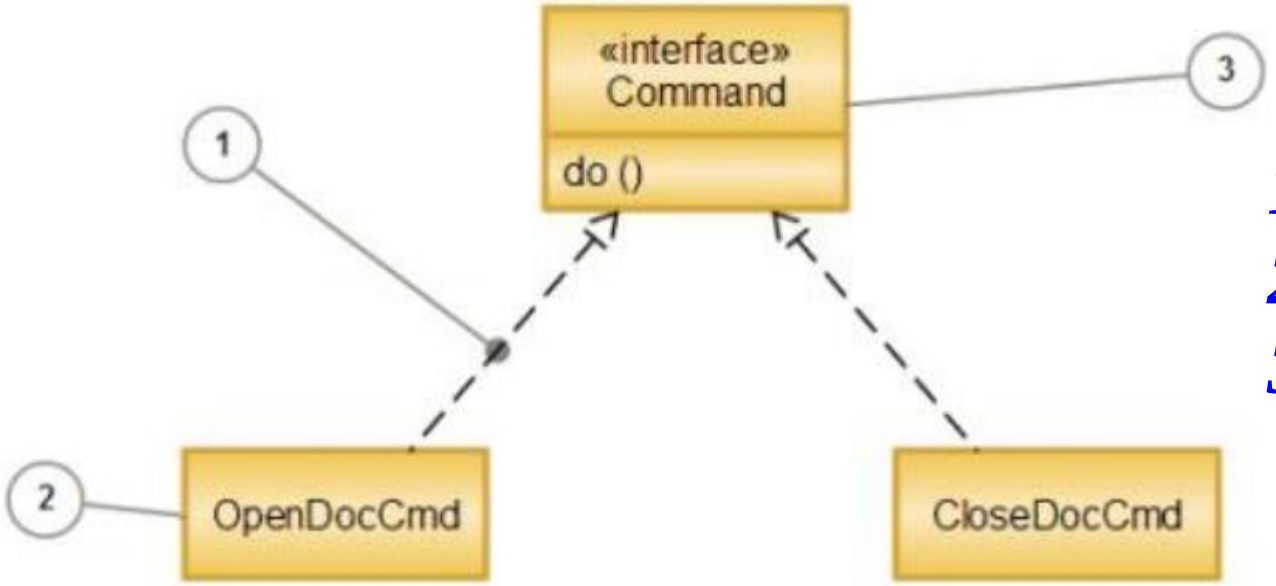
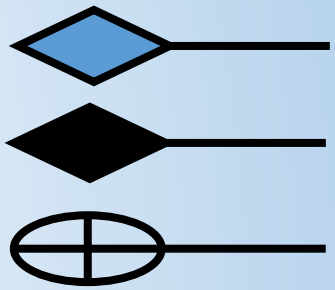


Рисунок 10.11 Відношення реалізації

1 – Реалізація
2 – Реалізовуюча сутність
3 – Сутність, що реалізується



Агрегація
Композиції
Включення

10.6 ОГЛЯД ДІАГРАМ ТА ЇХ ЗАСТОСУВАННЯ

Діаграми UML – основна структура, що накладається на модель, яка полегшує створення і використання моделі.

Конструкції мови – це тексти, які можуть бути написані усередині фігур сутностей або поряд з лініями відношень, рамки діаграм і їх фрагментів, значки, що приєднуються до ліній або поміщаються всередину фігур.

Канонічні типи діаграм (класична класифікація):

- Діаграма використання (Use Case Diagram).
- Діаграма класів (Class Diagram).
- Діаграма об’єктів (Object Diagram).
- Діаграма станів (State Chart Diagram).
- Діаграма діяльності (Activity Diagram).
- Діаграма послідовності (Sequence Diagram).
- Діаграма кооперації (Collaboration Diagram).
- Діаграма компонентів (Component Diagram).
- Діаграма розміщення (Deployment Diagram).

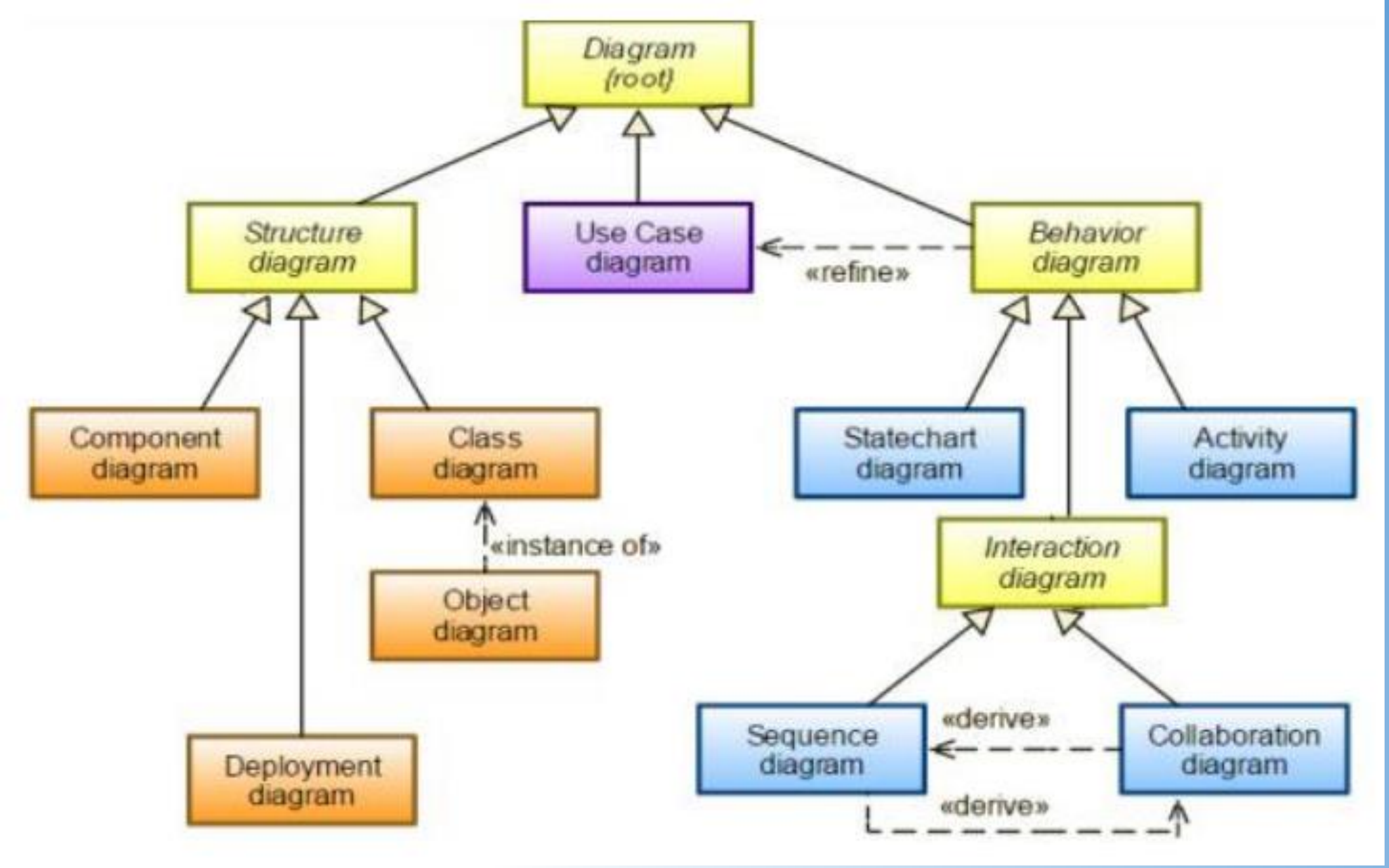


Рисунок 10.12 Ієрархія типів діаграм для UML 1

Список нових діаграм і їх назв:

- Діаграма внутрішньої структури (Composite Structure Diagram).
- Діаграма пакетів (Package Diagram).
- Діаграма автомата (State Machine Diagram).
- Діаграма комунікації (Communication Diagram).
- Оглядова діаграма взаємодії (Interaction Overview Diagram).
- Діаграма синхронізації (Timing Diagram).

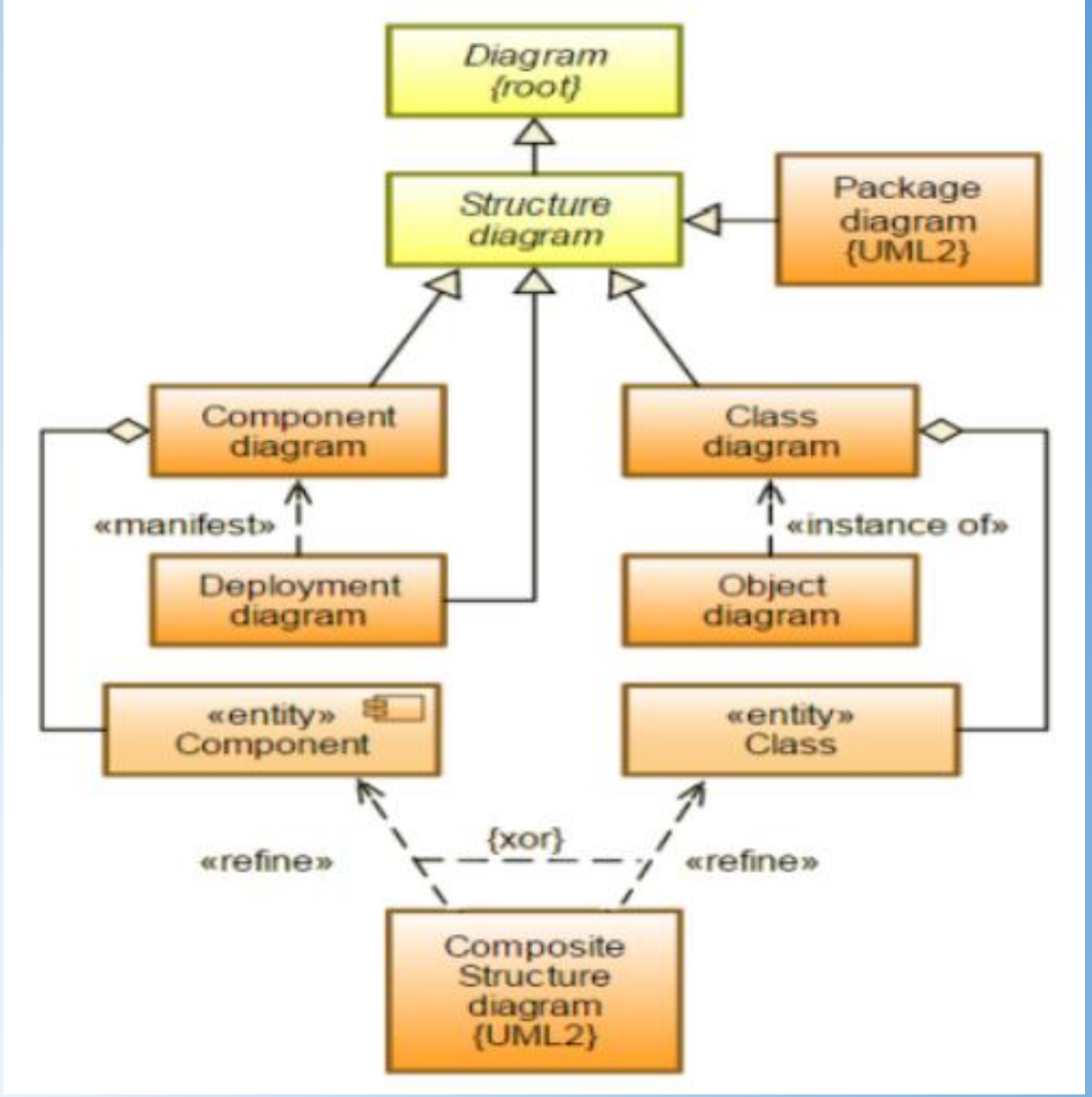
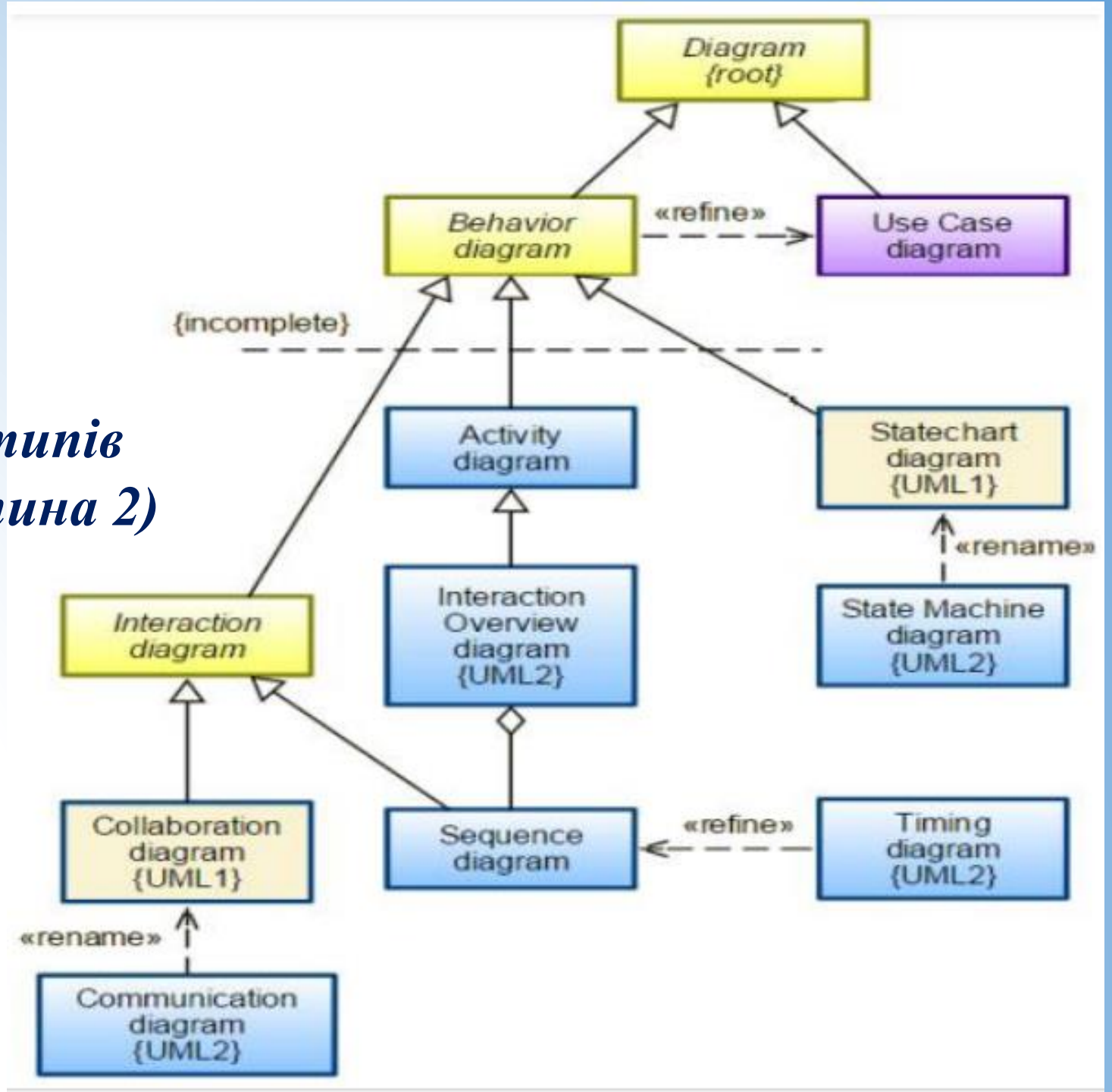


Рисунок 10.13 Ієрархія типів діаграм для UML 2 (Частина 1)

Рисунок 10.14 Ієрархія типів діаграм для UML 2 (Частина 2)



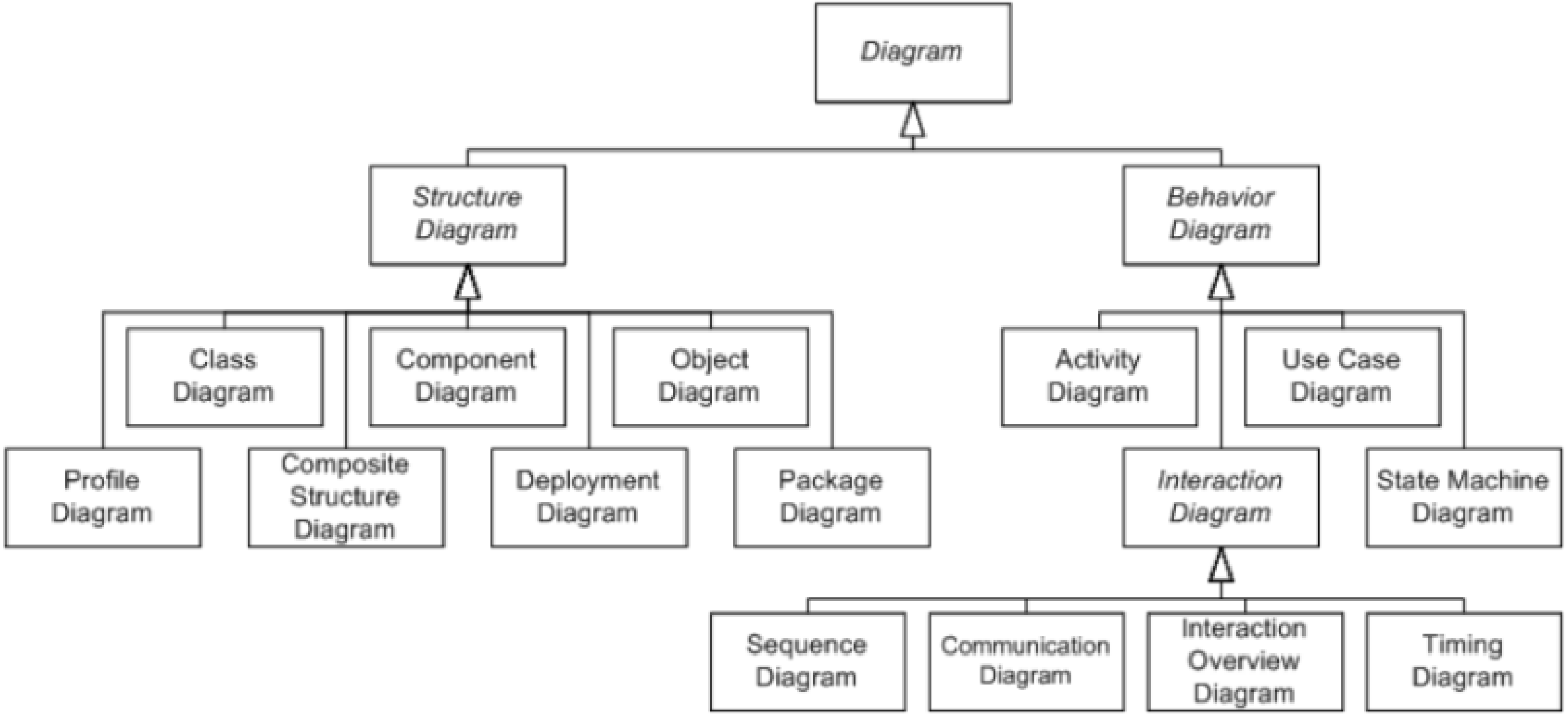


Рисунок 10.15 Ієрархія типів діаграм для UML 2.3

Рисунок 10.16 Нотація для діаграм (зовнішня рамка, ярличок з назвою діаграми)

Рисунок 10.17 Типи і теги діаграм

Назва діаграми	Тег (стандартний)	Тег (пропонований)
Діаграма використання	**use case** або **uc**	use case
Діаграма класів	class	class
Діаграма автомата	**state machine** або **stm**	state machine
Діаграма діяльності	**activity** або **act**	activity
Діаграма послідовності	**interaction** або **sd**	sd
Діаграма комунікації	**interaction** або **sd**	comm
Діаграма компонентів	**component** або **cmp**	component
Діаграма розміщення	не визначений	deployment
Діаграма об'єктів	не визначений	object
Діаграма внутрішньої структури	class	**class** або **component**
Оглядова діаграма взаємодії	**interaction** або **sd**	interaction
Діаграма синхронізації	**interaction** або **sd**	timing
Діаграма пакетів	**package** або **pkg**	package

Діаграми UML умовно розбиті на дві групи: загальні і спеціальні діаграми.

Рисунок 10.18

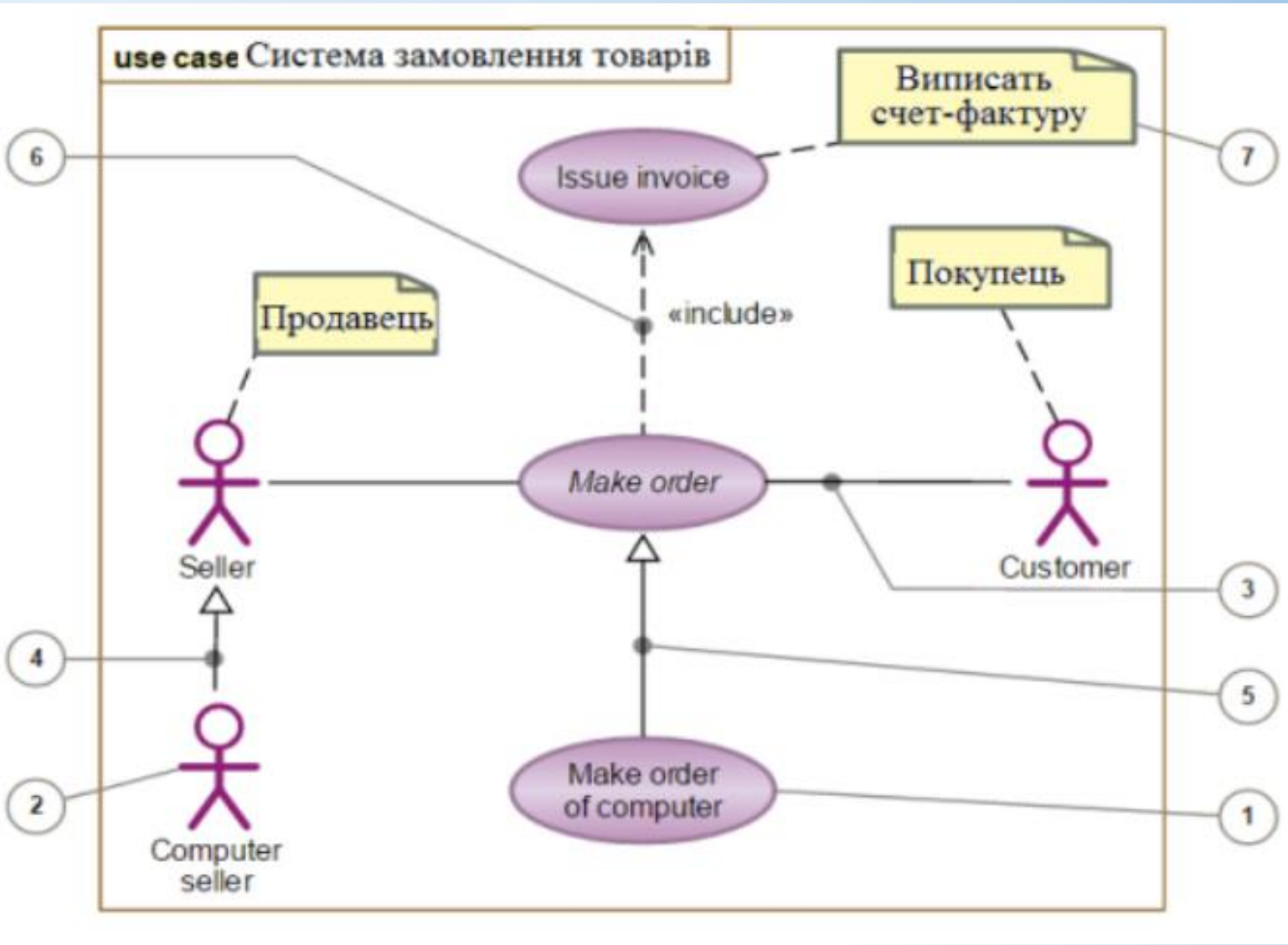
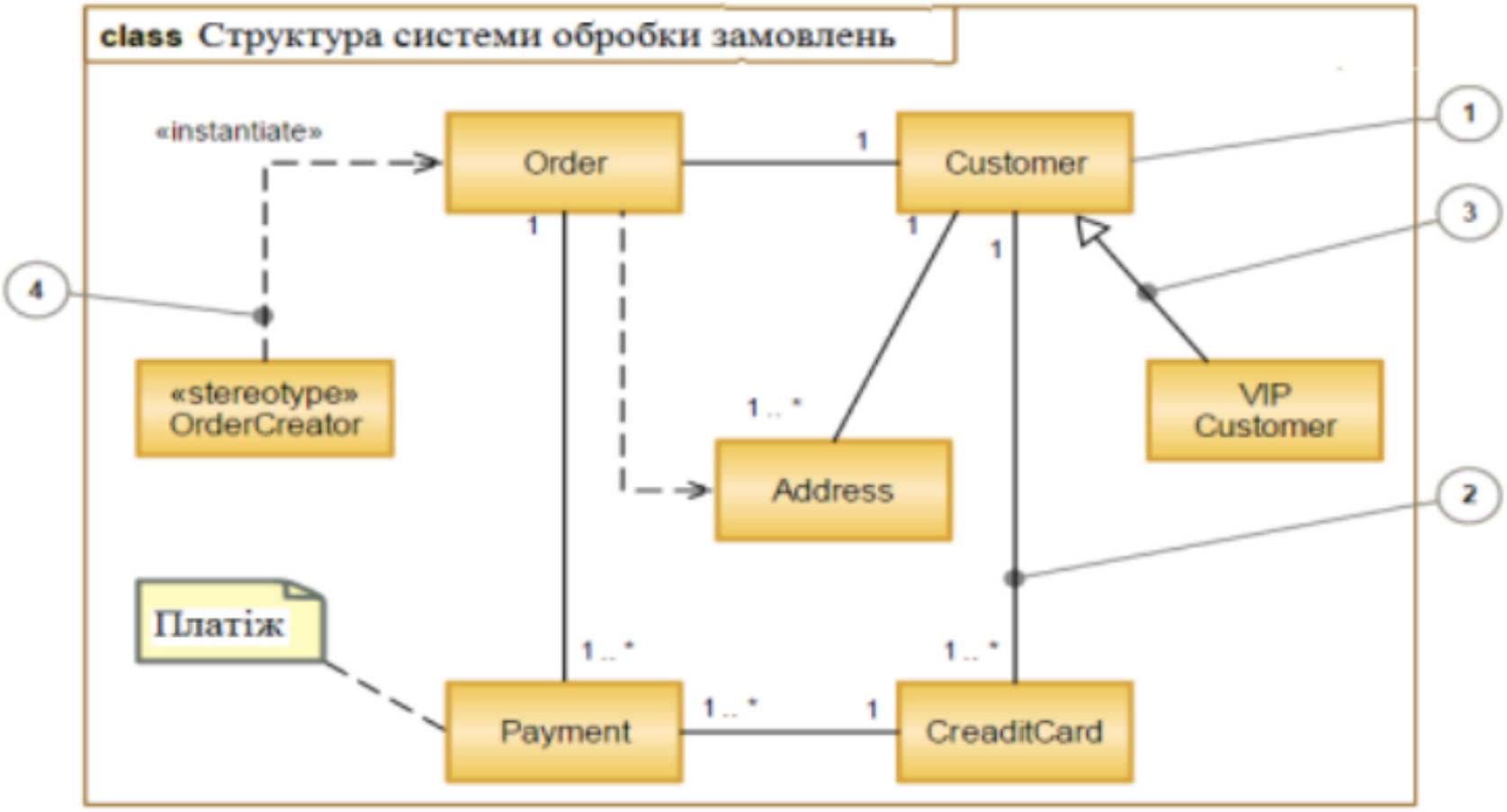


Рисунок 10.19 Нотація діаграми прецедентів

1 – Варіанти використання
2 – Діючі особи
3 – Асоціація між дійовою особою та варіантом використання
4 – Узагальнення між дійовими особами
5 – Узагальнення між варіантами використання
6 – Залежності (різних типів) між варіантами використання
7 – Коментарі



1 – Класи
2 – Асоціація між класами (з безліччю додаткових подробиць)
3 – Асоціація між класами
4 – Залежності (різних типів) між класами та між класами й інтерфейсами.

Рисунок 10.20 Нотація діаграми класів

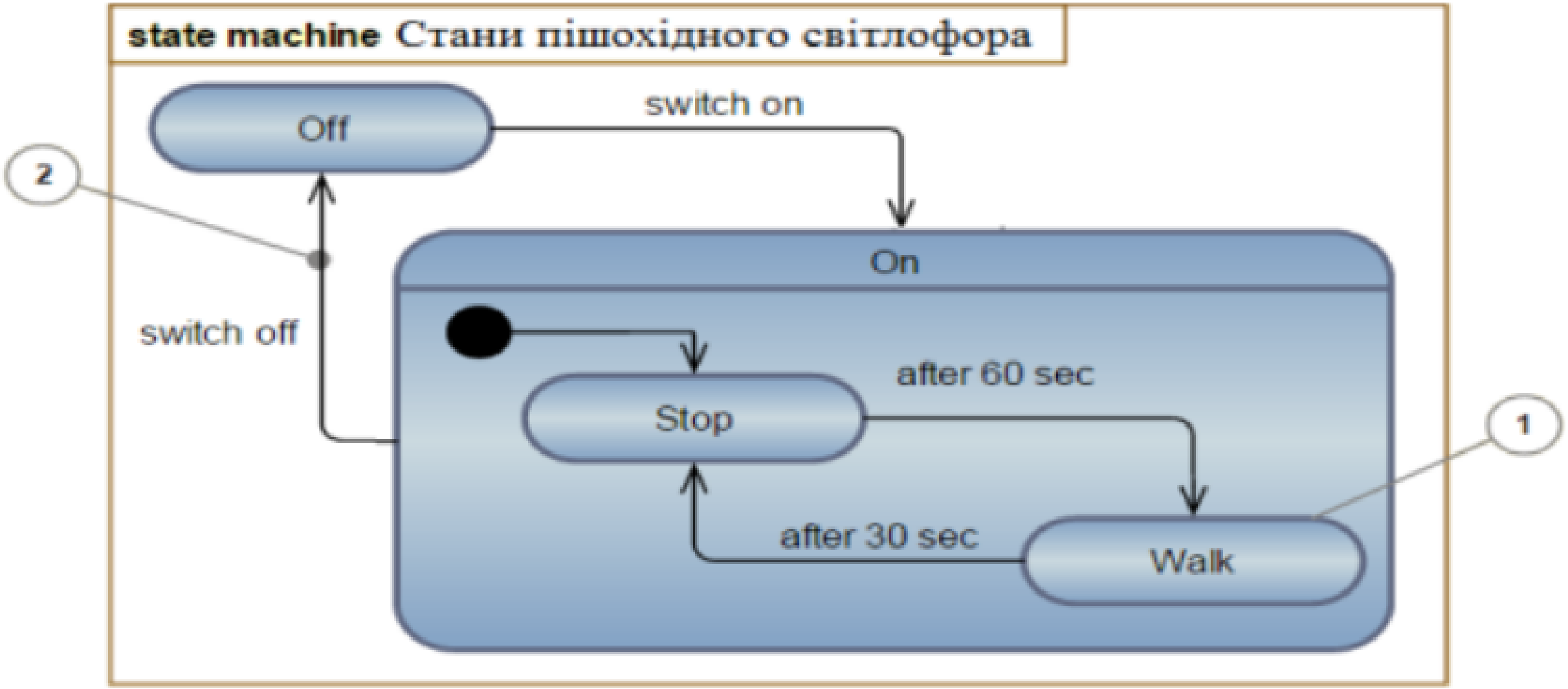


Рисунок 10.21 Нотація діаграми станів (1 – стани, 2 – переходи)

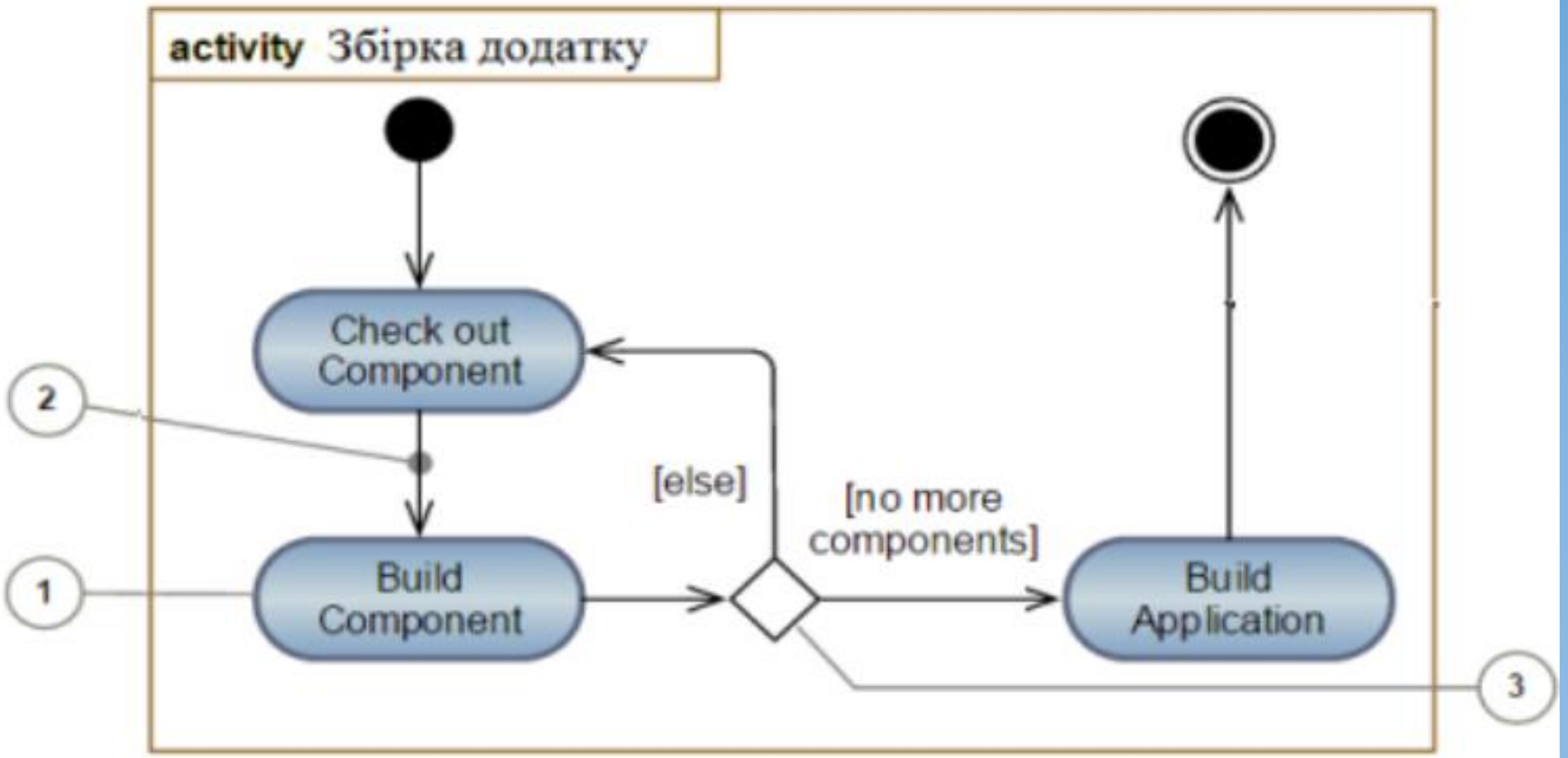
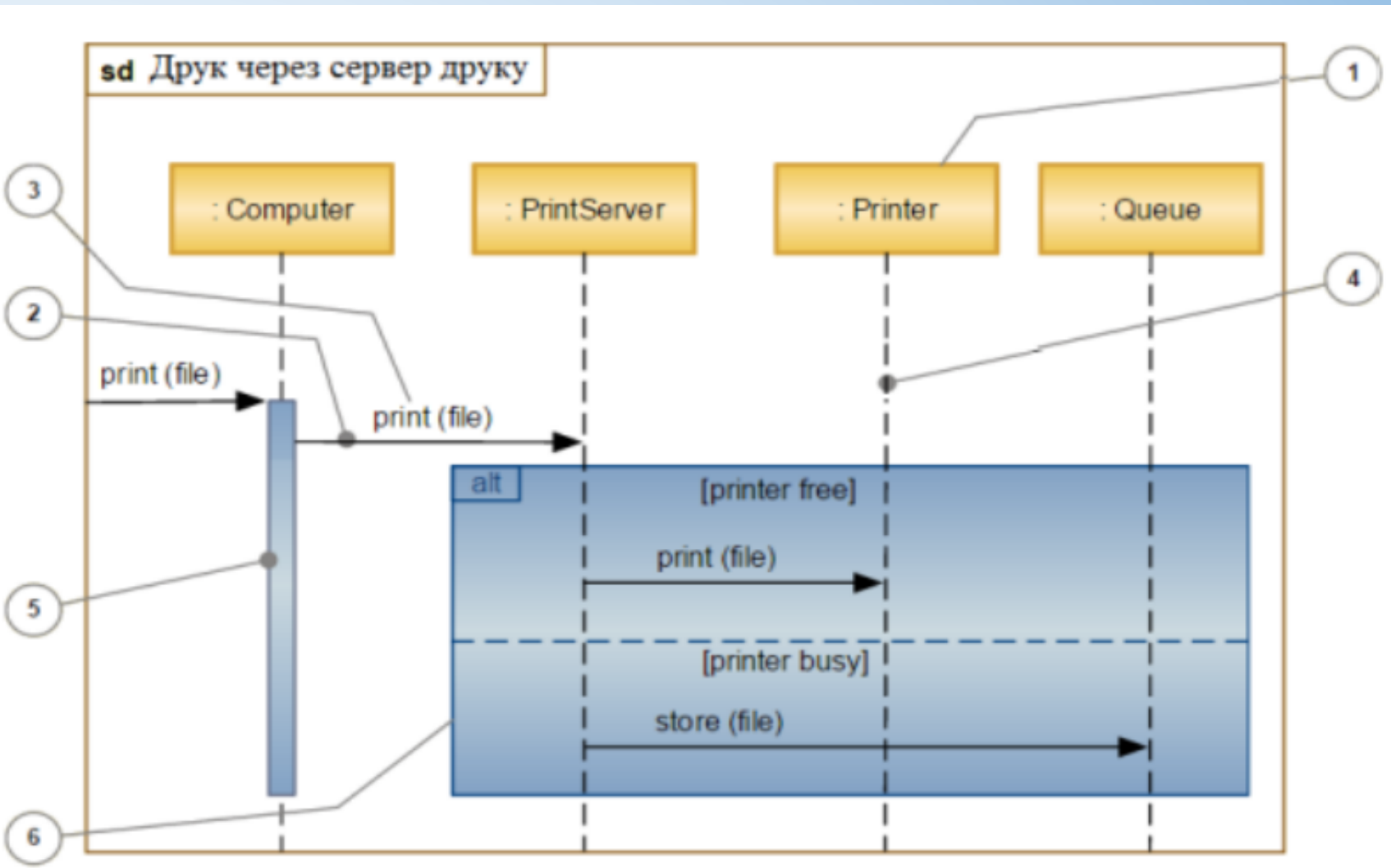


Рисунок 10.22 Нотація діаграми діяльності (1 – дія, 2 – переходи, 3 – розвилки, злиття, з'єднання, розгалуження)



1 – Екземпляри взаємодіючих класифікаторів
2 – Зв'язки
3 – Повідомлення
4 – Лінія життя (lifeline)
5 – Потік управління (execution occurrence)
6 – Складові кроки взаємодії (combined fragment)

Рисунок 10.23 Нотація діаграми послідовності

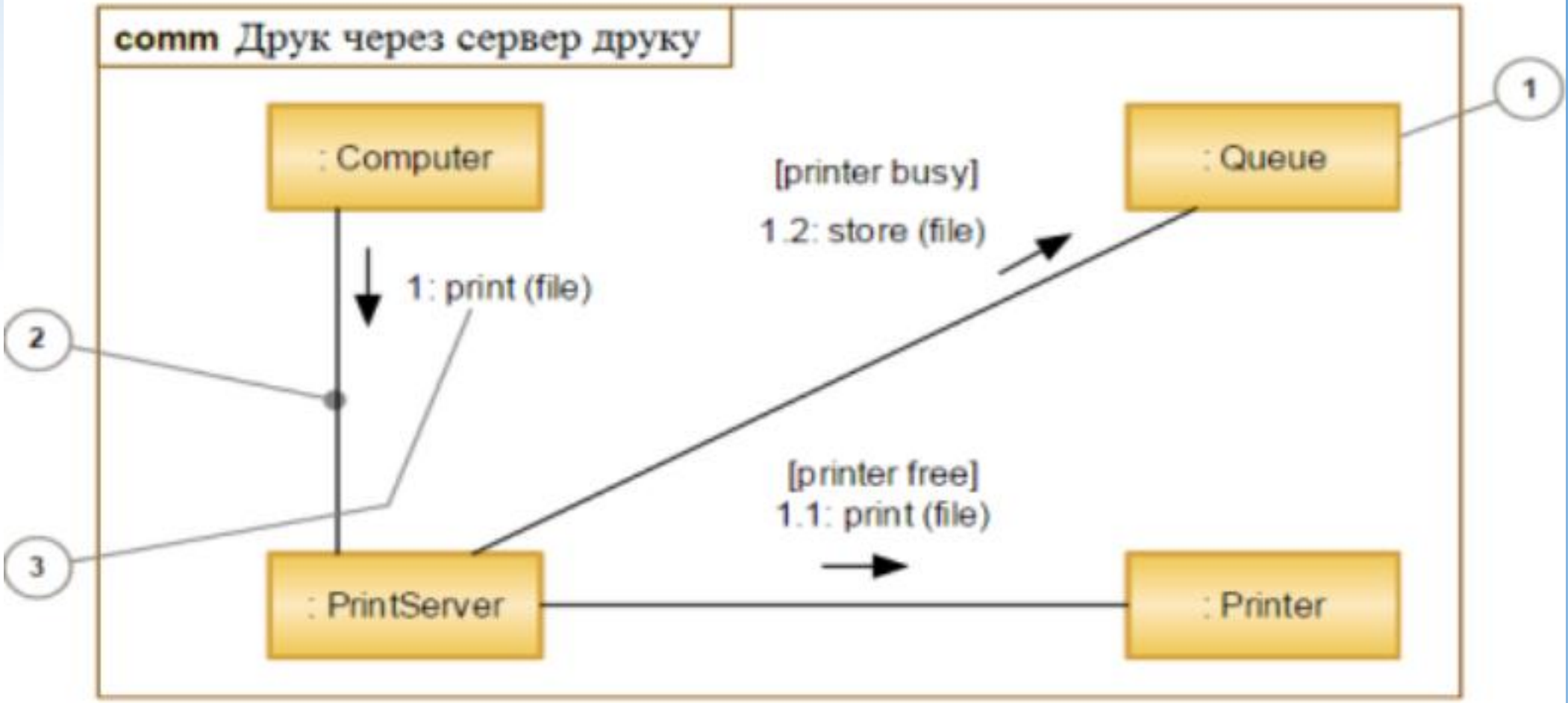


Рисунок 10.24 Нотація діаграми комунікації
(1 – екземпляри взаємодіючих класифікаторів, 2 – зв'язки,
3 – повідомлення)

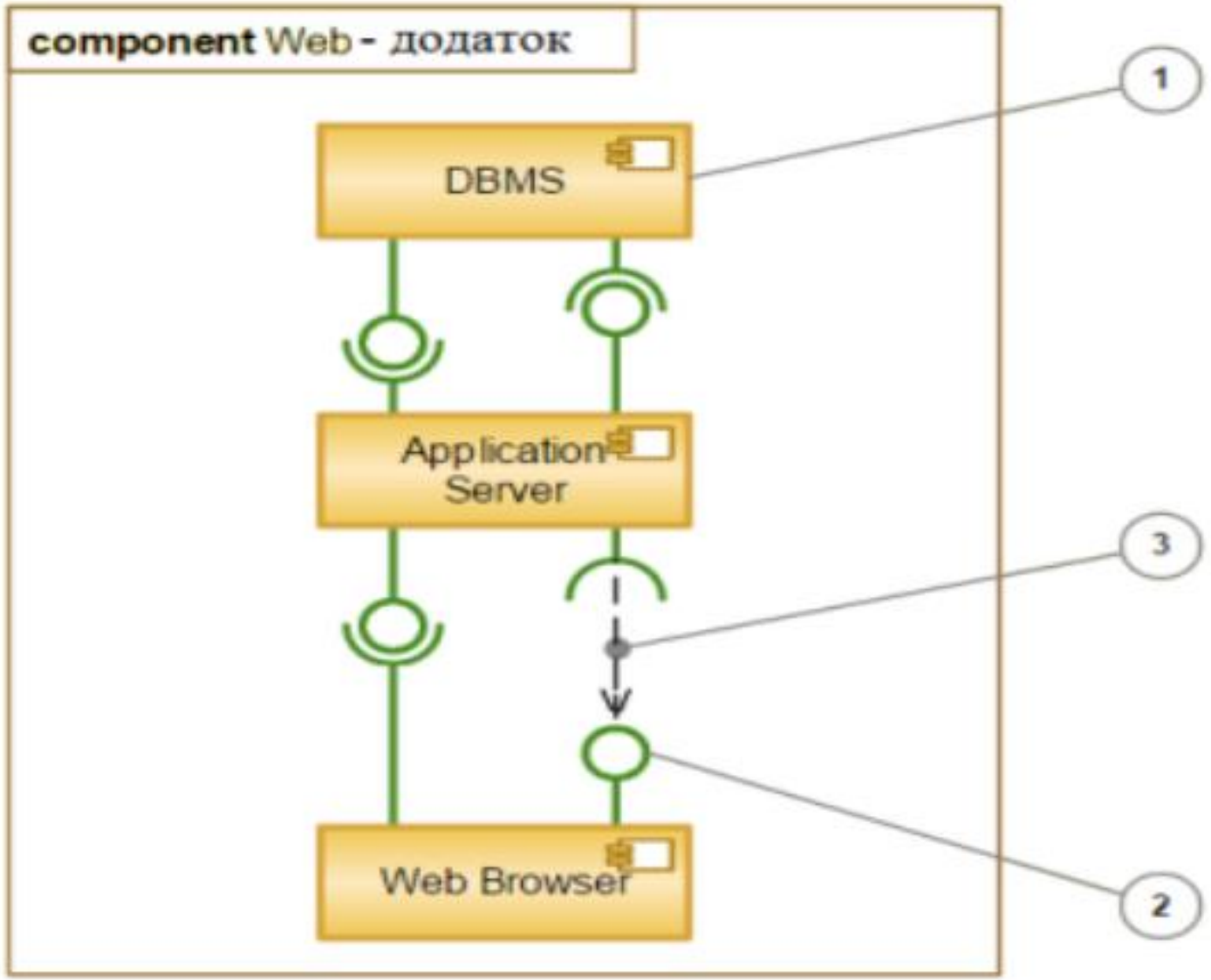
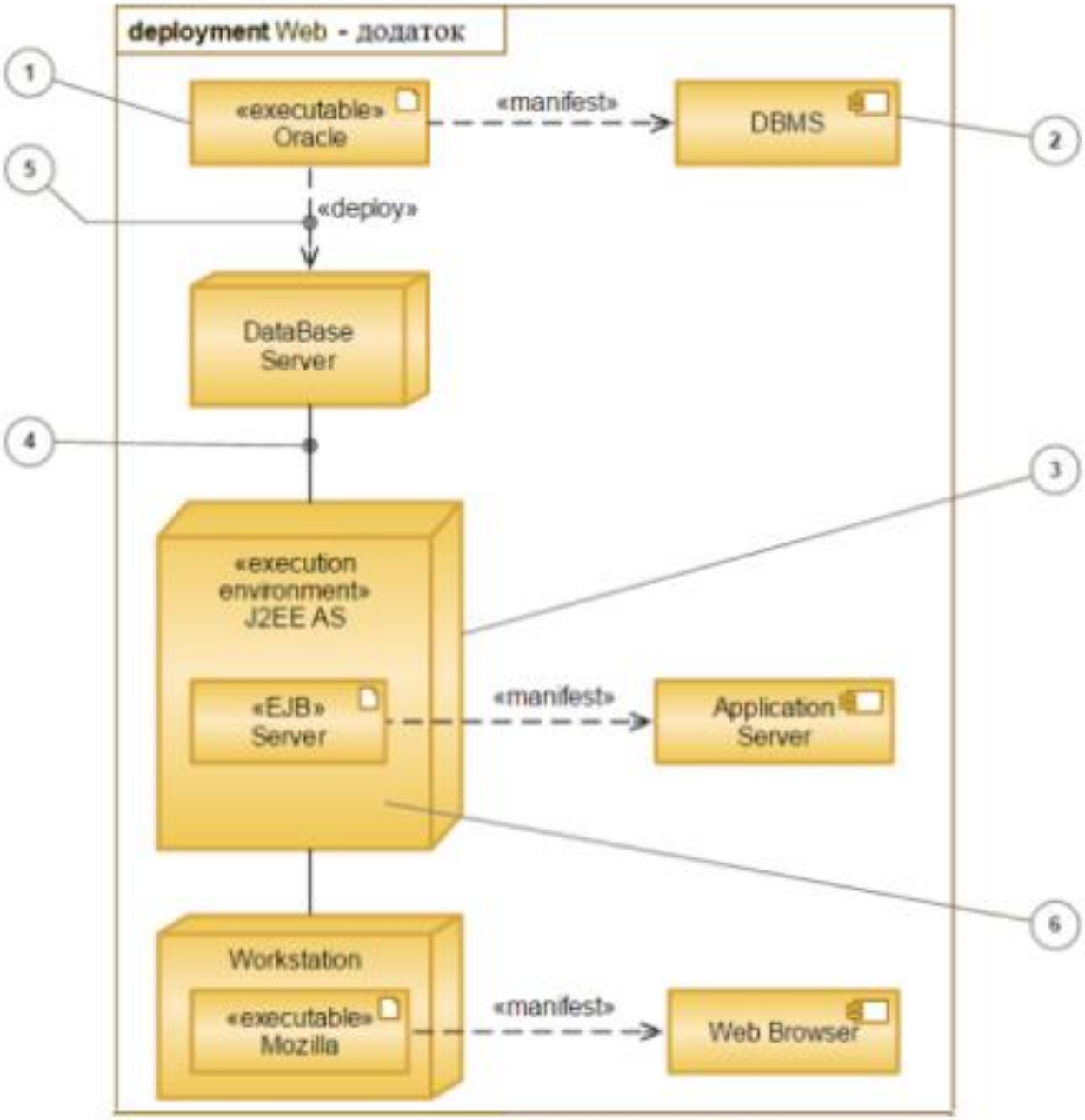


Рисунок 10.25 Нотація діаграми компонентів (1 – компоненти, 2 – інтерфейси, 3 – залежність між компонентами та інтерфейсами)



1 – Артефакт
2 – Компонент
3 – Вузол
4 – Асоціація між вузлами
5 – Відношення залежності «deploy»
6 – Фігура однієї сутності поміщається всередину фігури іншої сутності

Рисунок 10.26 Нотація діаграми розміщення

Рисунок 10.27

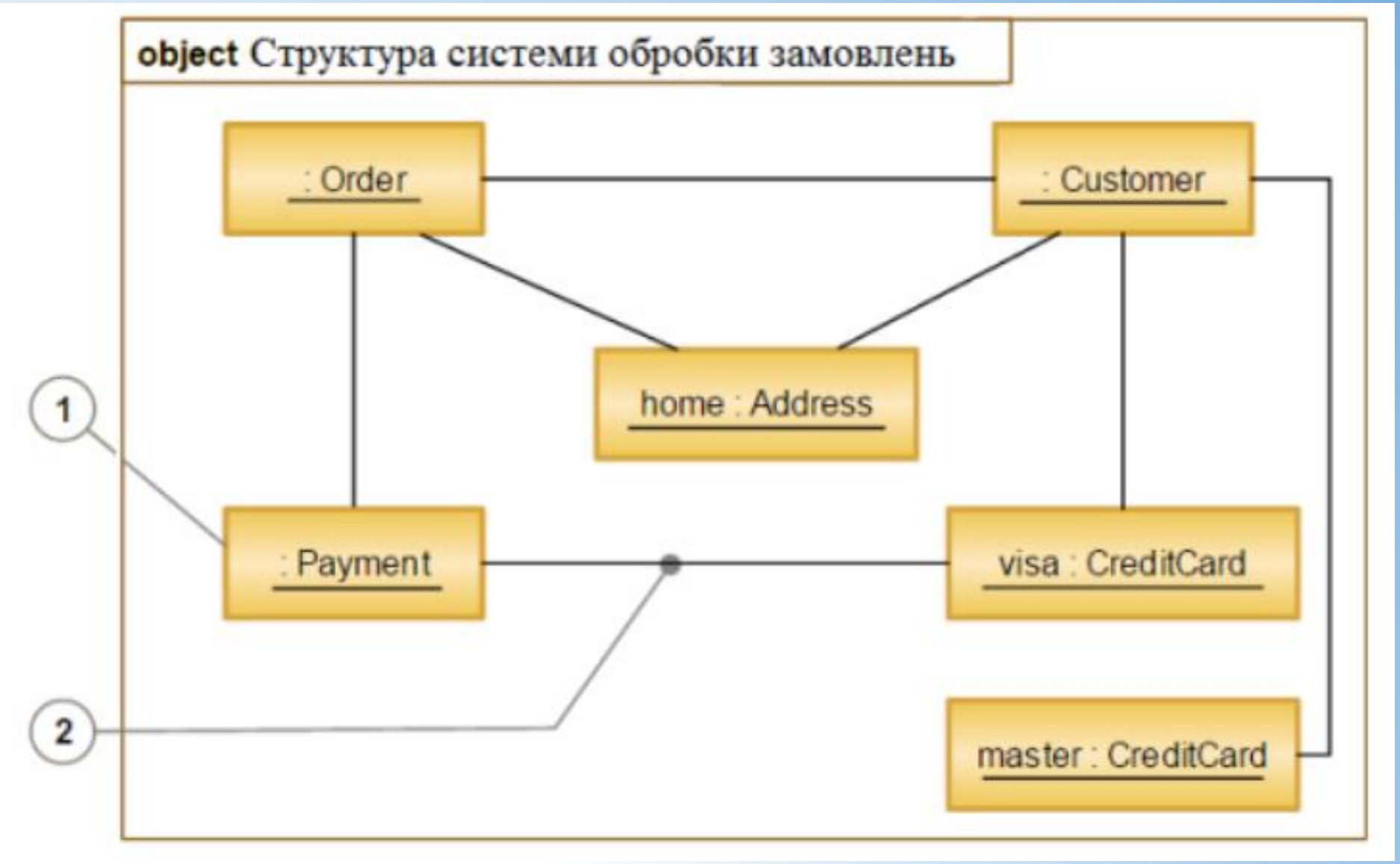
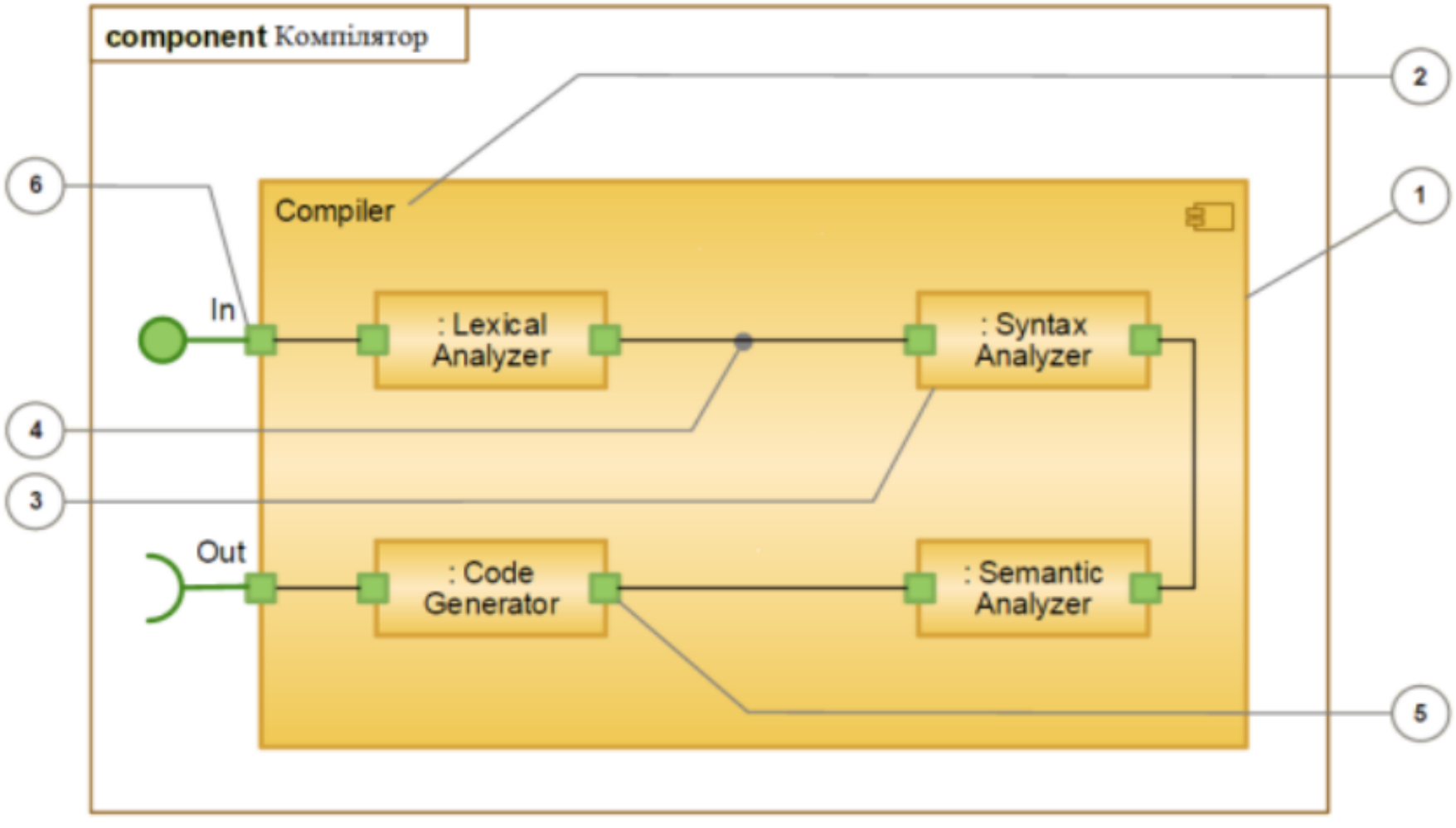


Рисунок 10.28 Нотація діаграми об'єктів
(1 – об'єкти (екземпляри класів), 2 – зв'язки (екземпляри асоціацій))



1 – Структурний класифікатор
2 – Ім'я класифікатора
3 – Екземпляр іншого класифікатора
4 – З'єднувач різних видів
5 – Порт
6 – Порт зв'язку із зовнішнім середовищем

Рисунок 10.29 Нотація діаграми внутрішньої структури

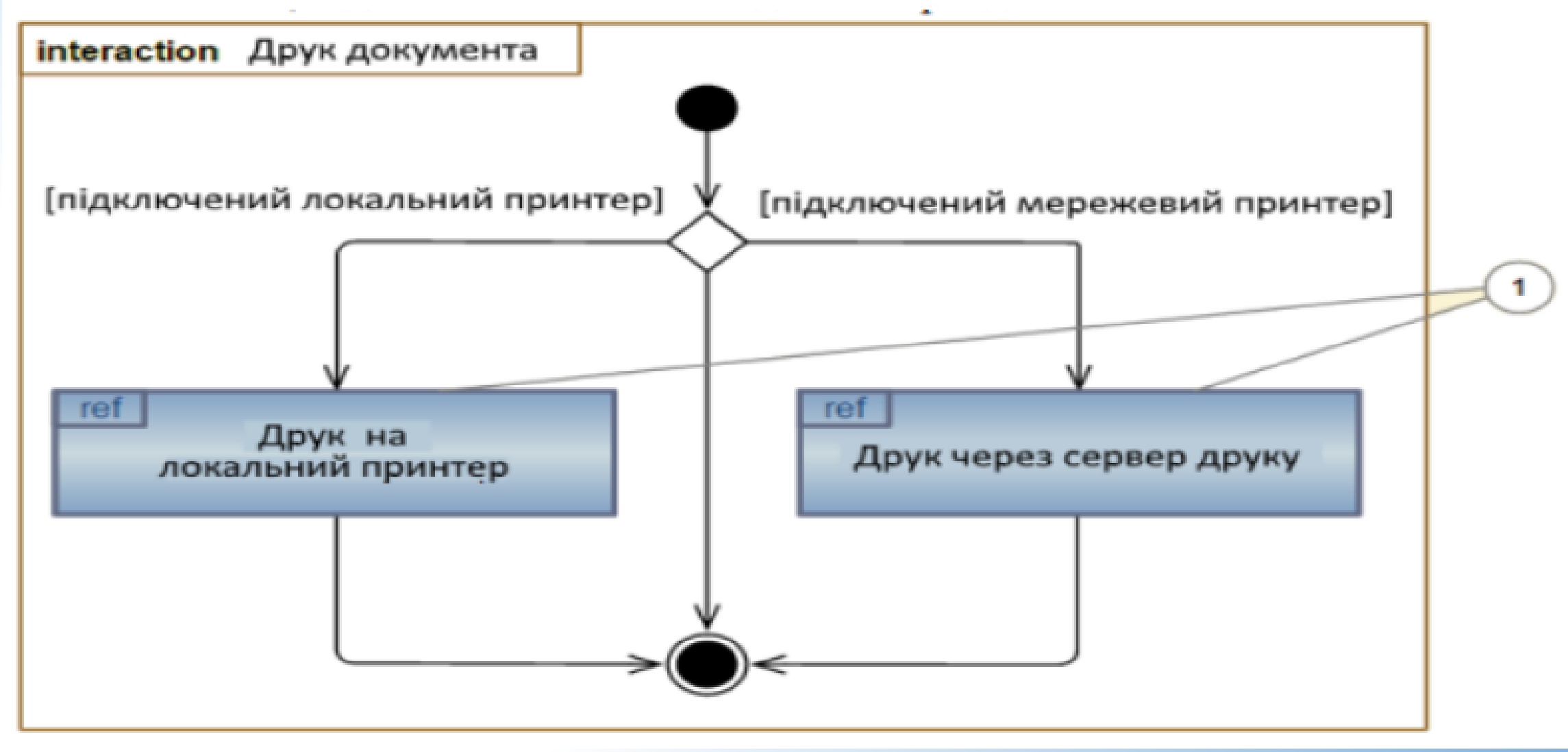


Рисунок 10.30 Нотація оглядової діаграми взаємодії (1 – посилання взаємодії)

Рисунок 10.31 Нотація діаграми синхронізації (1 – стани різних екземплярів класифікаторів, 2 – часова синхронізація екземплярів класифікаторів)

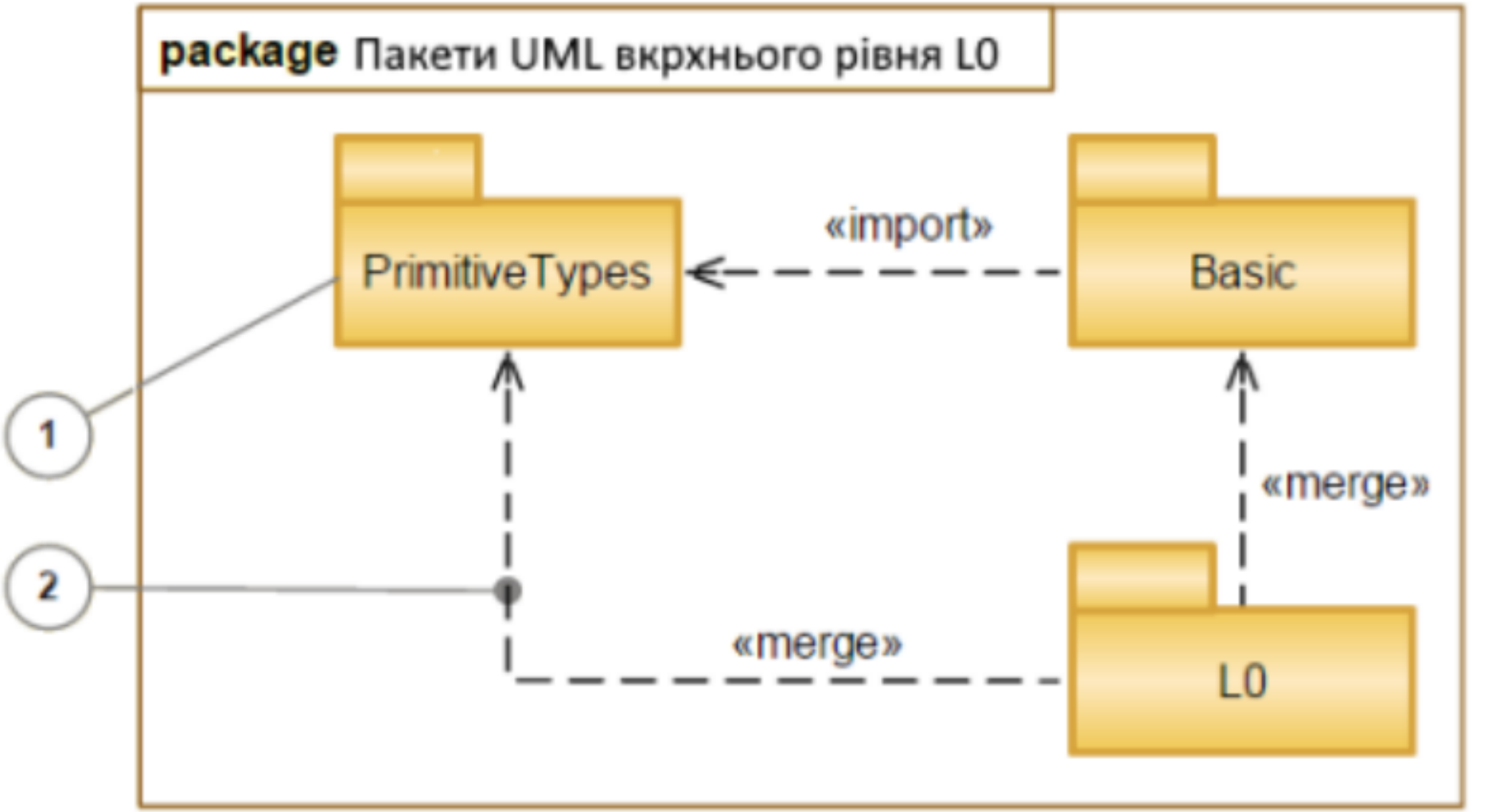


Рисунок 10.32 Нотація діаграми пакетів
(1 – пакети, 2 – залежності з різноманітними стереотипами)

ТЕМА №11. ПРОЄКТУВАННЯ ПОВЕДІНКОВИХ АСПЕКТІВ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

11.1 Діаграма варіантів використання (прецедентів)
11.2 Діаграма станів
11.3 Діаграма діяльності

11.1 ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ (ПРЕЦЕДЕНТІВ)

Діаграма варіантів використання – вихідне концептуальне подання (концептуальна модель системи) в процесі проєктування і розробки інформаційної системи.

Діаграма варіантів використання ***забезпечує*** опис функціонального призначення системи.

Діаграма варіантів використання ***дозволяє:***

- визначення спільних кордонів і контекстів модельованої предметної області;
- формулювання загальних вимог до функціональної поведінки іс
- розробку вихідної концептуальної системи
- підготовку вихідної документації

Суть діаграми ***полягає*** в поданні проєктованої системи у вигляді безлічі сутностей або акторів, які взаємодіють з системою за допомогою варіантів використання.

Варіант використання ***відображає*** набір дій, який здійснюється системою при взаємодії з актором.

Діаграма варіантів використання – ***граф спеціального виду***, який відображає конкретні варіанти використання, акторів і відношення між цими елементами.

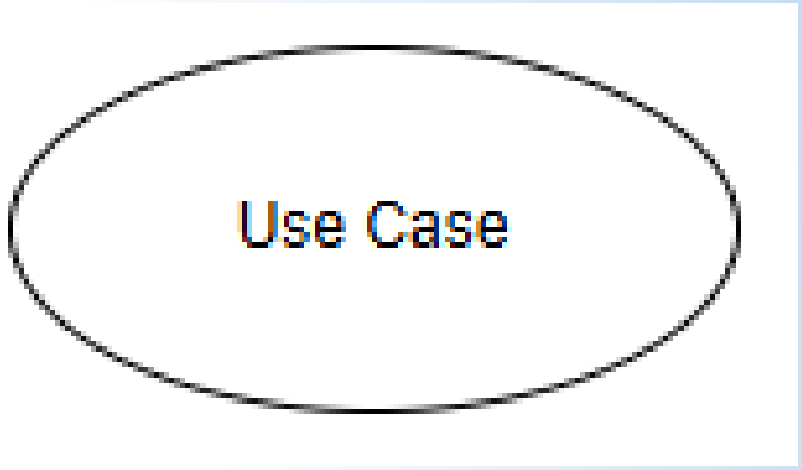
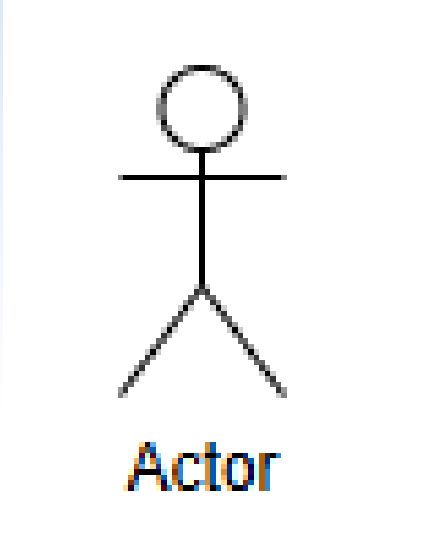


Рисунок 11.1 Графічне позначення варіанту використання



Actor – сутність, що взаємодіє з системою ззовні і використовує її функціональні можливості та яка служить для позначення узгодженої безлічі ролей, що грають користувачі в процесі взаємодії з проєктованою системою.

Рисунок 11.2 Графічне позначення Actor

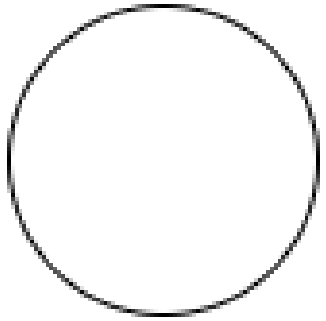
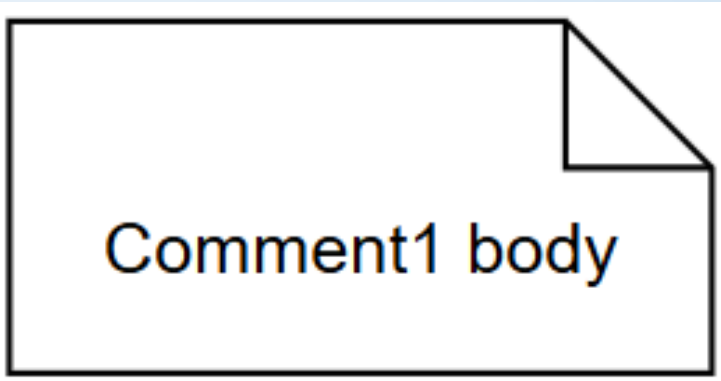


Рисунок 11.3 Графічне позначення інтерфейсів

Інтерфейси визначають сукупність операцій, які забезпечують необхідний набір сервісів або функціональності для акторів та містять тільки операції без вказівки особливостей їх реалізації.



Примітки призначені для включення в модель довільної текстової інформації, що має безпосереднє відношення до контексту розроблюваного проєкту.

Рисунок 11.4
Графічне позначення примітки

Рисунок 11.5

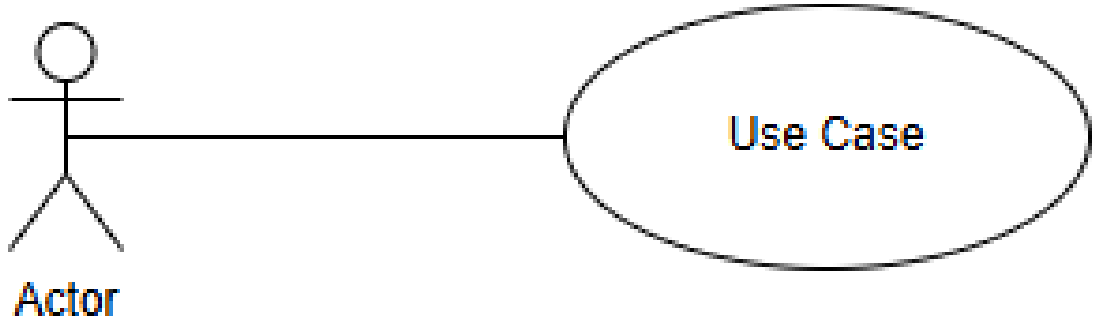


Рисунок 11.6 Графічне позначення асоціації

Відношення асоціації використовується при побудові всіх графічних моделей систем.

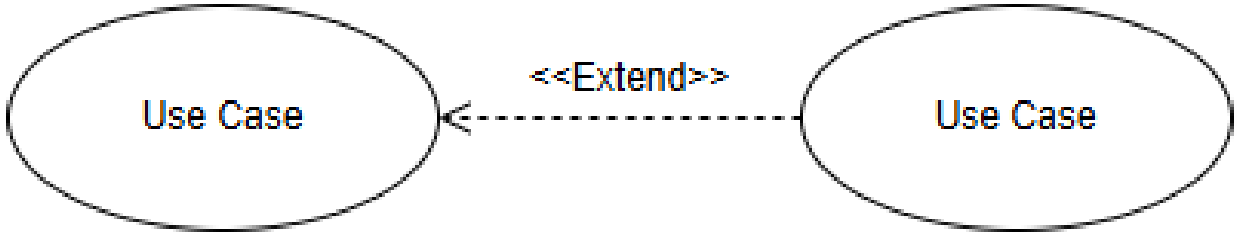


Рисунок 11.7 Графічне позначення розширення

Відношення розширення визначає взаємозв’язок екземплярів окремого випадку використання з більш загальним варіантом.

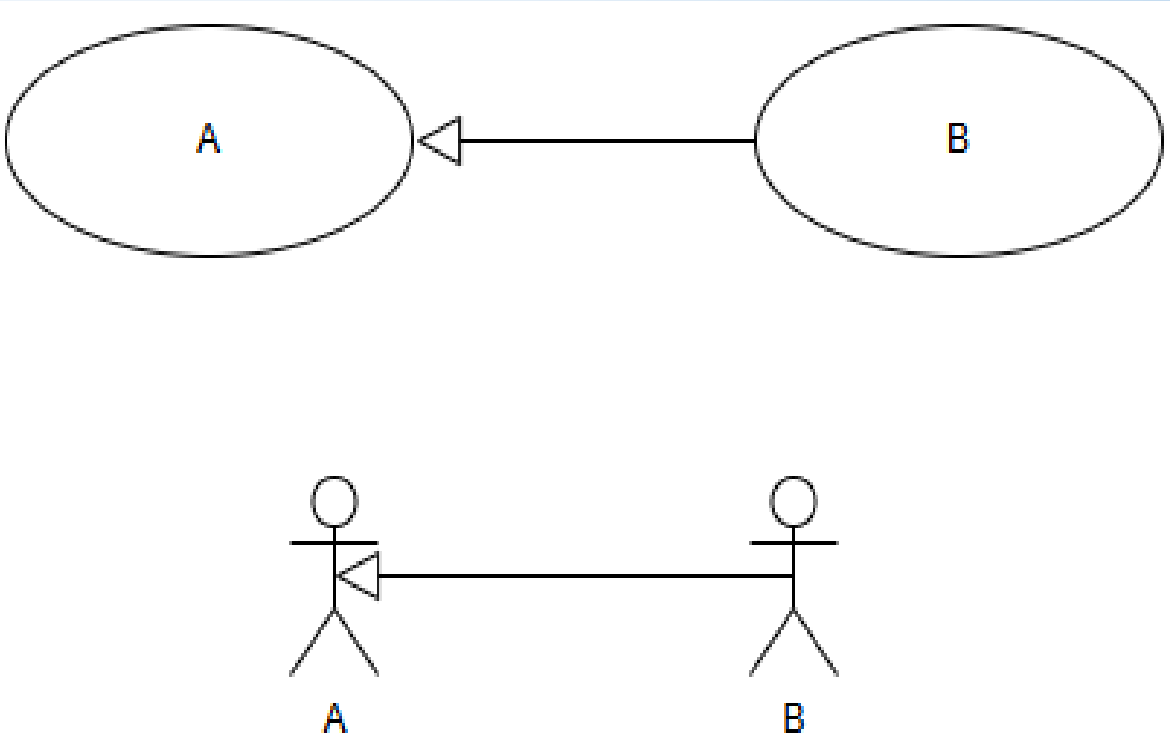


Рисунок 11.8 Графічне позначення узагальнення

Відношення узагальнення служить для вказівки того факту, що деякий варіант використання А може бути узагальнено до варіанту використання В.

Варіант А буде спеціалізацією варіанту В, причому В є предком або батьком по відношенню А, варіант А – нащадком по відношенню до варіанту використання В.

Властивості відносин узагальнення:

- дочірні прецеденти мають всі властивості предків;
- для одного предка може існувати декілька дочірніх прецедентів;
- може бути кілька батьків (множинне успадкування).

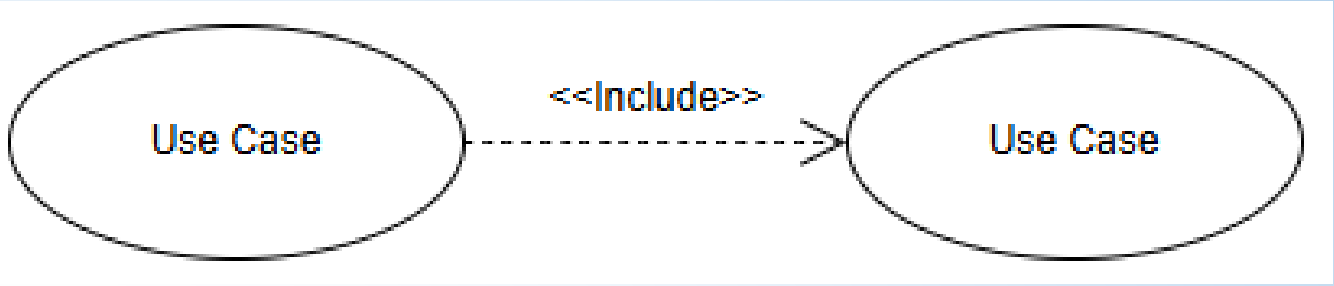


Рисунок 11.9 Графічне позначення включення

Відношення включення між двома варіантами використання вказує, що деяка задана поведінка для одного варіанта використання включається як складовий компонент послідовної поведінки іншого варіанту використання.

Рекомендована ***загальна кількість акторів у моделі – не більше 20, а варіантів використання – не більше 50***.

Будь-який з варіантів використання може бути підданий подальшій декомпозиції на безліч підваріантів використання окремих елементів, які утворюють вихідну сутність.

11.2 ДІАГРАМА СТАНІВ

Діаграма станів ***описує*** можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу.

Діаграма станів ***представляє*** динамічну поведінку сутностей, на основі специфікації їх реакції на сприйняття деяких конкретних подій.

Діаграма станів по суті є ***графом спеціального виду***, який представляє автомат. Вершинами графа є стани і деякі інші типи елементів автомата (псевдостани), а дуги графа служать для позначення переходів зі стану в стан.

Діаграми станів ***можуть бути*** вкладені одна в одну, створюючи вкладені діаграми більш детального представлення окремих елементів моделі.

Автомат – це пакет, в якому визначено множину понять, необхідних для подання поведінки модельованої сутності у вигляді дискретного простору з кінцевим числом станів і переходів.

Автомат ***описує*** поведінку окремого об’єкта у формі послідовності станів, які охоплюють всі етапи його життєвого циклу, починаючи від створення об’єкта і закінчуючи його знищенням.

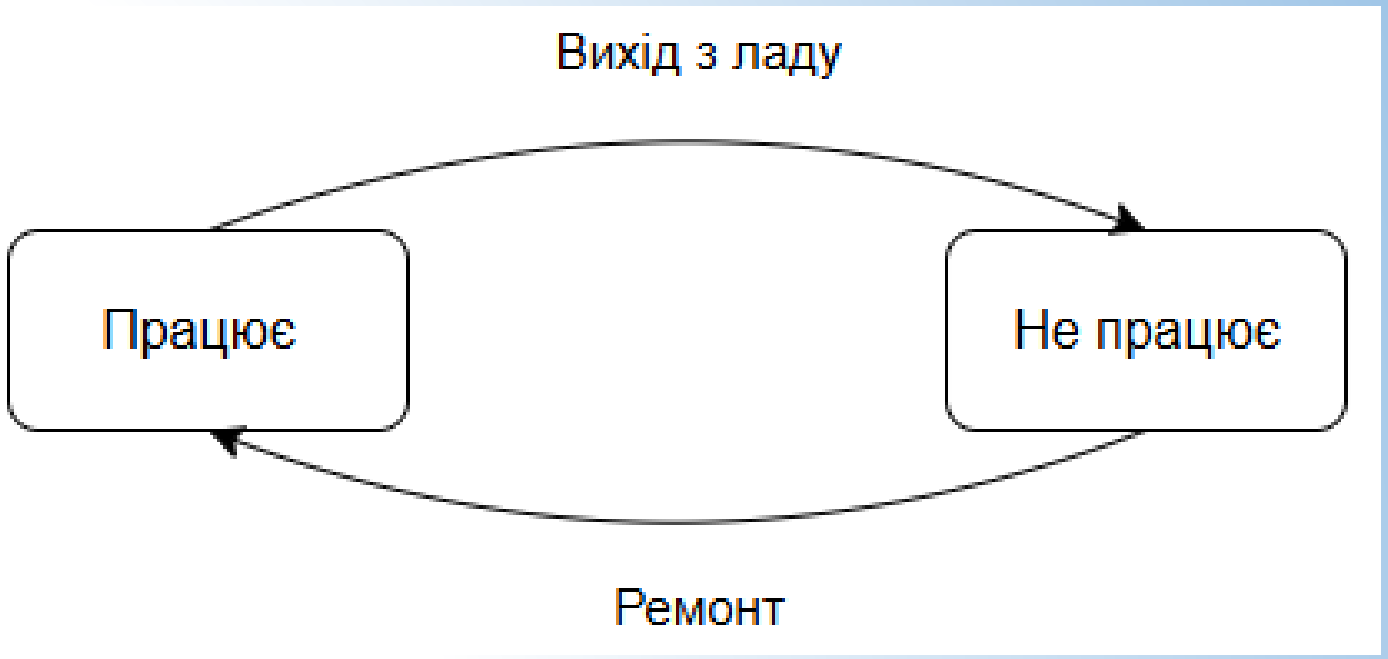


Рисунок 11.10 Приклад діаграми станів

Основними поняттями – ***стан*** і ***перехід***.

Головна відмінність – тривалість перебування системи в окремому стані суттєво перевищує час, який витрачається на перехід з одного стану в інший.

Автомат – динамічні аспекти системи, що моделюється, у вигляді орієнтованого графа, вершини якого відповідають станам, а дуги – переходам.

Властивість – ***виділення з усієї сукупності станів двох спеціальних: початкового і кінцевого станів.***

Вкладені автомати – підавтомати, які використовуються для внутрішньої специфікації процедур і функцій, що утворюють поведінку вихідного об’єкта.

Формалізм автомата заснований на виконанні ***обов'язкових умов***:

- Автомат не запам'ятовує історію переміщення зі стану в стан.
- У кожний момент часу автомат може перебувати в одному і тільки в одному зі своїх станів.
- Явно концепція часу не входить в формалізм автомата.
- Кількість станів автомата повинна бути обов'язково кінцевою (в мові UML розглядаються тільки кінцеві автомати), і всі вони повинні бути специфіковані явним чином.
- Граф автомата не повинен містити ізольованих станів і переходів.
- Автомат не повинен містити конфліктуючих переходів.

Стан – абстрактний метаклас, використовуваний для моделювання окремої ситуації, протягом якої має місце виконання деякої умови.

Стан ***може бути заданий*** у вигляді набору конкретних значень атрибутів класу чи об’єкта, при цьому зміна їх окремих значень буде відображати зміну стану модельованого класу або об’єкта.

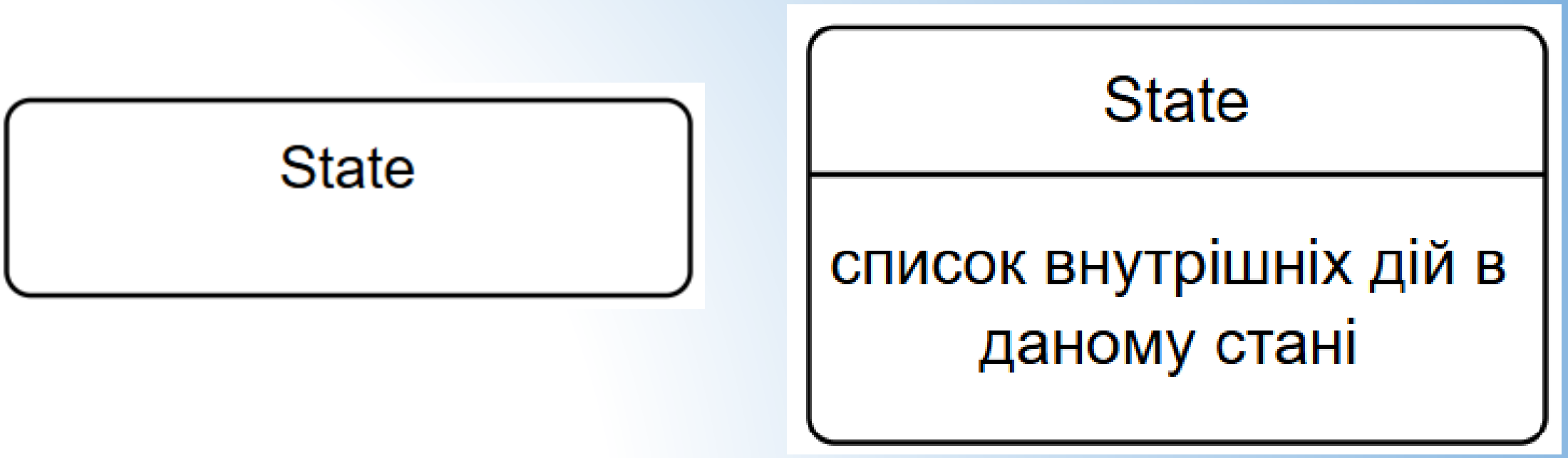


Рисунок 11.11 Графічне зображення станів

Атрибут: ***<мітка-дії ‘/’ вираз-дії>***

Мітка дії вказує на обставини або умови, за яких буде виконуватися діяльність, визначена виразом дії.

Вираз дії може використовувати будь-які атрибути та зв’язки, які належать області імен або контексту модельованого об’єкта.

- ***entry*** – ця позначка вказує на дію, специфіковану наступним за нею вираженням дії, що виконується в момент входу в даний стан (вхідна дія);
- ***exit*** – ця позначка вказує на дію, специфіковану наступним за нею вираженням дії, що виконується в момент виходу з цього стану (вихідна дія);
- ***do*** – ця мітка специфікує виконувану діяльність, яка виконується протягом усього часу, поки об’єкт знаходиться в даному стані, або до тих пір, поки не закінчиться обчислення, специфіковане наступним за нею вираженням дії;
- ***include*** – ця позначка використовується для звернення до підавтомату, при

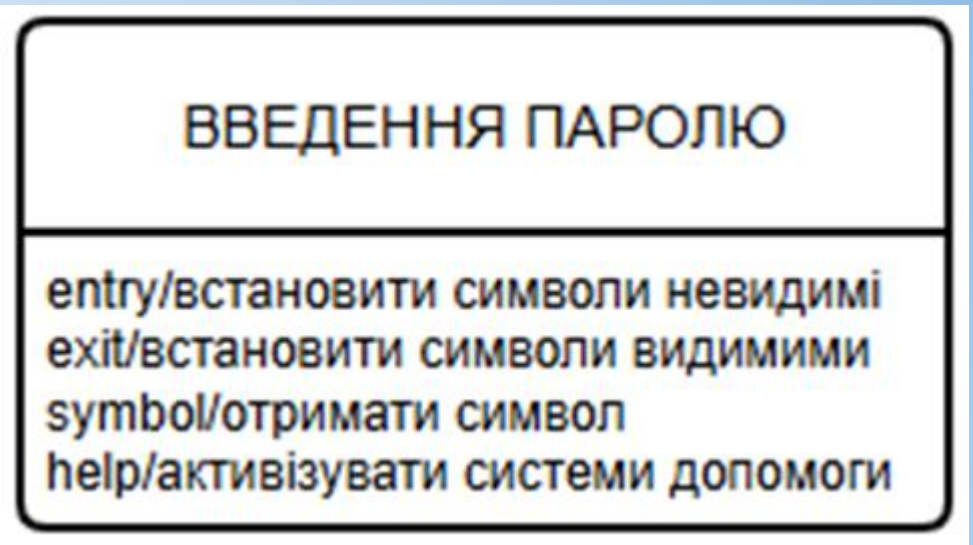


Рисунок 11.12 Приклад стану з непорожньою секцією внутрішніх дій

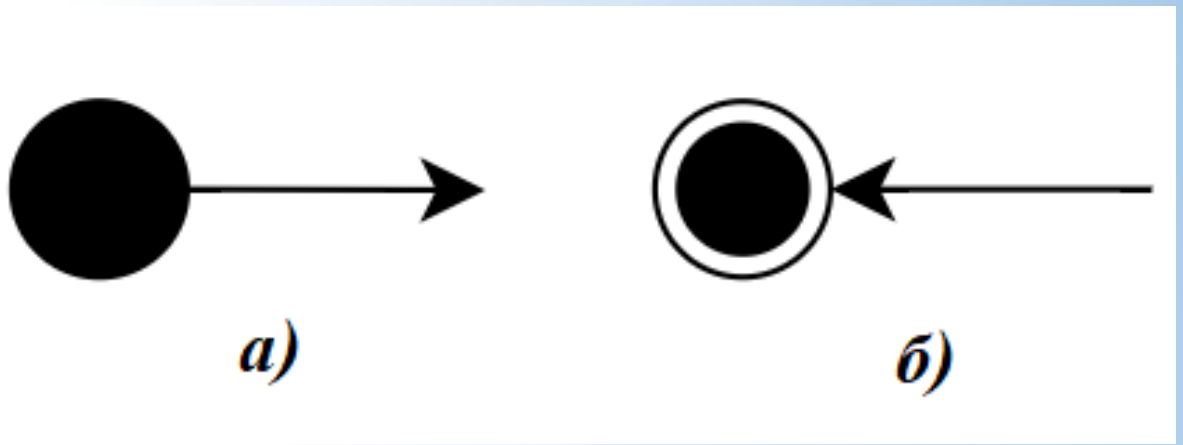


Рисунок 11.13 Графічне зображення початкового (а) і кінцевого (б) станів на діаграмі станів

Простий перехід – відношення між двома послідовними станами, яке вказує на факт зміни одного стану іншим.

Перехід ***здійснюється*** при настанні деякої події: закінчення виконання діяльності, отримання об’єктом повідомлення або прийом сигналу.

На переході ***вказується*** назва події та можуть зазначатися дії, вироблені об’єктом у відповідь на зовнішні події при переході з одного стану в інший.

Спрацювання переходу може залежати не тільки від настання деякої події, але і від виконання певної умови, званої сторожовою умовою.

Загальний формат тексту:

<сигнатура події>‘[‘<сторожова умова>‘]’<вираз дії>

Сигнатура подій:

<назва події>‘(‘<список параметрів, розділених комами>‘)’

Сторожова умова записується в прямих дужках після події-тригера і являє собою деякий булевий вираз, який иповинен приймати одне з двох взаємно виключаючих значень: «істина» або «брехня».

З контексту діаграми станів повинна явно слідувати семантика цього виразу, а для запису виразу може використовуватися синтаксис мови об’єктних обмежень.

Обчислення істинності сторожової умови відбувається тільки після виникнення асоційованої з ним події-тригера, ініціюючої відповідний перехід.

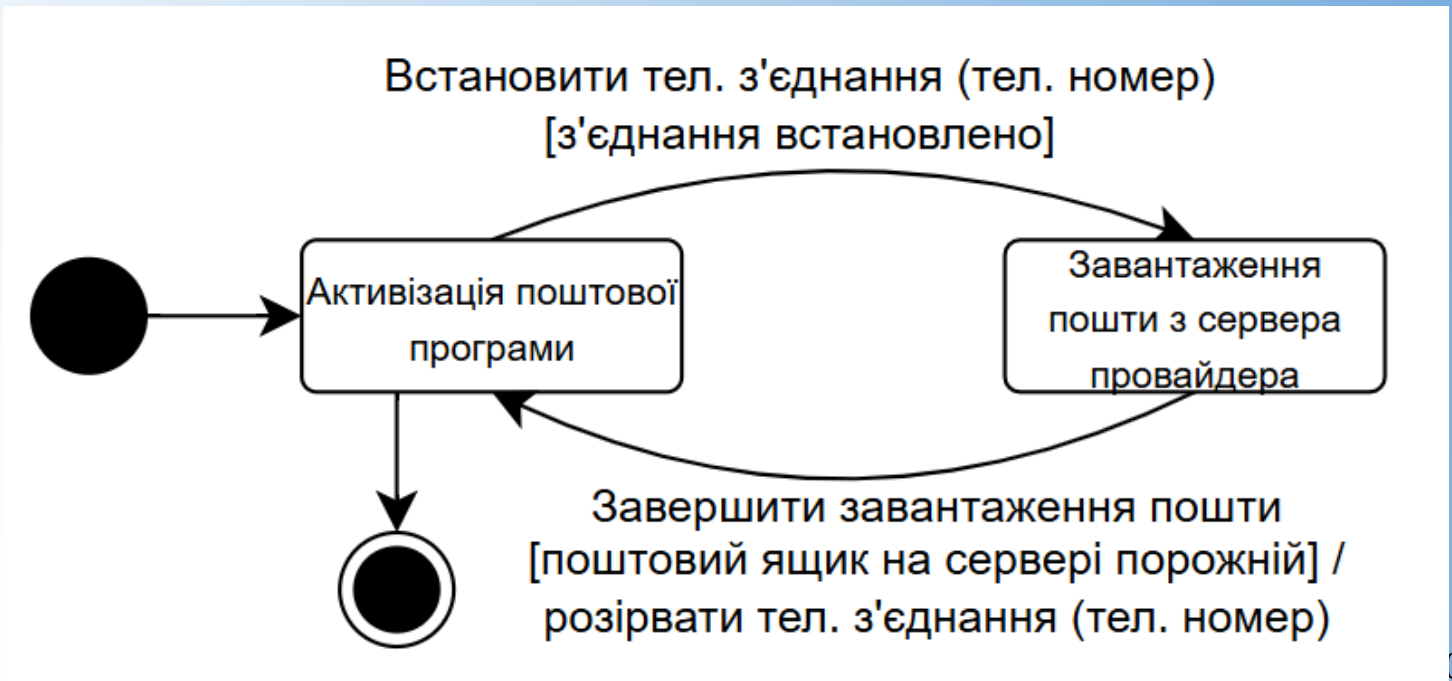


Рисунок 11.14 Діаграма станів для моделювання поштової програми-клієнта

Вираз дії виконується в тому і тільки в тому випадку, коли перехід спрацьовує, та являє собою атомарну операцію, виконувану відразу після спрацьовування відповідного переходу до початку яких би то не було дій в цільовому стані.

Вираз записується після знаку ***«/»*** в рядку тексту, приєднаного до відповідного переходу.

Вираз дії може містити цілий список окремих дій, розділених символом ***«,»*** або символом ***«;»***.

Обов'язкова вимога – усі дії зі списку повинні чітко відрізнятися між собою і слідувати в порядку їх запису.

На синтаксис запису виразів дії ***не накладається ніяких обмежень***.

Складений стан – такий складний стан, який складається з інших вкладених у нього станів.

Складений стан може містити два або більше паралельних підавтомати або кілька послідовних підстанів.

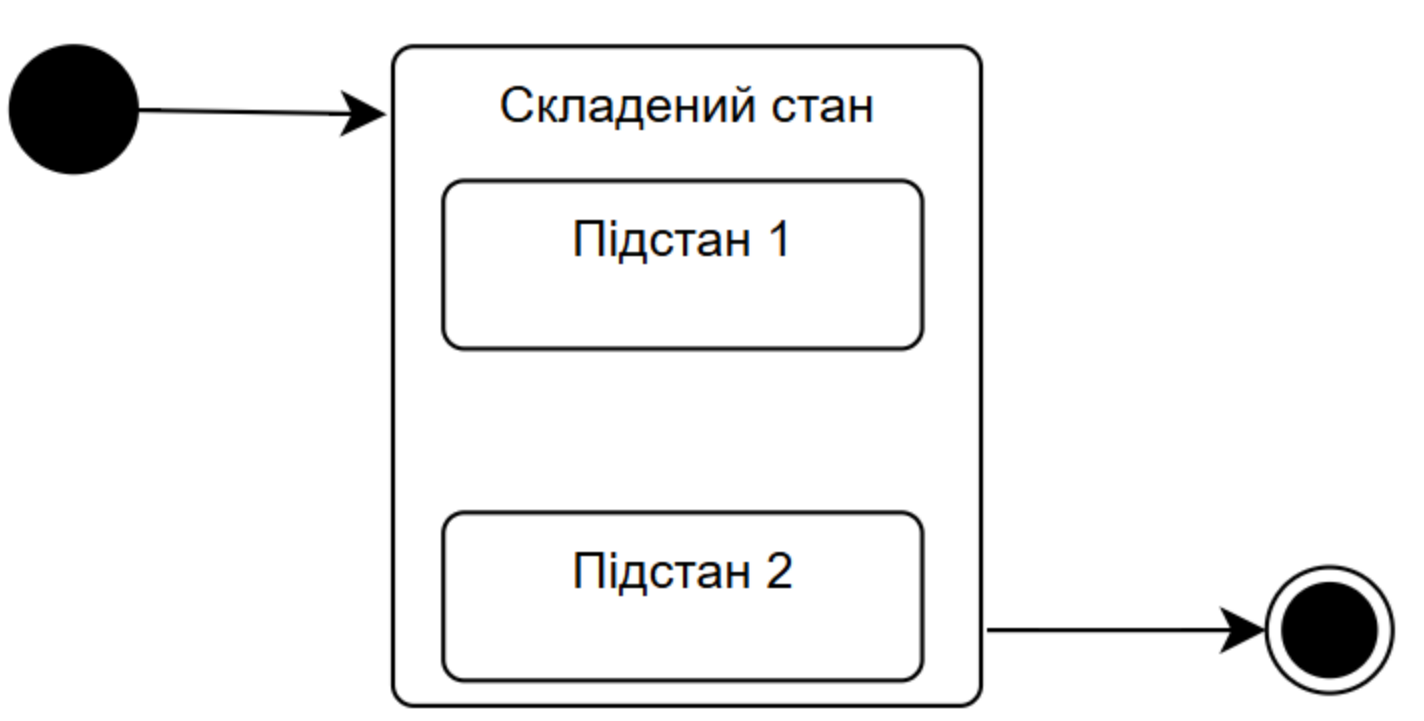


Рисунок 11.15 Графічне подання складеного стану з двома вкладеними в нього послідовними підстанами

Послідовні підстани використовуються для моделювання такої поведінки об’єкта, під час якої в кожен момент часу об’єкт може знаходитися в одному і тільки одному підстані. Поведінка об’єкта представляє собою послідовну зміну підстанів, починаючи від початкового і закінчуючи кінцевим підстаном. Складений стан може містити в якості вкладених підстанів початковий і кінцевий стани.

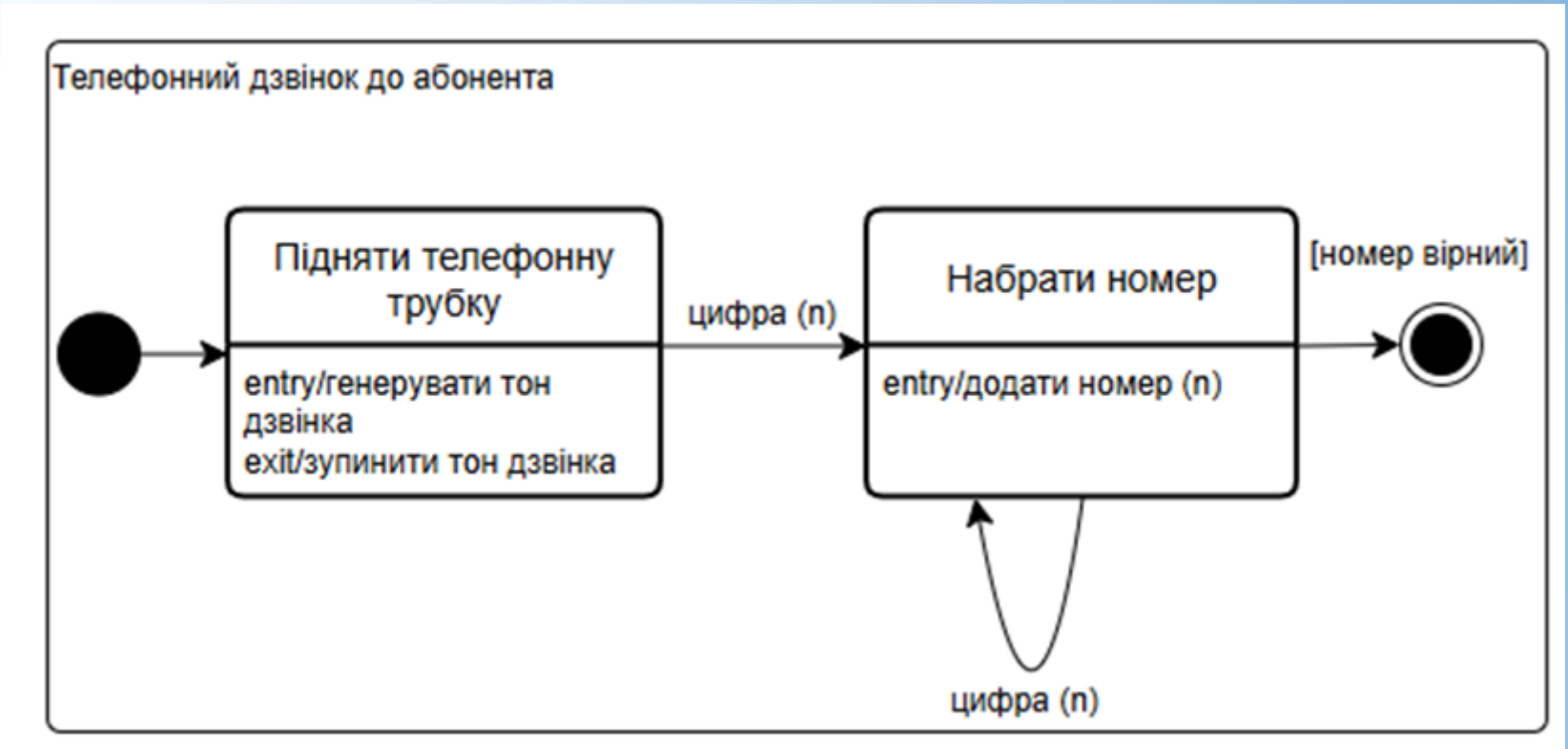


Рисунок 11.16 Приклад складеного стану з двома вкладеними послідовними підстанами

Паралельні підстани дозволяють специфікувати два і більше підавтомати, які можуть виконуватися паралельно всередині складеної події.

Окремі паралельні підстани можуть складатися з декількох послідовних підстанів, тому за визначенням об’єкт може знаходитися тільки в одному з послідовних підстанів підавтомату.

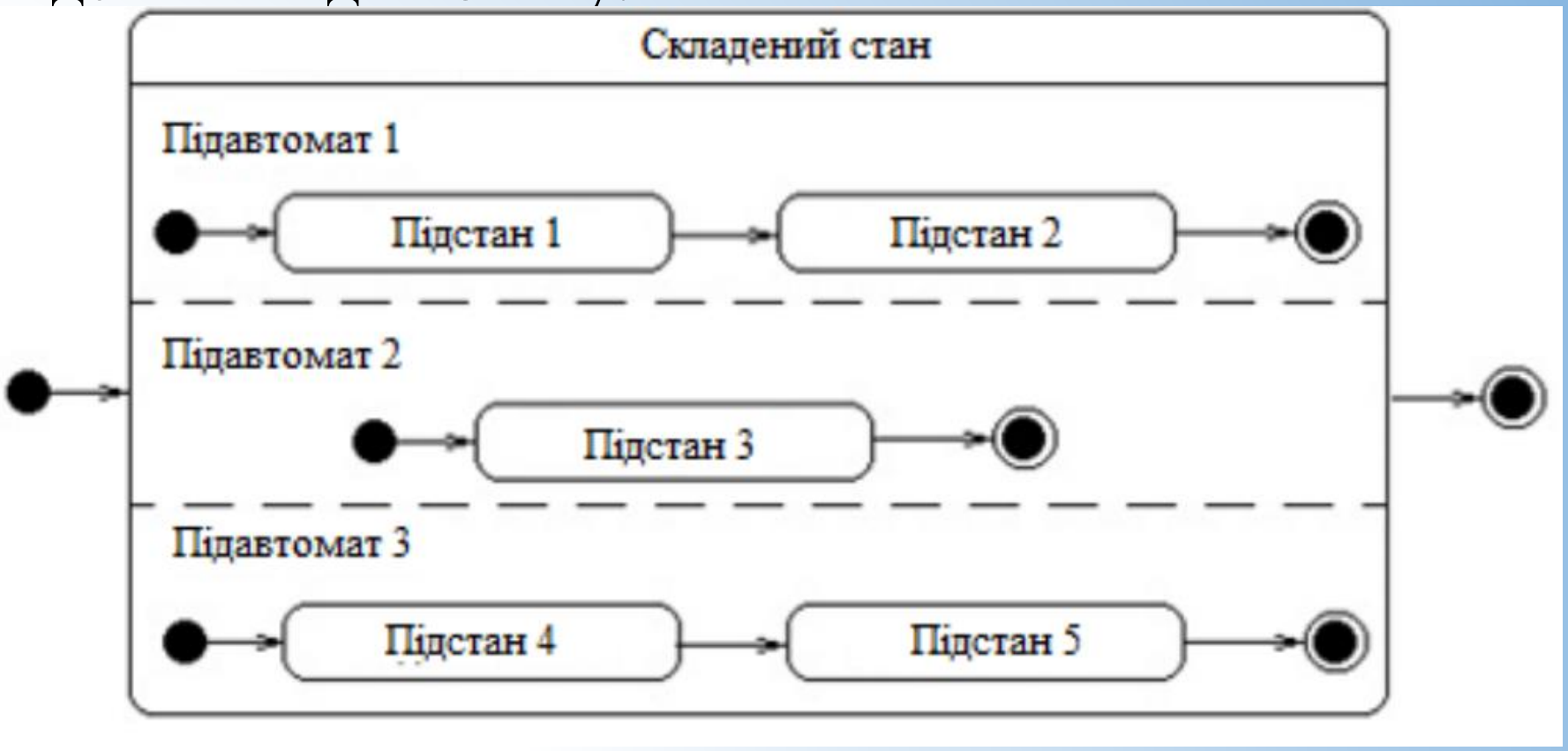


Рисунок 11.17 Графічне зображення складеного стану з вкладеними паралельними підстанами

У деяких випадках буває бажано приховати внутрішню структуру складеного стану, тому допускається не розкривати на вихідній діаграмі станів даний складений стан, а вказати у правому нижньому куті спеціальний символ-піктограму.

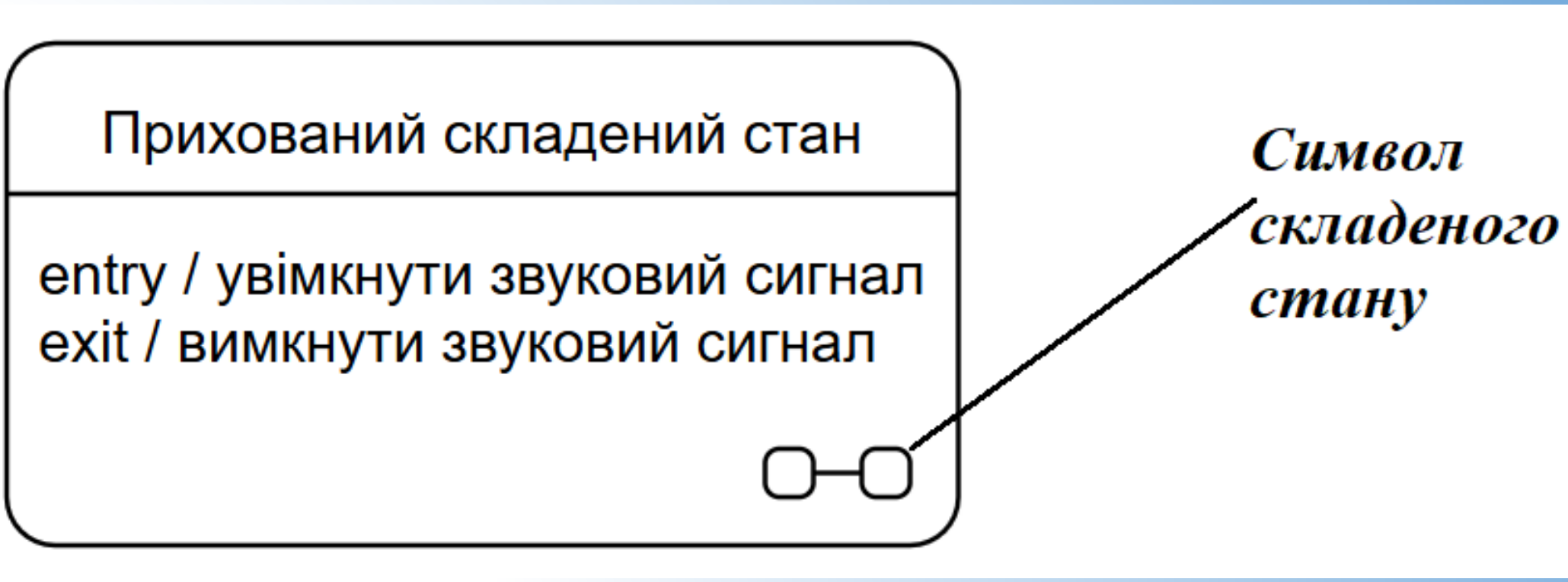


Рисунок 11.18 Складений стан з прихованою внутрішньою структурою та спеціальною піктограмою (позначкою наявності підстанів)

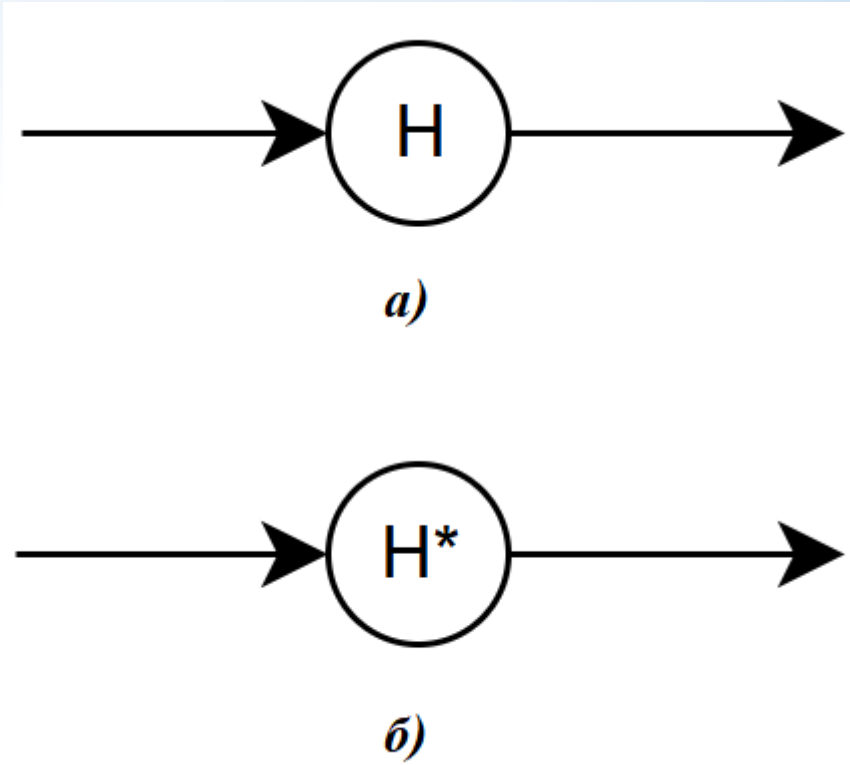


Рисунок 11.19 Графічне зображення недавнього (а) і давнього (б) історичного стану

Історичний стан застосовується в контексті складеного стану та використовується для запам'ятовування того з послідовних підстанів, який був поточним в момент виходу з складного стану.

Існує два різновиди історичного стану: ***недавній*** і ***давній.***

Недавній історичний стан – перший підстан в складеному стані, і перехід ззовні в цей складений стан повинен вести безпосередньо в цей історичний стан. При першому переході в недавній історичний стан він замінює собою початковий стан підавтомата.

Давній історичний стан служить для запам'ятовування всіх підстанів будь-якого рівня вкладеності для поточного підавтомата.

Складні переходи на діаграмі стані

Переходи між паралельними станами

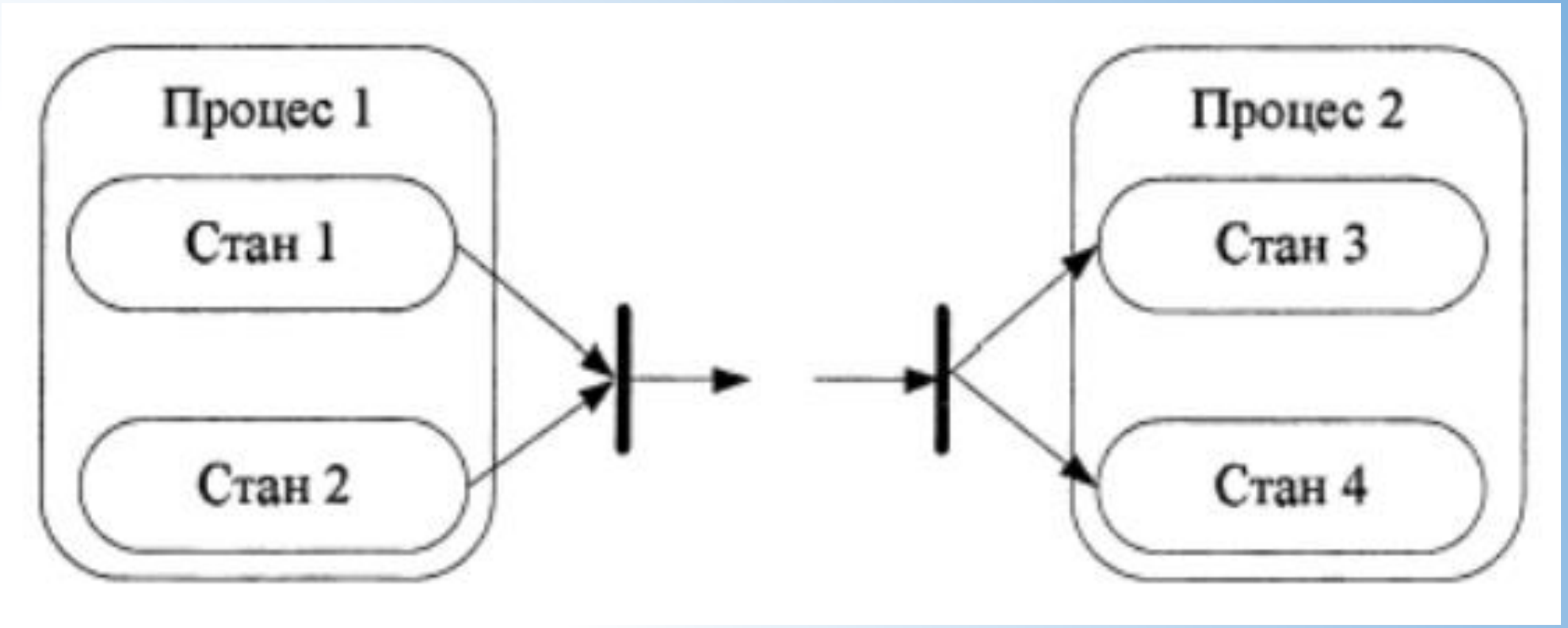


Рисунок 11.20 Графічне зображення паралельного переходу з паралельних станів (а) і паралельного переходу в паралельні стани (б)

Переходи між складовими станами

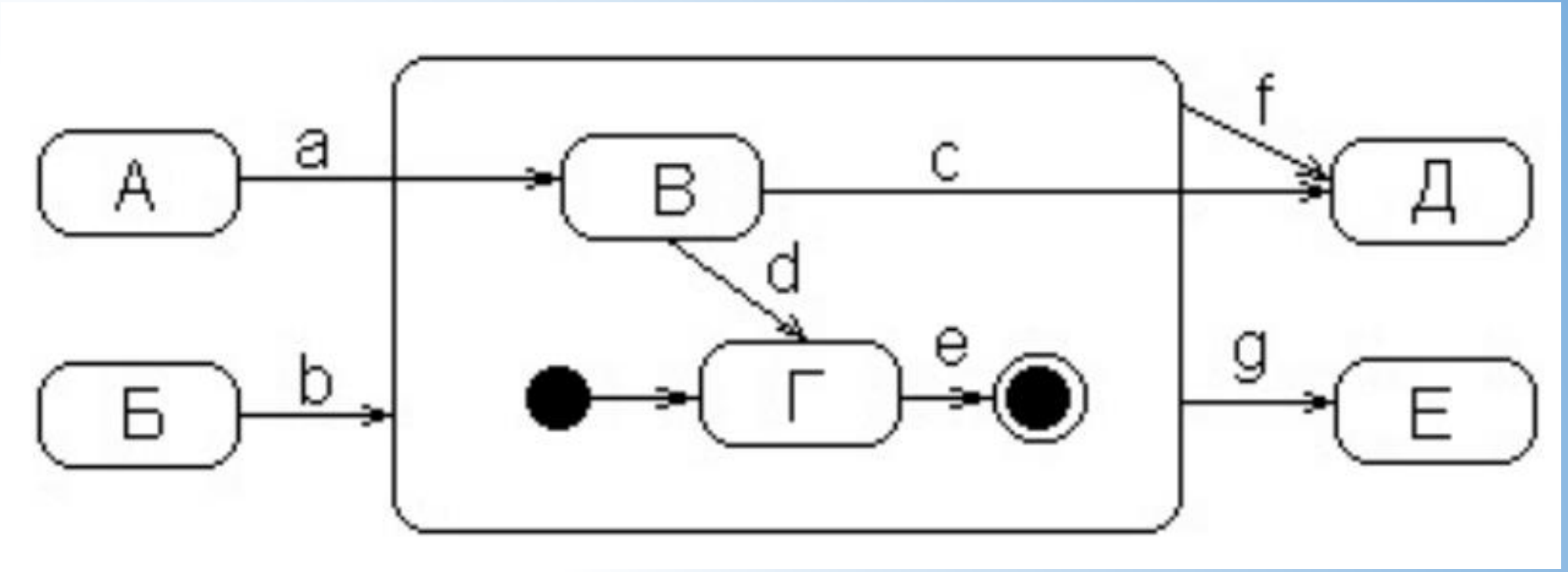


Рисунок 11.21 Різні варіанти переходів в (з) складений стан

Синхронізуючі стани

Рисунок 11.22 Діаграма станів для прикладу з будівництвом будинку

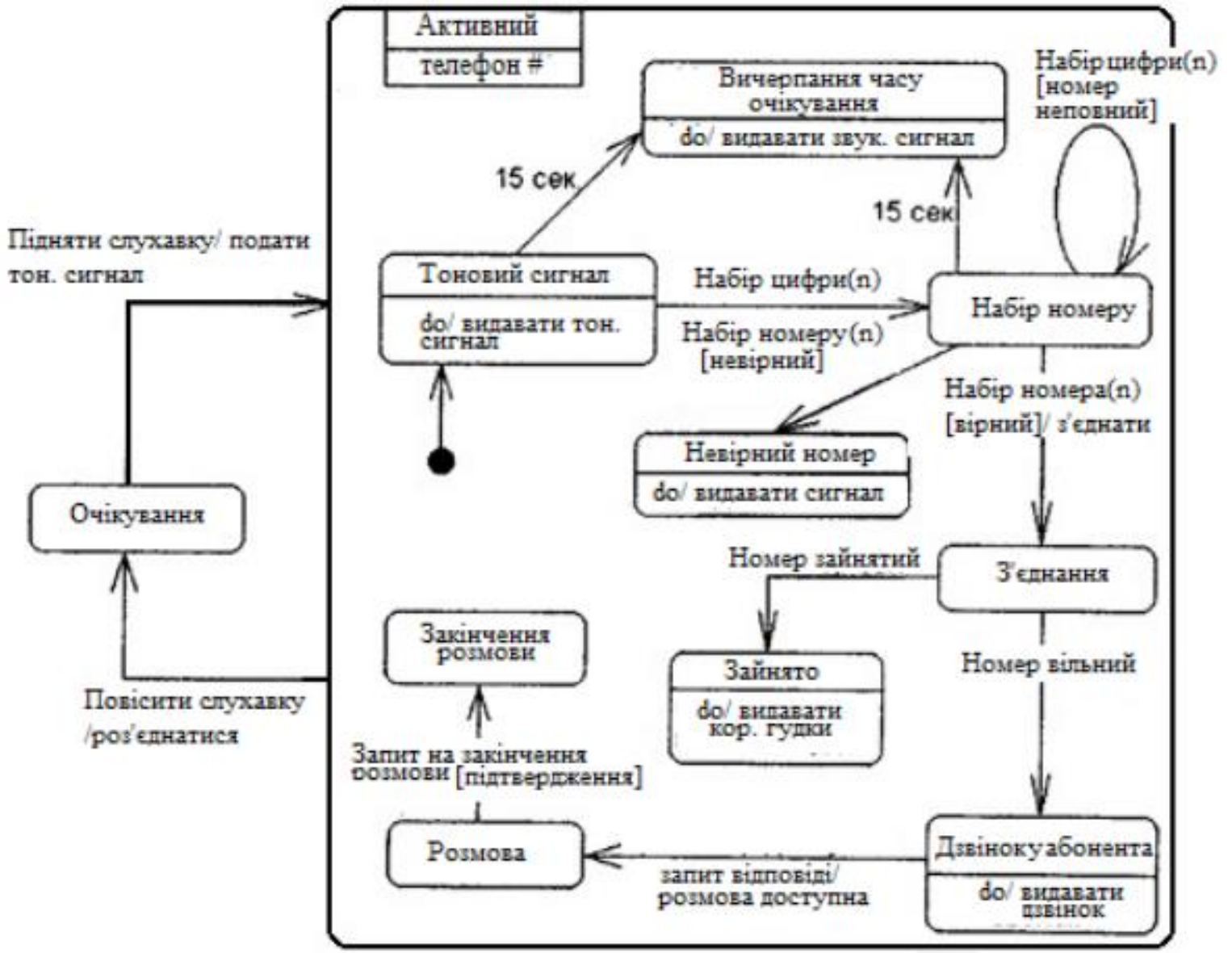


Рисунок 11.23 Діаграма станів процесу функціонування телефонного апарату

За своїм ***призначенням*** діаграма станів не є обов'язковим поданням до моделі і як би «приєднується» до того елемента, який, за задумом розробників, має нетривіальне поводження протягом свого життєвого циклу.

При виділенні станів і переходів ***слід пам'ятати***, що тривалість спрацьовування окремих переходів повинна бути істотно меншою, ніж знаходження модельованого об'єкта у відповідних станах.

Кожний з станів ***має характеризуватися*** певною стійкістю в часі.

Усі переходи ***повинні бути*** явно специфіковані.

Необхідно постійно стежити, щоб об'єкт в кожний момент міг перебувати тільки в єдиному стані.

Слід ***обов'язково перевіряти***, що ніякі два переходи з одного стану не можуть спрацювати одночасно.

Використання історичних станів ***виправдано*** в тому випадку, коли необхідно організувати обробку виняткових ситуацій (переривань) без втрати даних або виконаної роботи.

Кожен з підавтоматів ***може мати*** тільки один історичний стан.

11.3 ДІАГРАМА ДІЯЛЬНОСТІ

Діаграми діяльності використовуються для моделювання процесу виконання операцій.

Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує тільки при завершенні цієї операції в попередньому стані.

Графічно діаграма діяльності ***представляється*** у формі графа діяльності, вершинами якого є стани дії, а дугами – переходи від одного стану дії до іншого.

Основним напрямком використання діаграм діяльності є візуалізація особливостей реалізації операцій класів, коли необхідно представити алгоритми їх виконання.

Стан дії є спеціальним випадком стану з деякою вхідною дією та принаймні одним переходом, що виходить зі стану, тому він неявно передбачає, що вхідна дія вже завершилася.

Стан дії ***не може мати*** внутрішніх переходів, оскільки він є елементарним.

Використання стану дії ***полягає*** в моделюванні одного кроку виконання алгоритму (процедури) або потоку управління.

а) Звичайна дія *б) Вираз*

Рисунок 11.24 Графічне зображення стану дії

Рисунок 11.25 Графічне зображення позначки наявності піддіяльності

Кожна діаграма діяльності повинна мати єдиний початковий і єдиний кінцевий стани. Кожна діяльність починається в початковому стані та закінчується в кінцевому стані, а діаграму діяльності прийнято розташовувати таким чином, щоб дії слідували зверху вниз.

При побудові діаграми діяльності використовуються тільки нетригерні переходи, які переводять діяльність в подальший стан відразу, як тільки закінчиться дія в попередньому стані.

Якщо зі стану дії виходить єдиний перехід, то він може існувати як не позначений. Якщо ж таких переходів кілька, то спрацювати може тільки один з них, тоді для кожного з таких переходів має бути явно записана сторожова умова в прямих дужках.

Рисунок 11.26 Фрагмент діаграми діяльності для алгоритму знаходження коренів квадратного рівняння

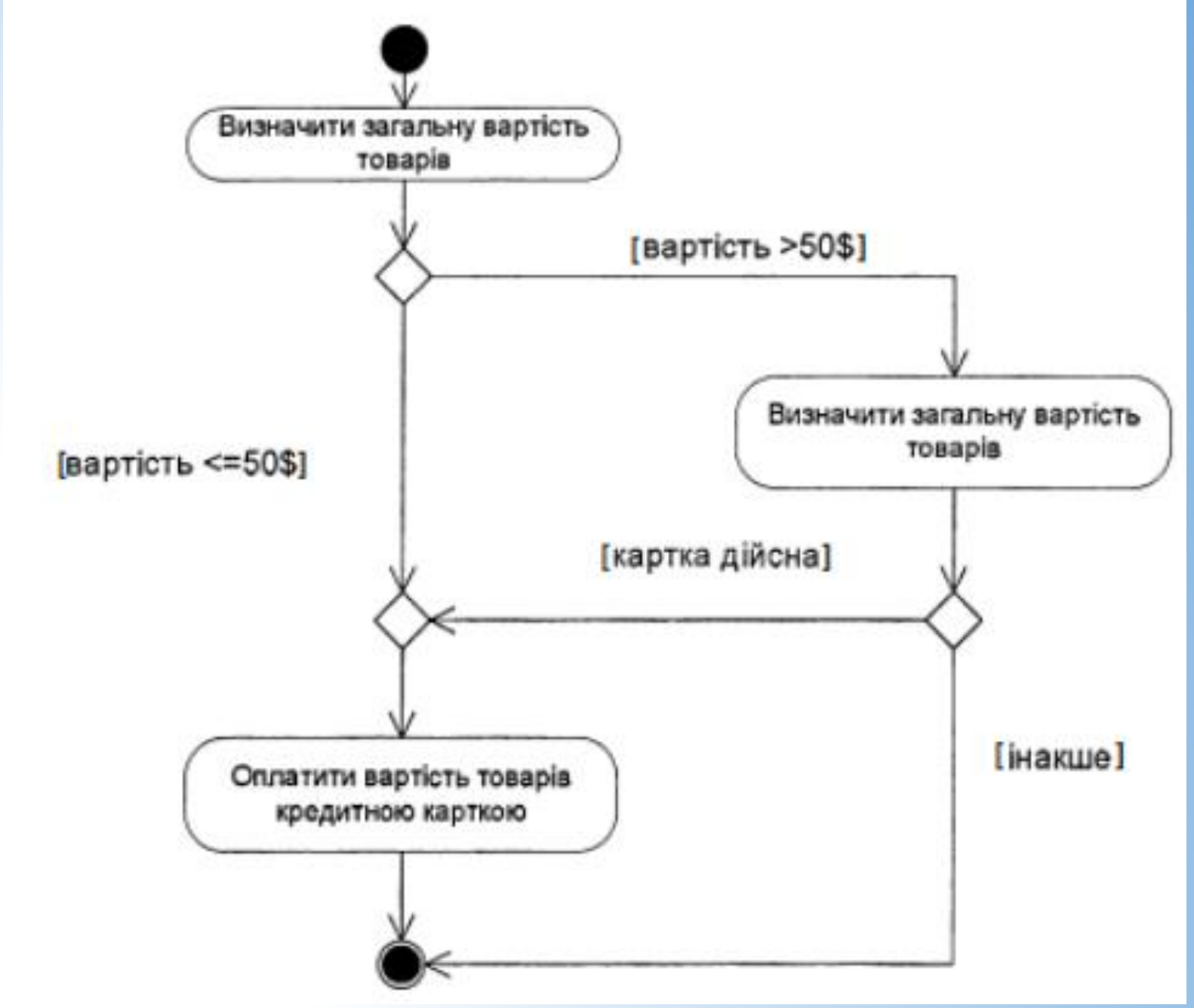


Рисунок 11.27 Різні варіанти розгалужень на діаграмі діяльності

Для розпаралелювання обчислень використовується спеціальний символ для розділення та злиття паралельних обчислень або потоків управління. Таким символом є пряма риска.

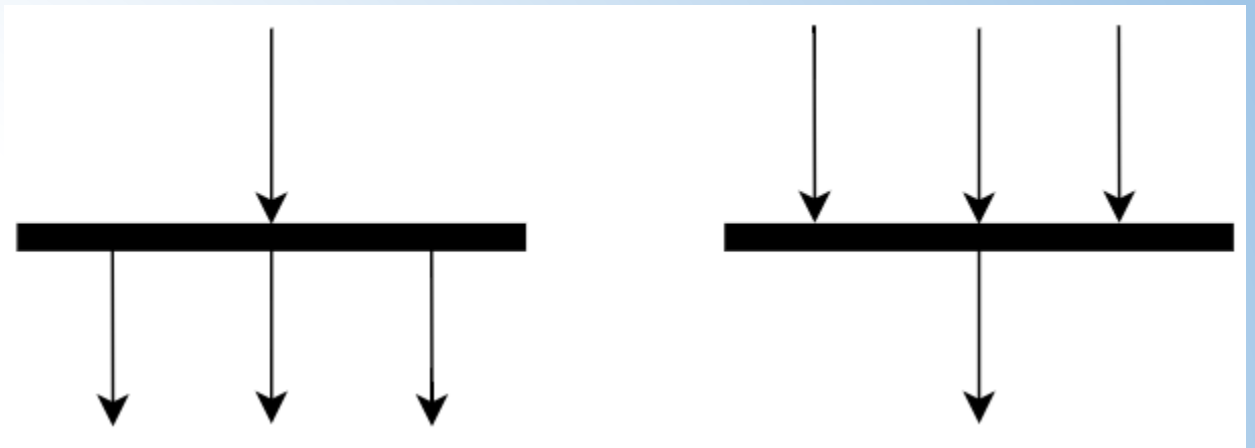
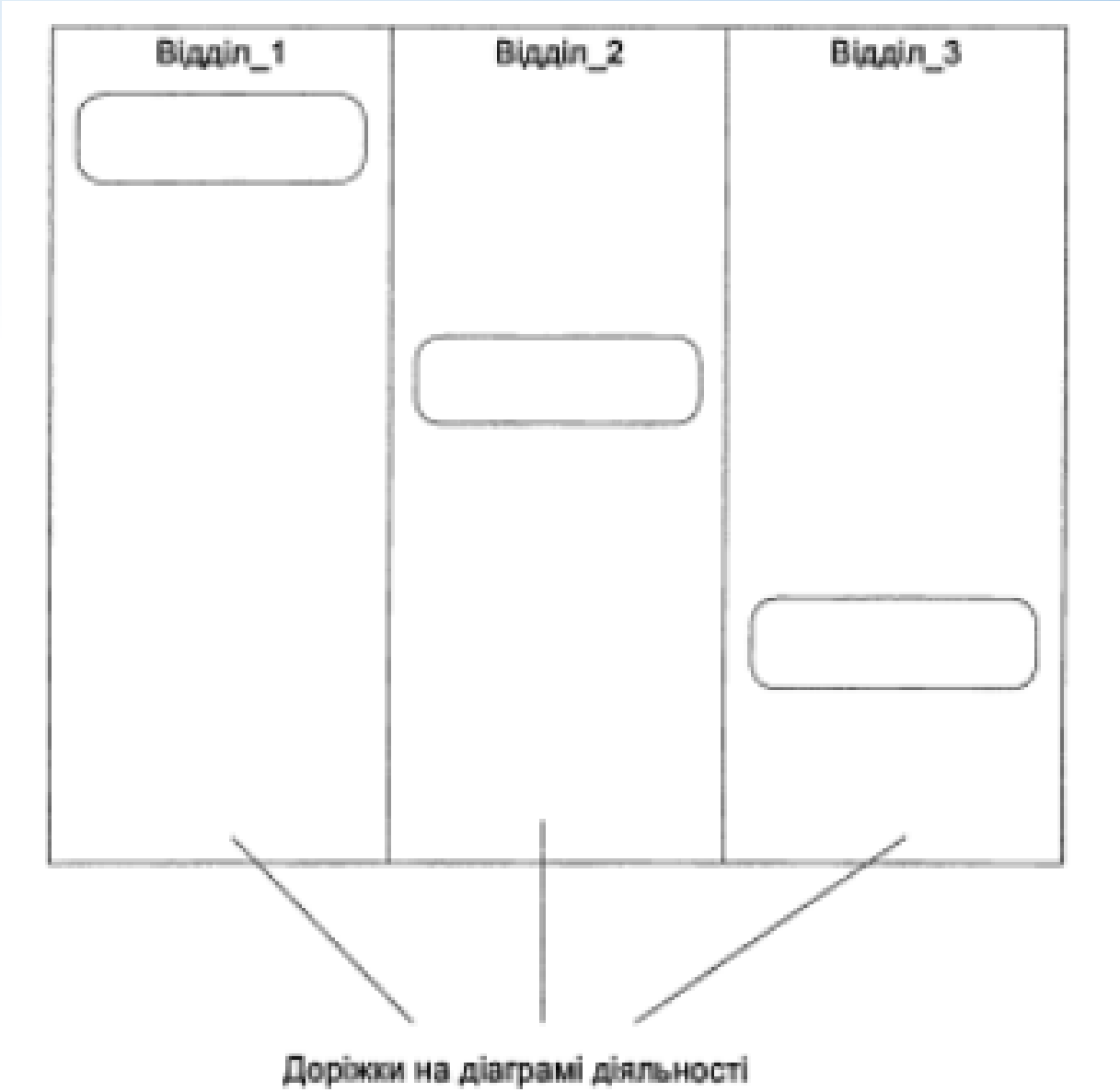


Рисунок 11.28 Графічне зображення розподілу і злиття паралельних потоків управління



Дві сусідні лінії утворюють доріжку, а група станів між цими лініями виконується окремим підрозділом (відділом, групою, відділенням, філією) компанії.

Рисунок 11.30 Варіант діаграми діяльності з доріжками

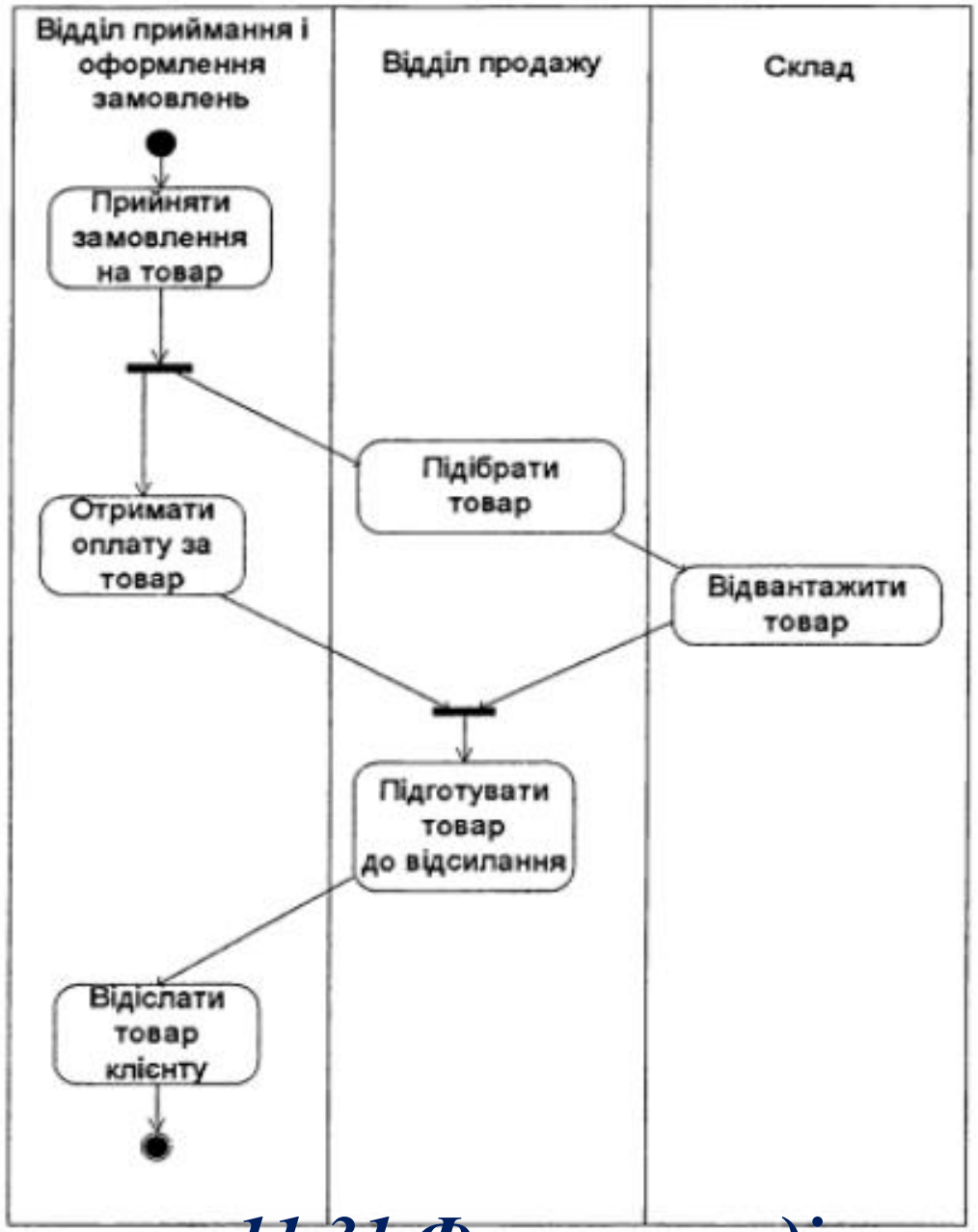


Рисунок 11.31 Фрагмент діаграми діяльності для торгової компанії

Назви підрозділів явно вказуються у верхній частині доріжки. Перетинати лінію доріжки можуть тільки переходи, які в цьому випадку позначають вихід або вхід потоку управління до відповідного підрозділу компанії. Порядок проходження доріжок не несе будь-якої семантичної інформації і визначається розумінням зручності.

Для графічного представлення об’єктів використовуються прямокутник класу, з тією відмінністю, що ім’я об’єкта підкреслюється.

Якщо об’єкт розташований на кордоні двох доріжок, то це може означати, що перехід до наступного стану дії в сусідній доріжці асоційований з готовністю деякого документа (об’єкт в деякому стані).

Якщо ж об’єкт цілком розташований усередині доріжки, то і стан цього об’єкта цілком визначається діями даної доріжки.

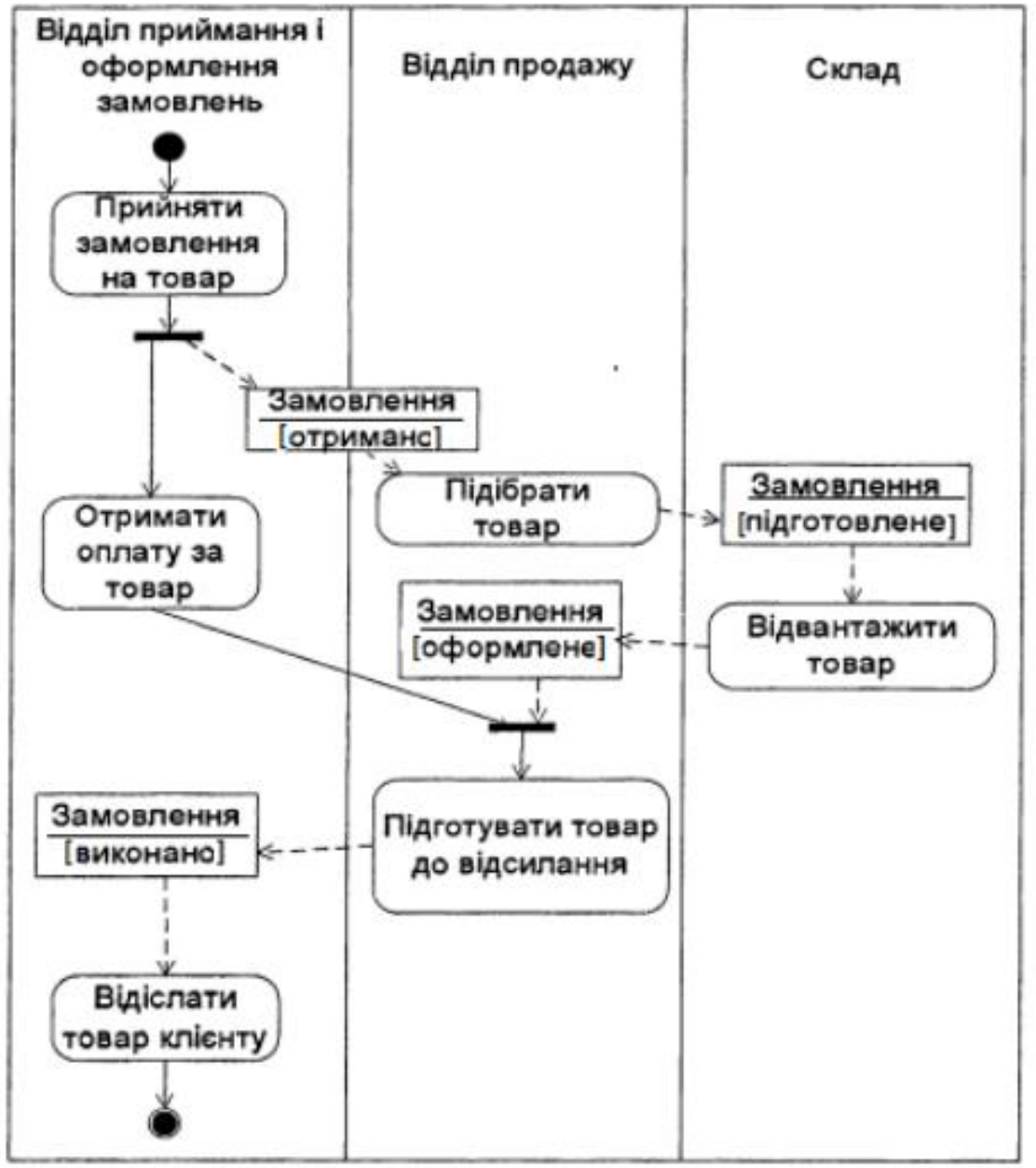


Рисунок 11.32 Фрагмент діаграми діяльності торгової компанії з об’єктом-замовлення

Рисунок 11.33 Діаграма діяльності з синхронізацією паралельних дій

Застосування доріжок і об'єктів відкриває додаткові можливості для наочного представлення бізнес-процесів, дозволяючи специфікувати діяльність підрозділів компаній і фірм.

Діаграма будується для окремого класу, варіанти використання, окремої операції класу або цілої підсистеми.

Побудову діаграми діяльності починають з виділення піддіяльностей, які в сукупності утворюють діяльність підсистем.

Діаграма діяльності не містить засобів вибору оптимальних рішень.

При побудові діаграм діяльності складних систем можуть бути успішно використані різні класи мереж Петрі і нейронних мереж.

ТЕМА №12. ПРОЄКТУВАННЯ АСПЕКТІВ ВЗАЄМОДІЇ СКЛАДОВИХ ОБ'ЄКТІВ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

12.1 Діаграма кооперації

12.2 Діаграма послідовності

12.1 ДІАГРАМА КООПЕРАЦІЇ

Діаграми кооперації – діаграми, на яких відображаються об’єкти та їх взаємозв’язки з наголосом на об’єкти, які беруть участь у обміні повідомленнями.

Діаграма кооперації ***відображає*** динамічну взаємодію об’єктів у рамках одного прецеденту використання.

Діаграма кооперації ***призначена*** для специфікації структурних аспектів взаємодії.

Головна особливість діаграми кооперації полягає в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об’єктами, які беруть участь в цій взаємодії.

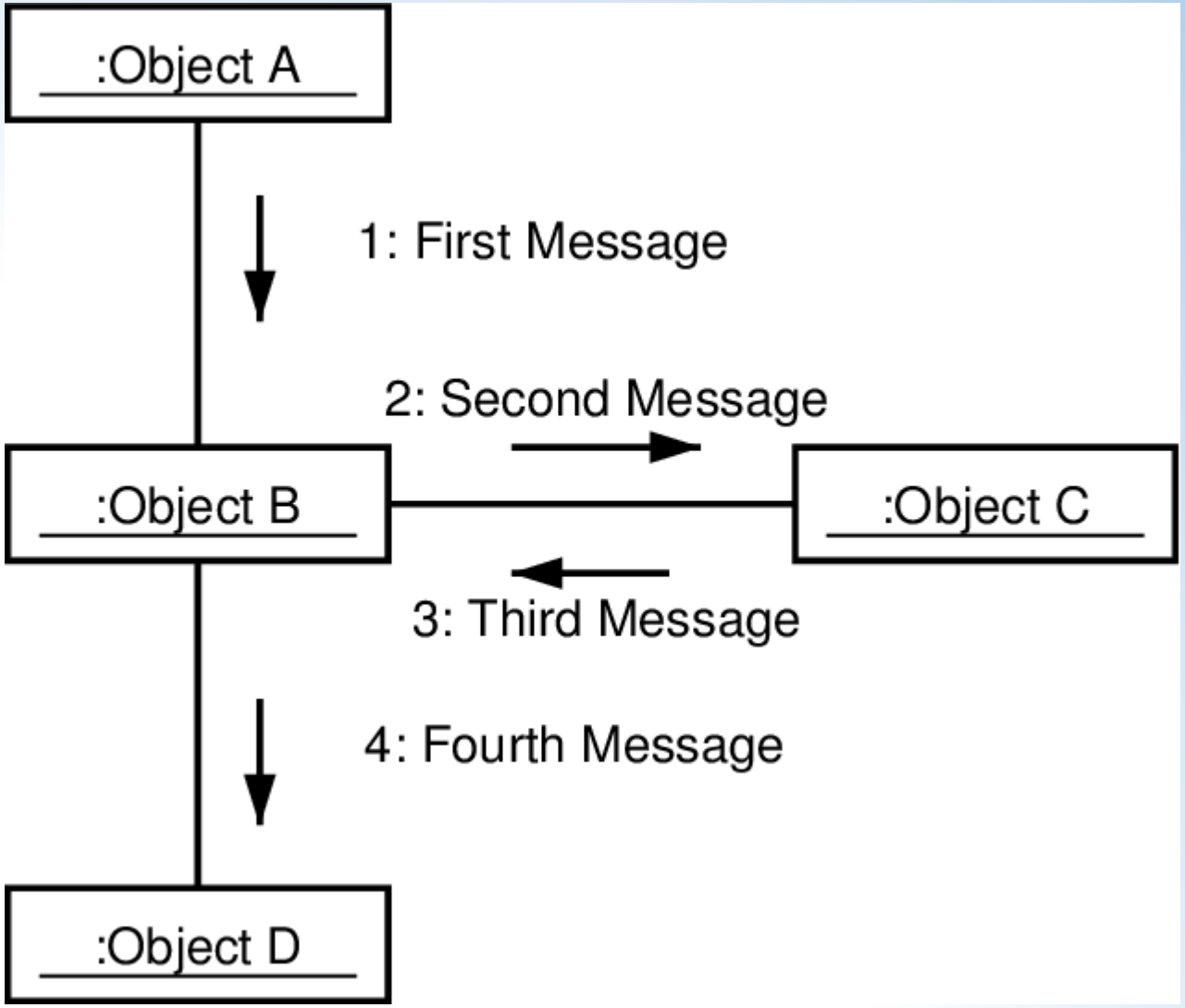


Рисунок 12.1

На діаграмі кооперації зображуються тільки відносини між об’єктами, що грають певні ролі у взаємодії.

Послідовність взаємодій і паралельних потоків може бути визначена за допомогою порядкових номерів.

Поведінка системи може описуватися на рівні окремих об’єктів, які обмінюються між собою повідомленнями, щоб досягти потрібної мети або реалізувати певний сервіс.

Поняття кооперації (Collaboration) служить для позначення безлічі взаємодіючих з певною метою об’єктів в загальному контексті модельованої системи.

Мета кооперації – специфікувати особливості реалізації окремих найбільш значущих операцій в системі.

Кооперація ***визначає*** структуру поведінки системи в термінах взаємодії учасників цієї кооперації.

Кооперація може бути представлена на ***двох рівнях***:

- на рівні специфікації
- на рівні прикладів

Діаграма кооперації на рівні специфікацій

Рисунок 12.2 Загальне представлення кооперації на діаграмах рівня специфікації

‘/’<Ім’я ролі класифікатора>‘:’<Ім’я класифікатора>[‘:’< Ім’я класифікатора >]*

/ R – роль з ім’ям R без вказівки класу.
: C – анонімна роль на базі класу C.
/ R : C – роль з ім’ям R на основі класу C.

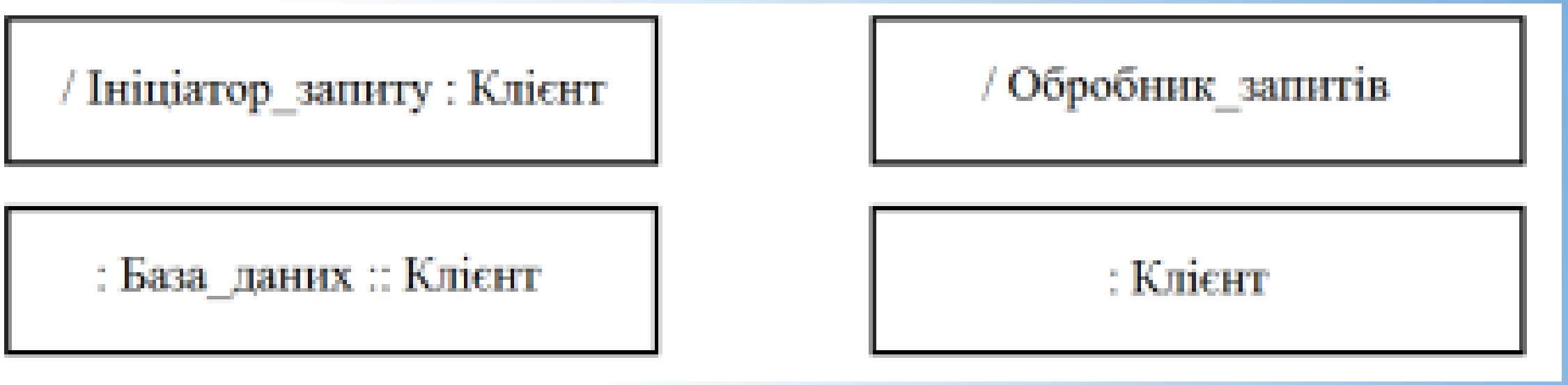


Рисунок 12.3 Приклади

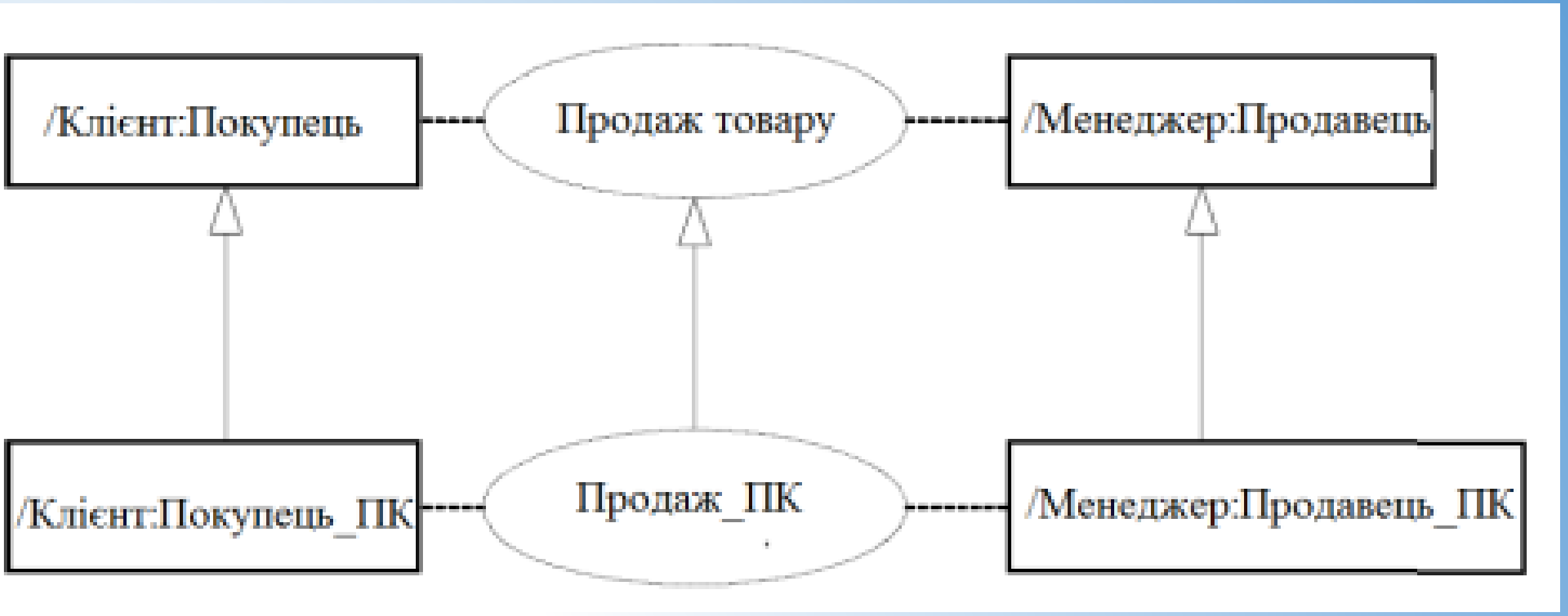


Рисунок 12.4 Графічне зображення відношення узагальнення між окремими кооперації рівня специфікації

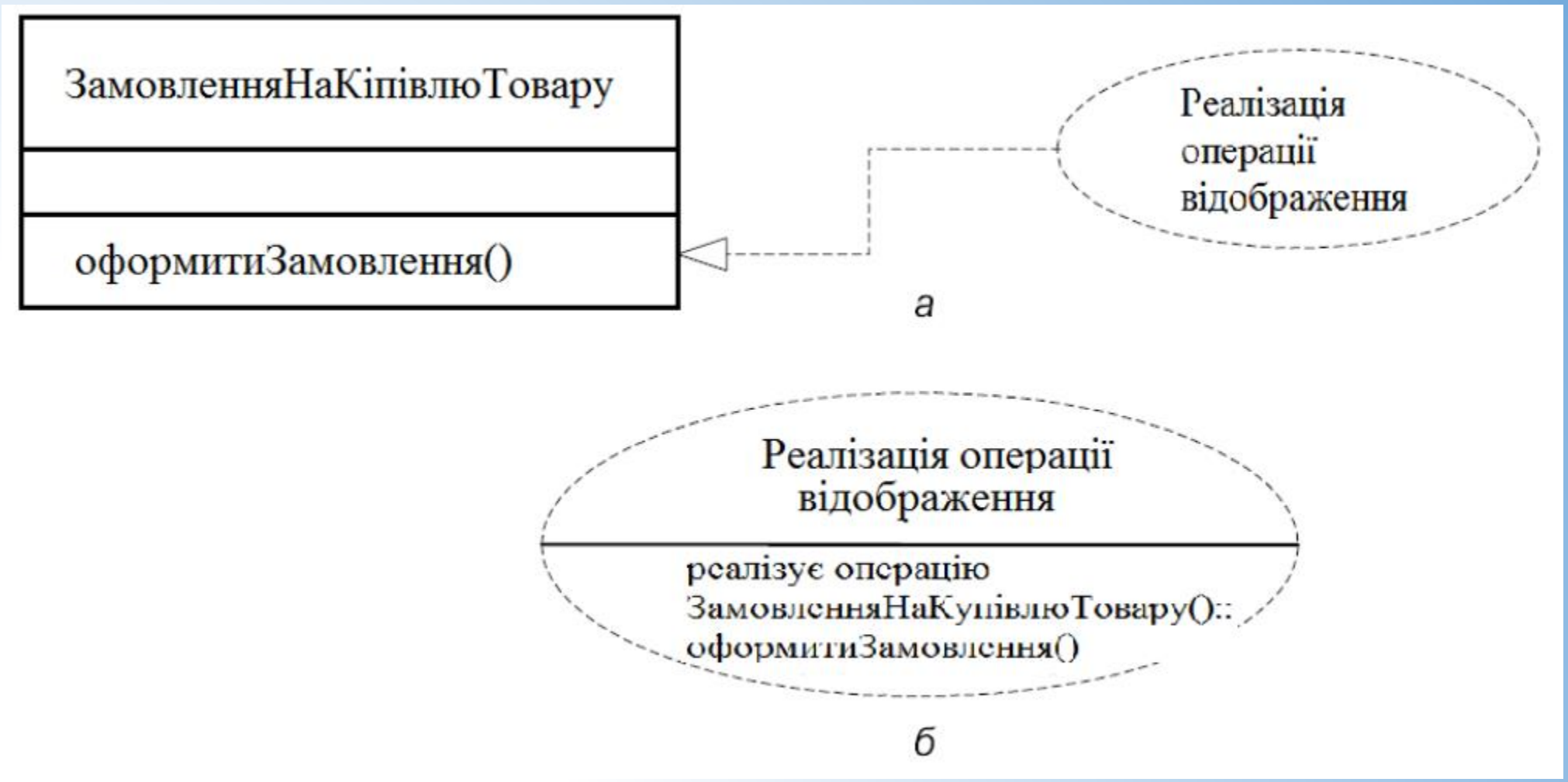


Рисунок 12.5 Способи подання кооперації, яка реалізує операцію класу

Діаграма кооперації на рівні прикладів

Об'єкт (object) є окремим екземпляром класу, який створюється на етапі виконання програми.

<Ім'я об'єкту>'/'<Ім'я ролі класифікатора>':'< Ім'я класифікатора>['':'< Ім'я класифікатора >]*

: C – анонімний об'єкт, утворений на основі класу C.
/ R – анонімний об'єкт, який грає роль R.
/ R : C – анонімний об'єкт, утворений на основі класу C і грає роль R.
o / R – об'єкт з ім'ям o, який грає роль R.
o : C – об'єкт з ім'ям o, утворений на основі класу C.
o / R : C – об'єкт з ім'ям o, утворений на основі класу C і грає роль R.
o – «об'єкт-сирота» з ім'ям o.
o : – «об'єкт-сирота» з ім'ям o.

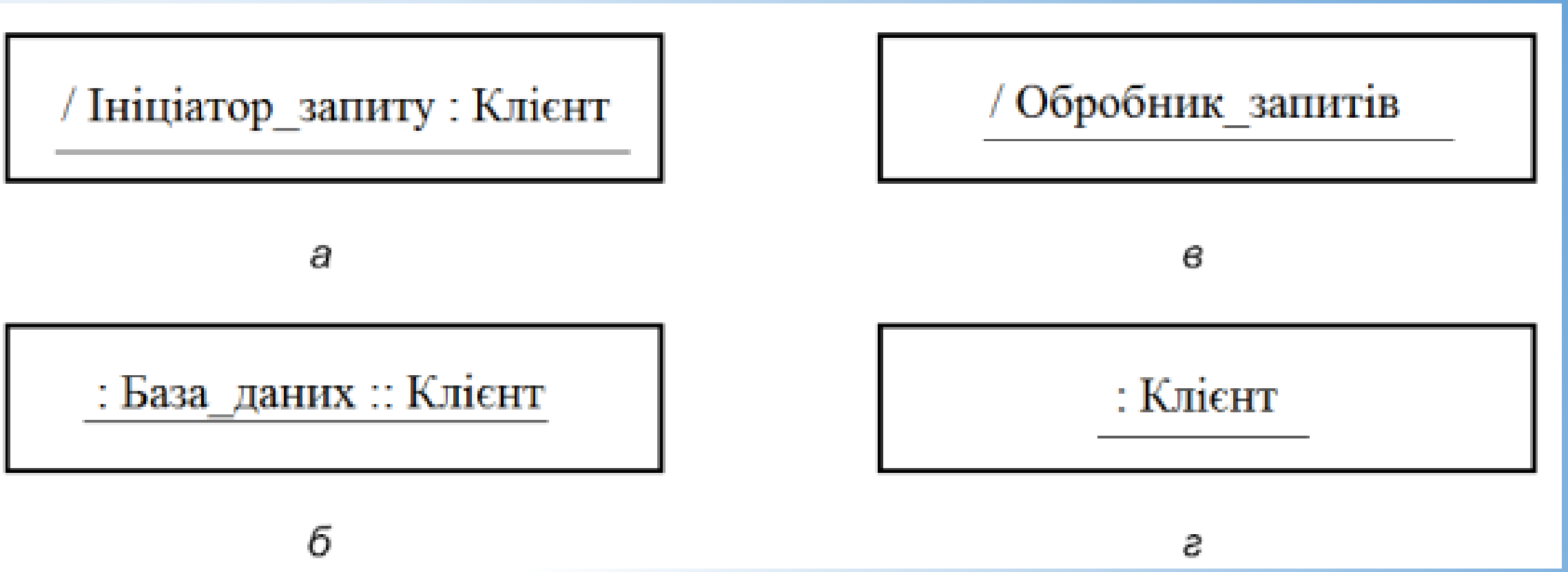


Рисунок 12.6 Приклади різних варіантів запису імен об'єктів, ролей і класів на діаграмах кооперації

Мультиоб’єкт – безліч об’єктів на одному з кінців асоціації.

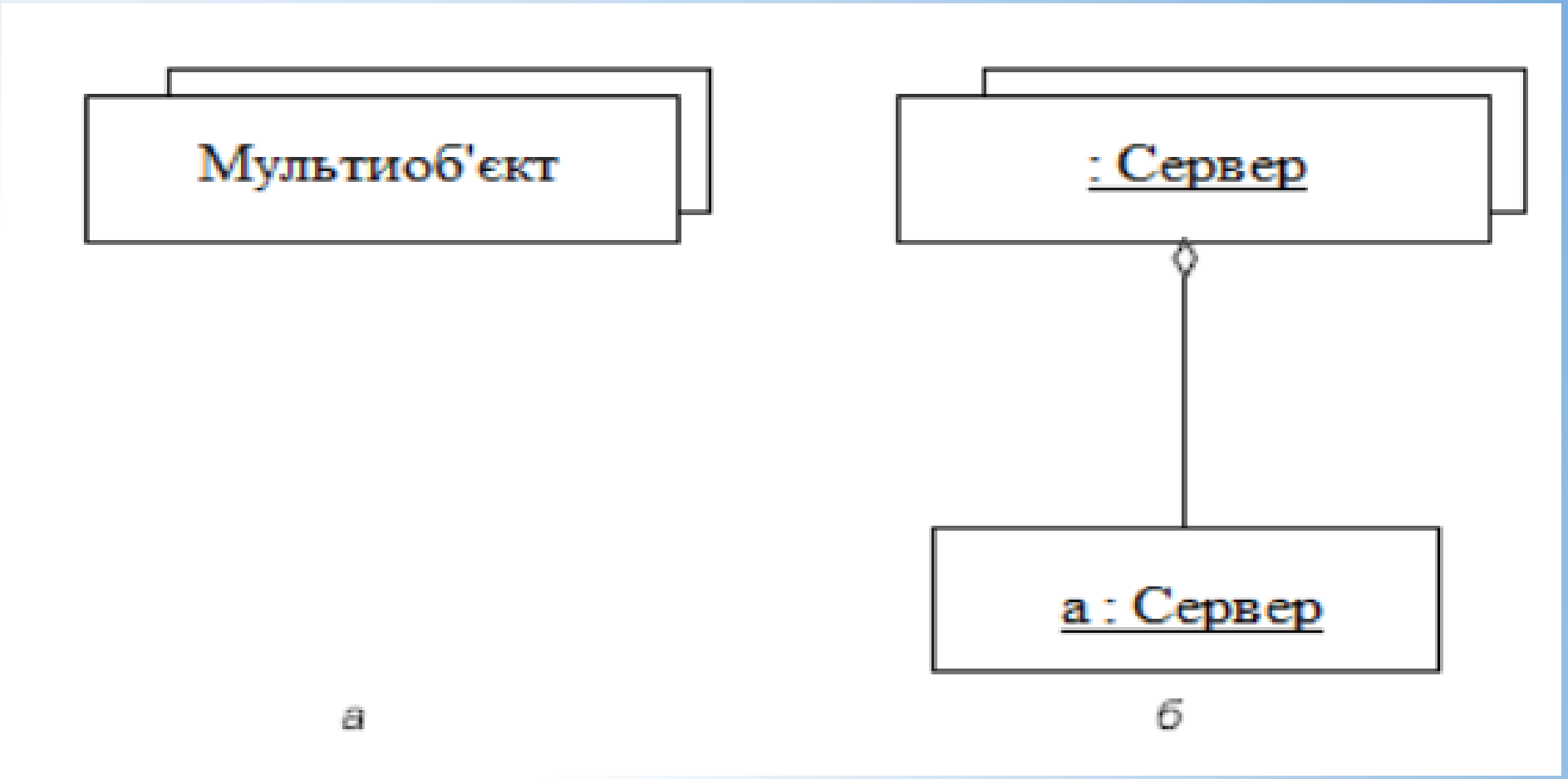


Рисунок 12.7 Графічне зображення мультиоб’єтів на діаграмі кооперації

Пасивний об'єкт оперує тільки даними і не може ініціювати діяльність з управління іншими об'єктами.

Активний об'єкт (active object) має свою власну нитку (thread) управління і може ініціювати діяльність з управління іншими об'єктами.

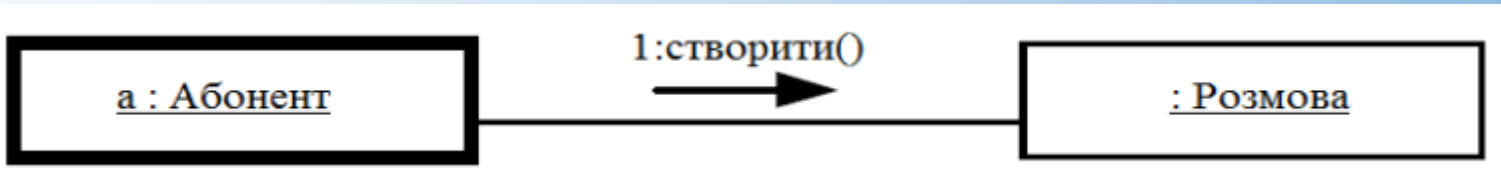


Рисунок 12.8 Графічне зображення активного об'єкта (ліворуч) на діаграмі кооперації

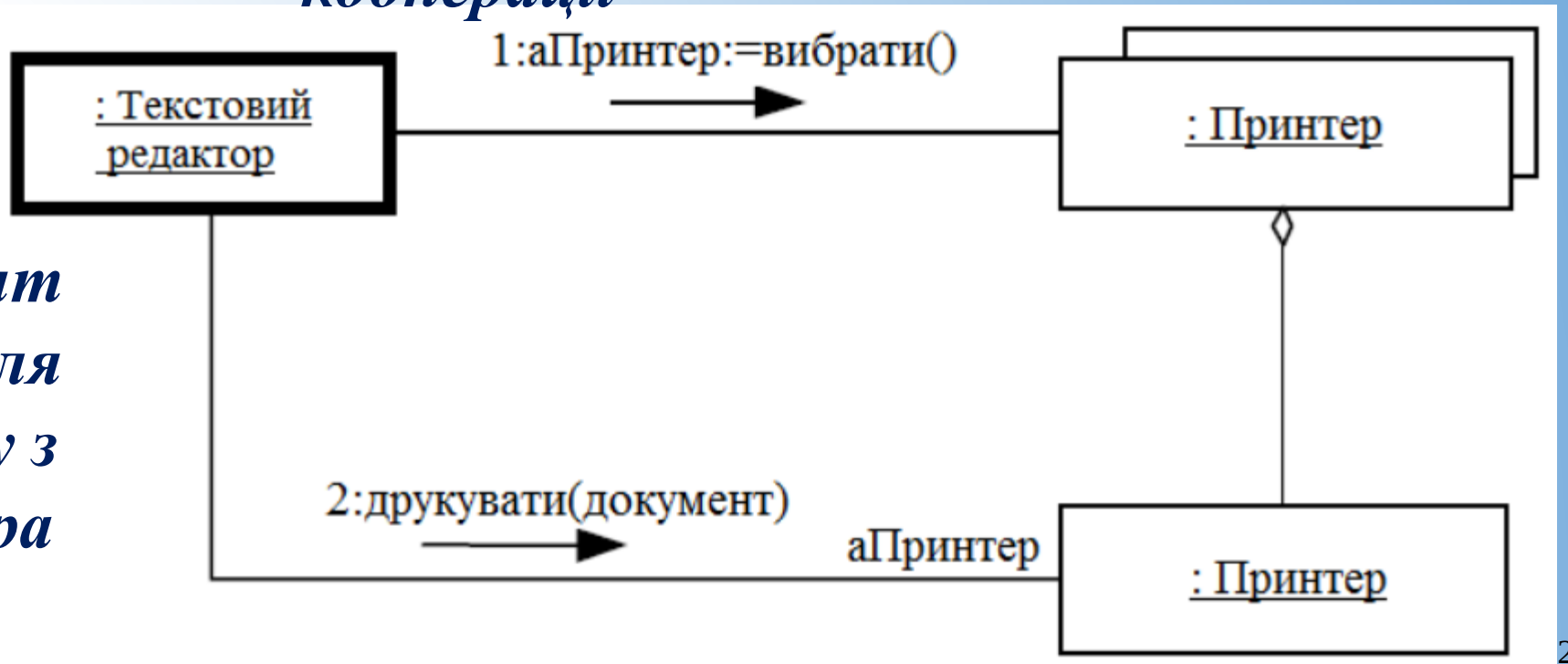


Рисунок 12.9 Фрагмент діаграми кооперації для виклику функції друку з текстового редактора

Складений об'єкт (composite object) або об'єкт-контейнер призначений для представлення об'єкта, що має власну структуру та внутрішні потоки управління.

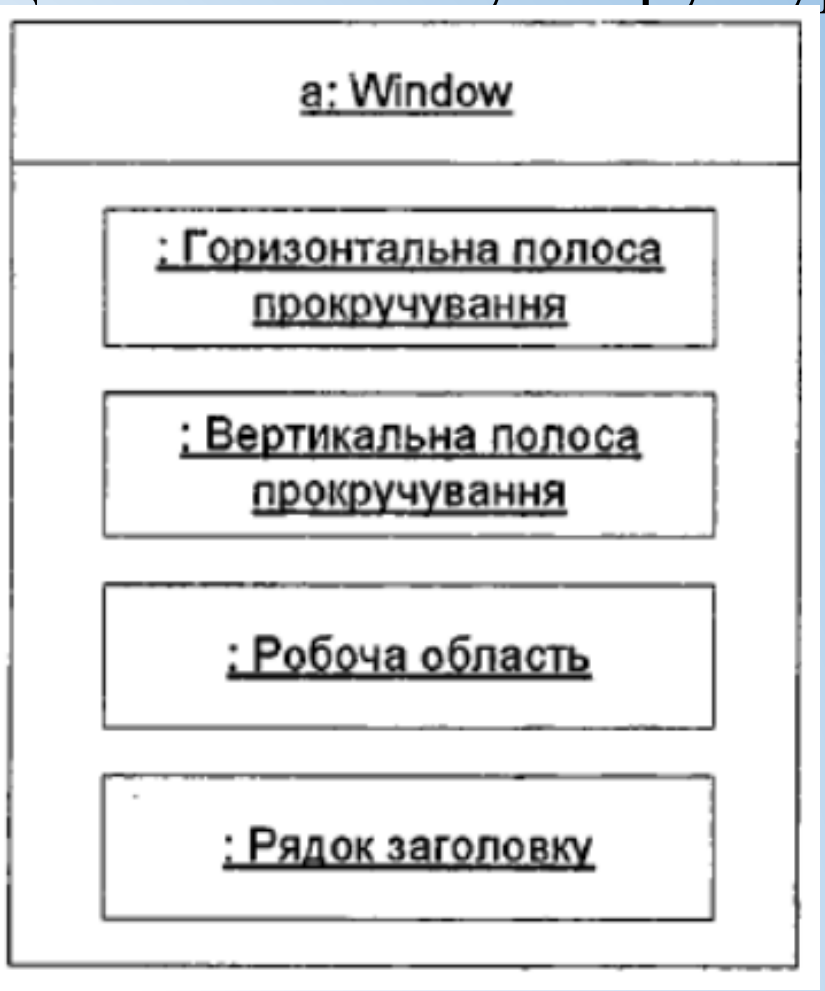


Рисунок 12.10 Графічне зображення складеного об'єкта на діаграмі кооперації

Зв'язок (link) – екземпляр або приклад довільної асоціації.

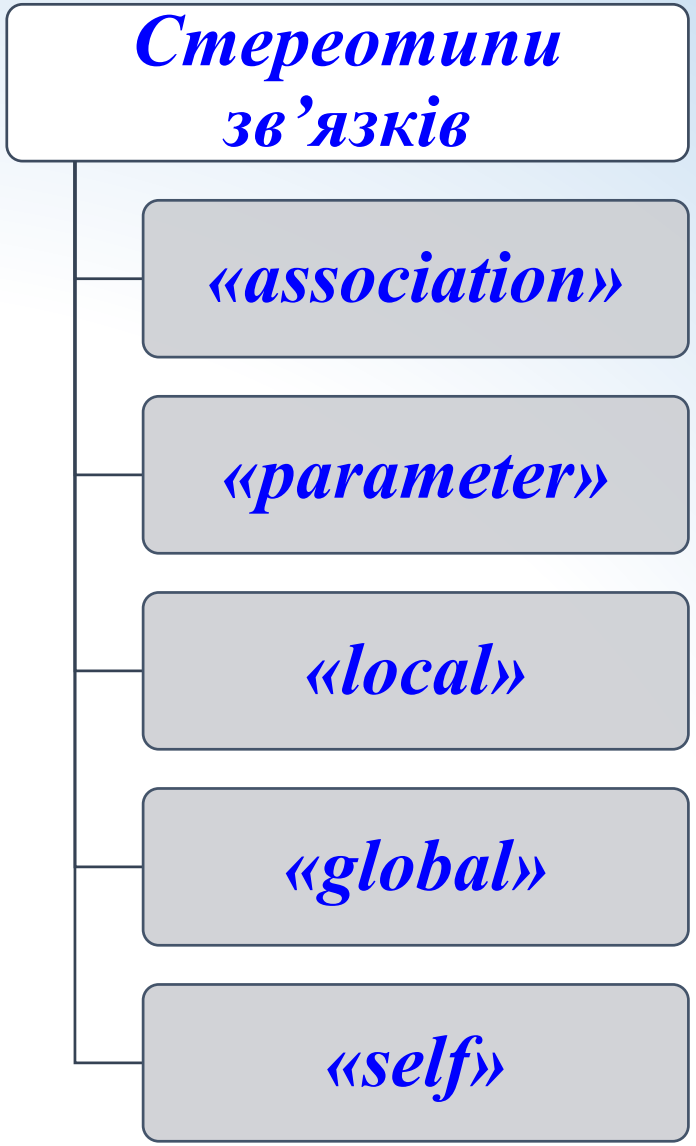


Рисунок 12.11

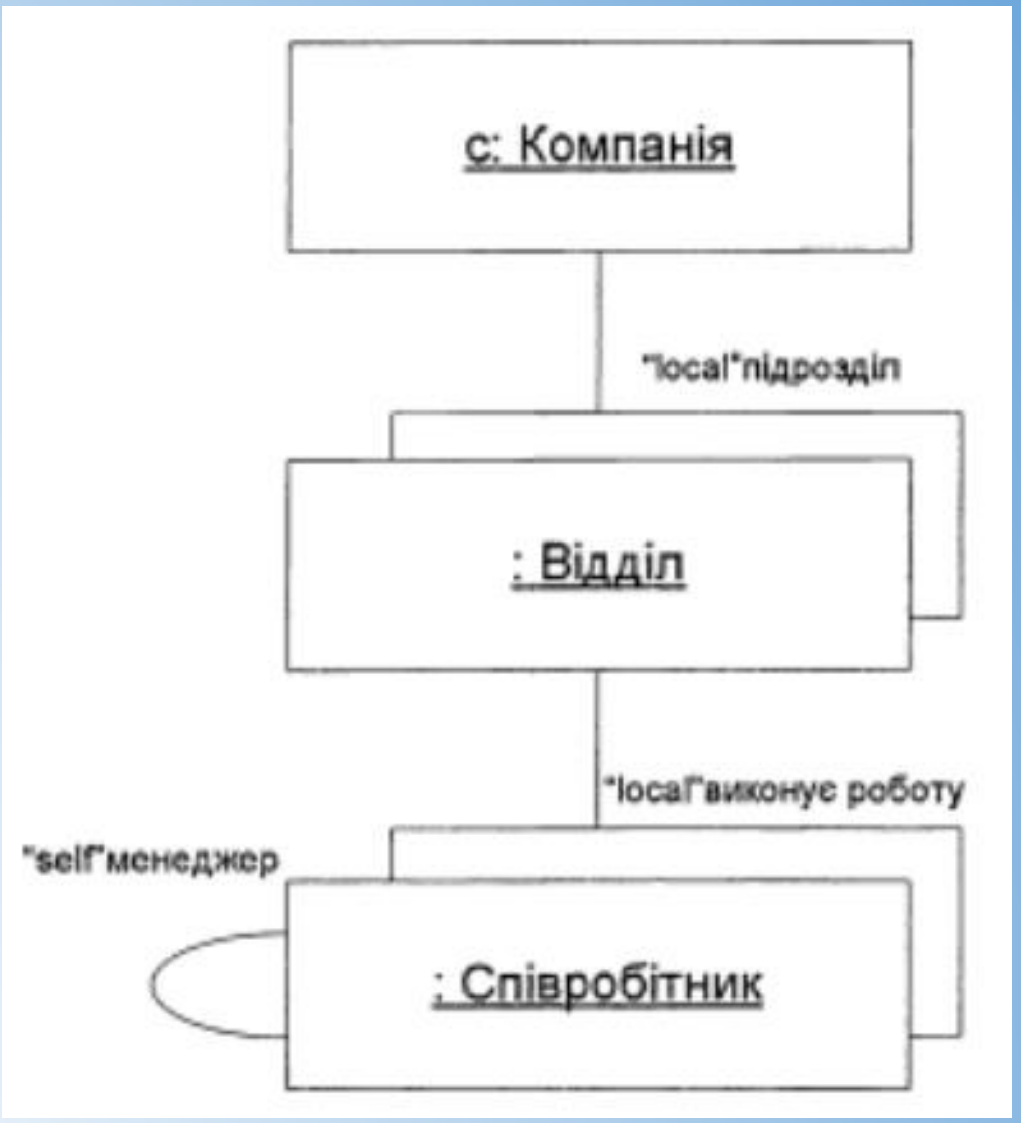


Рисунок 12.12 Графічне зображення зв'язків з різними стереотипами

Повідомлення на діаграмі кооперації специфікує комунікацію між двома об’єктами, один з яких передає іншому певну інформацію.

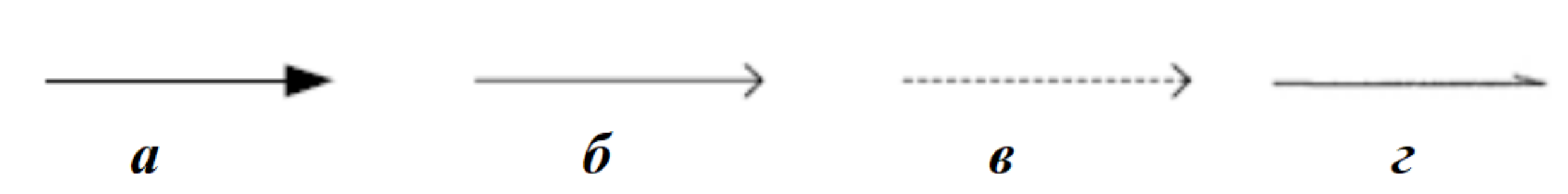


Рисунок 12.13 Графічне зображення різних типів повідомлень на діаграмі кооперації

<Попередні повідомлення><[Сторожова умова]><Вираз послідовності>

<Повертаєме значення:=ім’я повідомлення><(Список аргументів)>

0<Номер повідомлення‘,’><Номер повідомлення, ‘>‘ / ‘

Сторожова умова – булевий вираз, який призначений для синхронізації окремих ниток потоку управління.

[(x> = 0) & (x <= 255)] 1.2: відобразити_на_екрані_колір(x)

[кількість цифр номера = 7] 3.1: набрати_телефонний_номер()

Вираз послідовності – це розділений точками список окремих термів послідовностей, після якого записується двокрапка:

0<Термін послідовності‘.’>< Термін послідовності ‘.’>‘:’

[Ціле число | Ім’я] [Символ рекурентності]

Два випадки запису рекурентності:

- ‘*’ ‘[‘Пропозиція-ітерація‘]’ для запису ітеративного виконання відповідного виразу.
- ‘[‘ Пропозиція-умова ‘]’ для запису розгалуження.

Рисунок 12.14 Початковий фрагмент діаграми кооперації для прикладу моделювання звичайної телефонної розмови

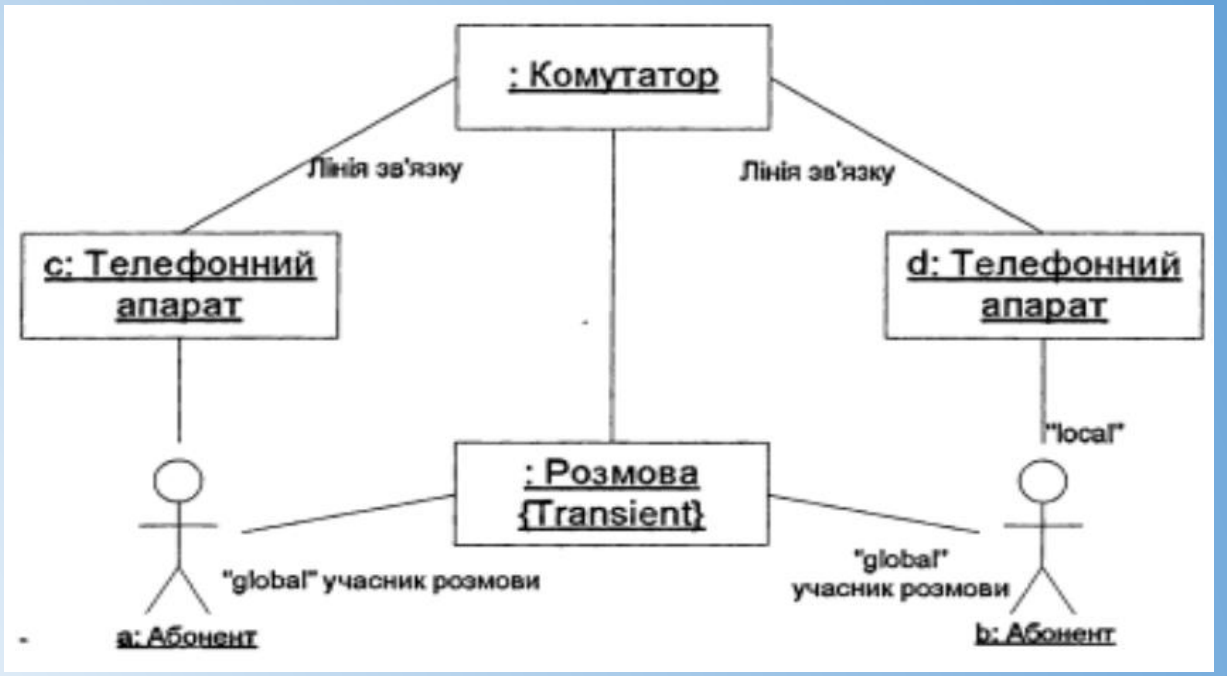


Рисунок 12.15 Фрагмент діаграми кооперації, доповнений стереотипами ролей зв'язків, іменами асоціацій і позначеним значенням об'єкта

Рисунок 12.16 Остаточний варіант діаграми кооперації для моделювання телефонної розмови

Побудову діаграми кооперації можна починати відразу після побудови діаграми варіантів використання, де кожен з варіантів використання може бути специфікований у вигляді окремої діаграми кооперації рівня специфікації.

Після побудови діаграми класів, кожна з діаграм кооперації може уточнюватися у вигляді відповідної діаграми рівня прикладів.

При побудові діаграм кооперації рівня специфікації бажано застосовувати найбільш зрозумілу замовнику термінологію, уникаючи технічних фраз і словосполучень.

Процес побудови діаграми кооперації рівня прикладів повинен бути узгоджений з процесами побудови діаграми класів і діаграми послідовності.

12.2 ДІАГРАМА ПОСЛІДОВНОСТІ

Діаграма послідовності відображає взаємодії об’єктів, впорядкованих за часом. На діаграмі послідовності зображуються виключно ті об’єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об’єктами.

Один вимір – зліва направо, у вигляді вертикальних ліній, кожна з яких зображає лінію життя окремого об’єкта, який бере участь у взаємодії.

Другий вимір діаграми послідовності – вертикальна тимчасова вісь, спрямована зверху вниз.

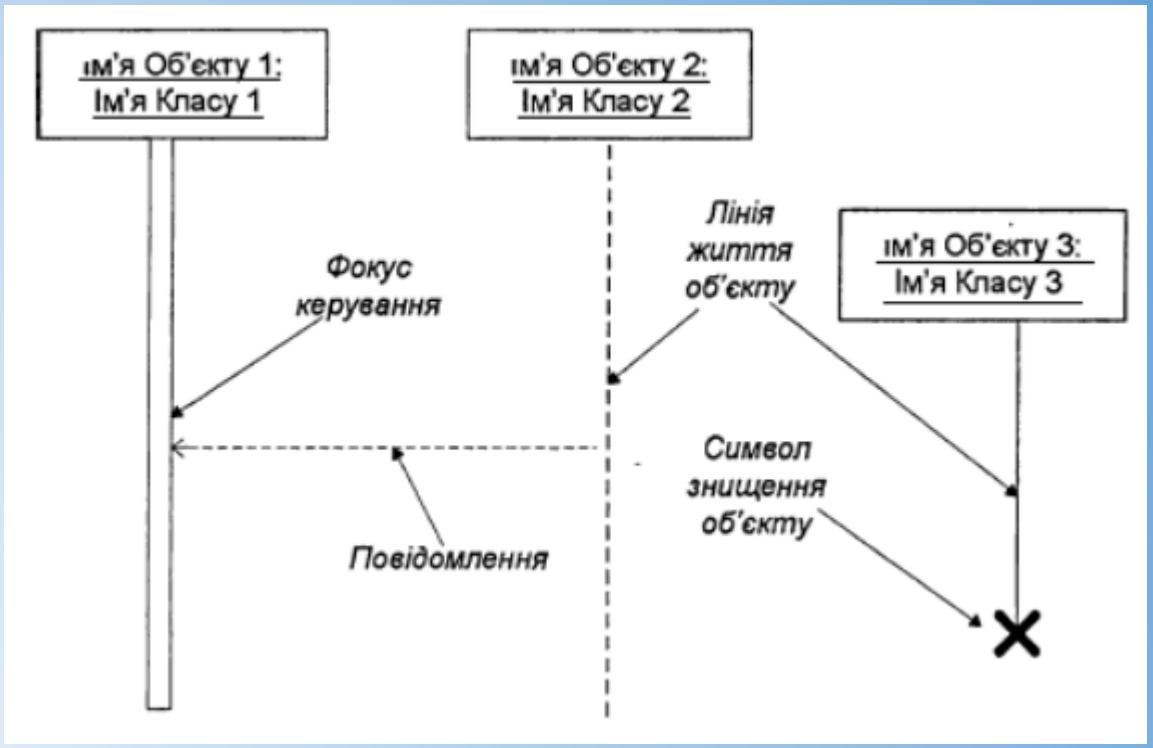


Рисунок 12.17 Різні графічні примітиви діаграми послідовності

Лінія життя об'єкта (object lifeline) служить для позначення періоду часу, протягом якого об'єкт існує в системі.

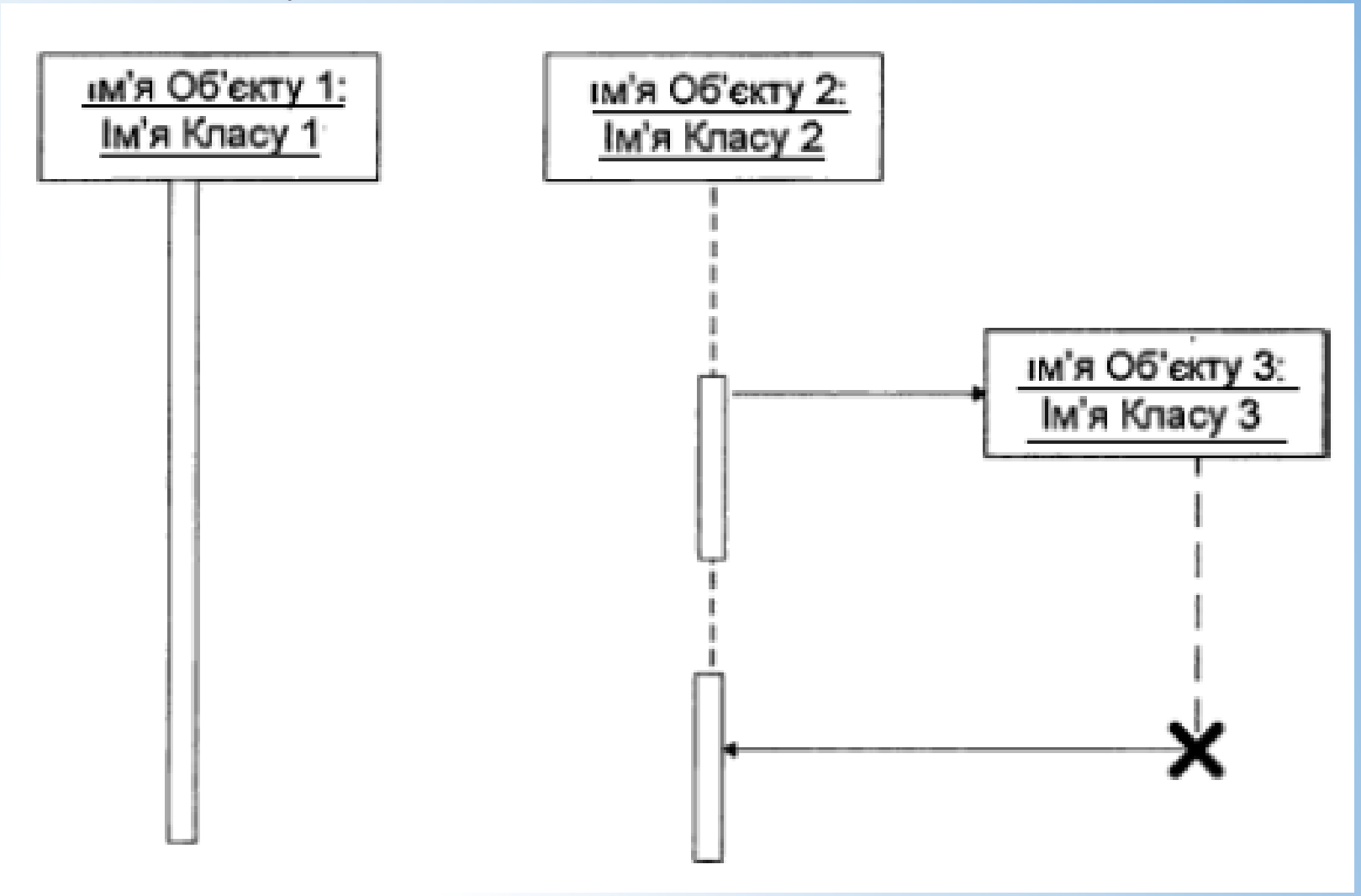


Рисунок 12.18 Графічне зображення різних варіантів ліній життя і фокусів управління об'єктів

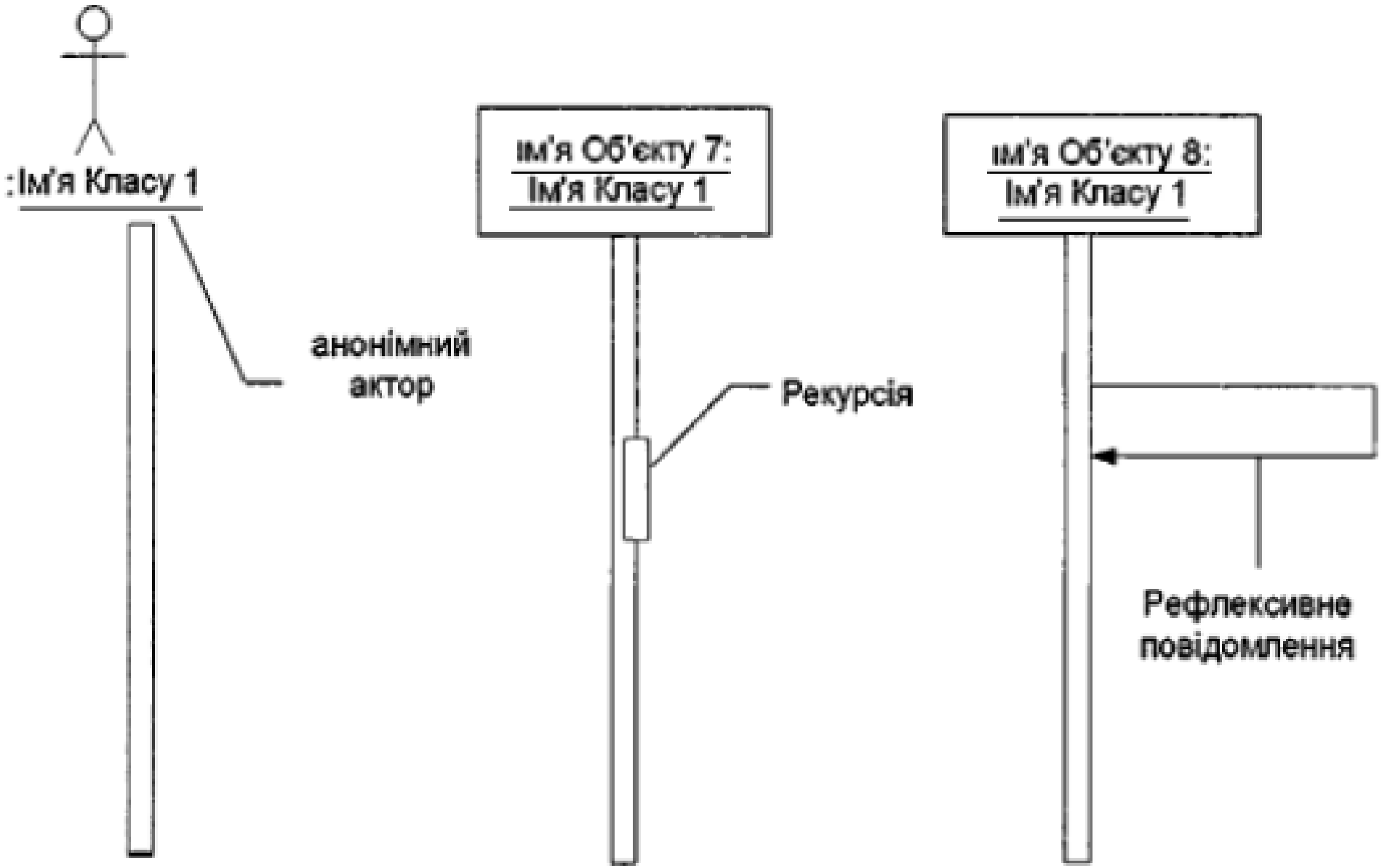


Рисунок 12.19 Графічне зображення актора, рекурсії та рефлексивного повідомлення на діаграмі послідовності

Повідомлення (message) представляє собою закінчений фрагмент інформації, який відправляється одним об’єктом іншому, при цьому прийом повідомлення ініціює виконання певних дій, спрямованих на вирішення окремого завдання тим об’єктом, якому це повідомлення надіслано.

Кожне повідомлення має напрямок від об’єкта, який ініціює і відправляє повідомлення, до об’єкта, який його отримує.

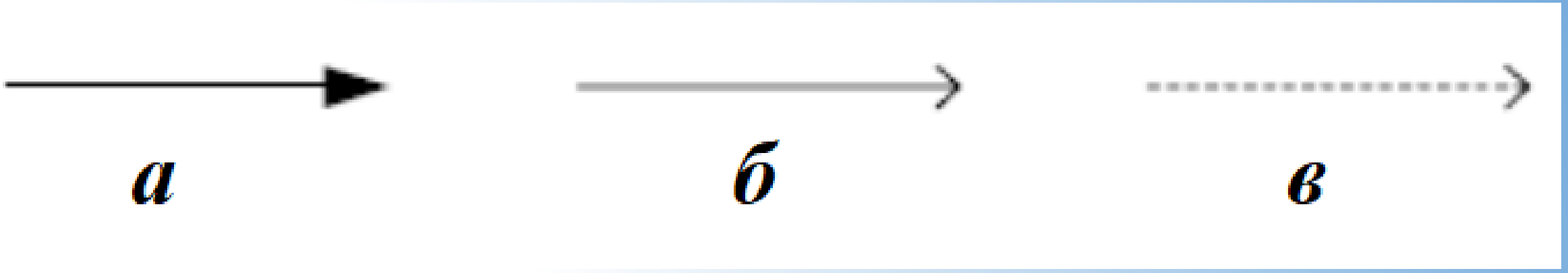


Рисунок 12.20 Графічне зображення різних видів повідомлень між об’єктами на діаграмі послідовності

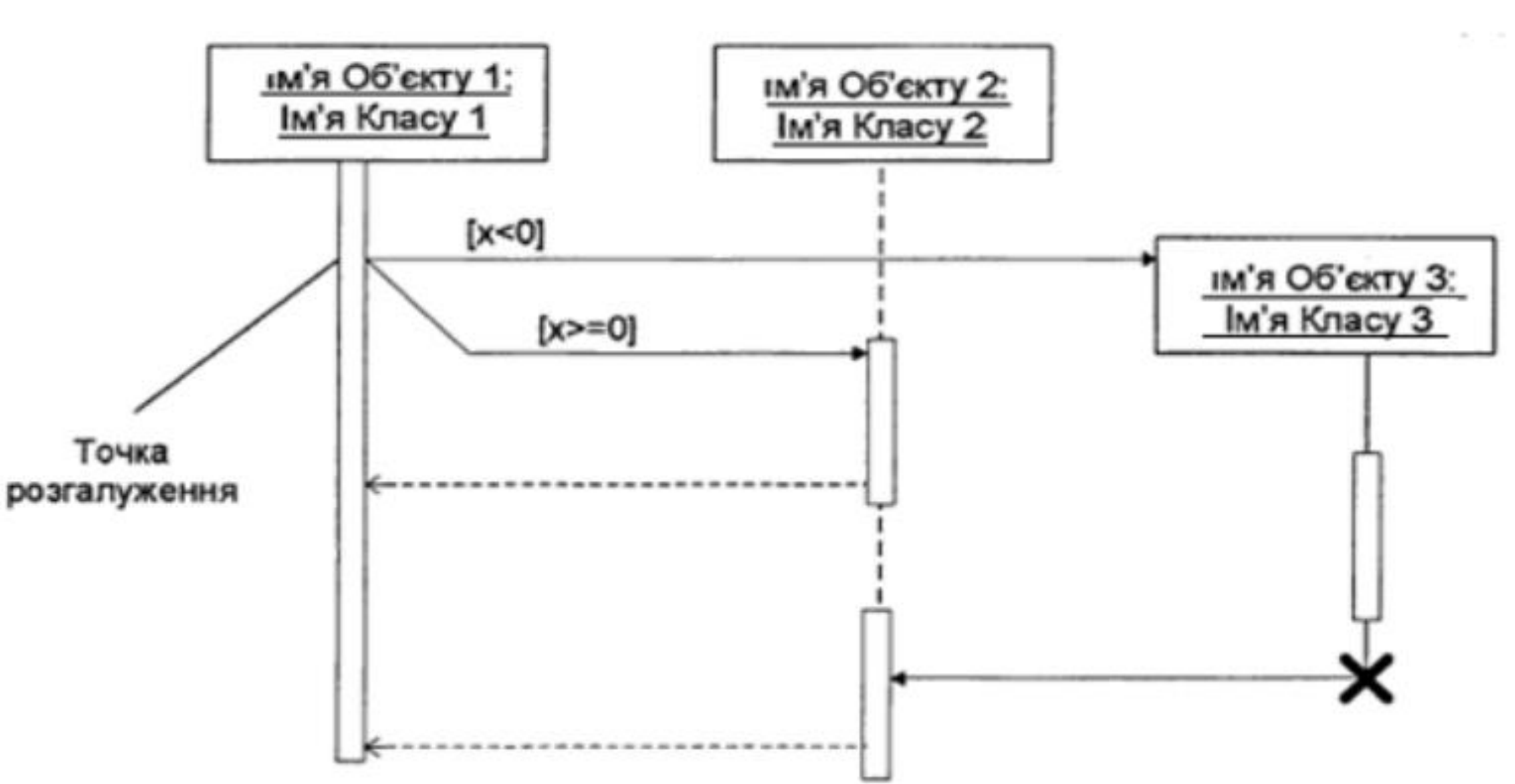


Рисунок 12.21 Графічне зображення бінарного розгалуження потоку керування на діаграмі послідовності

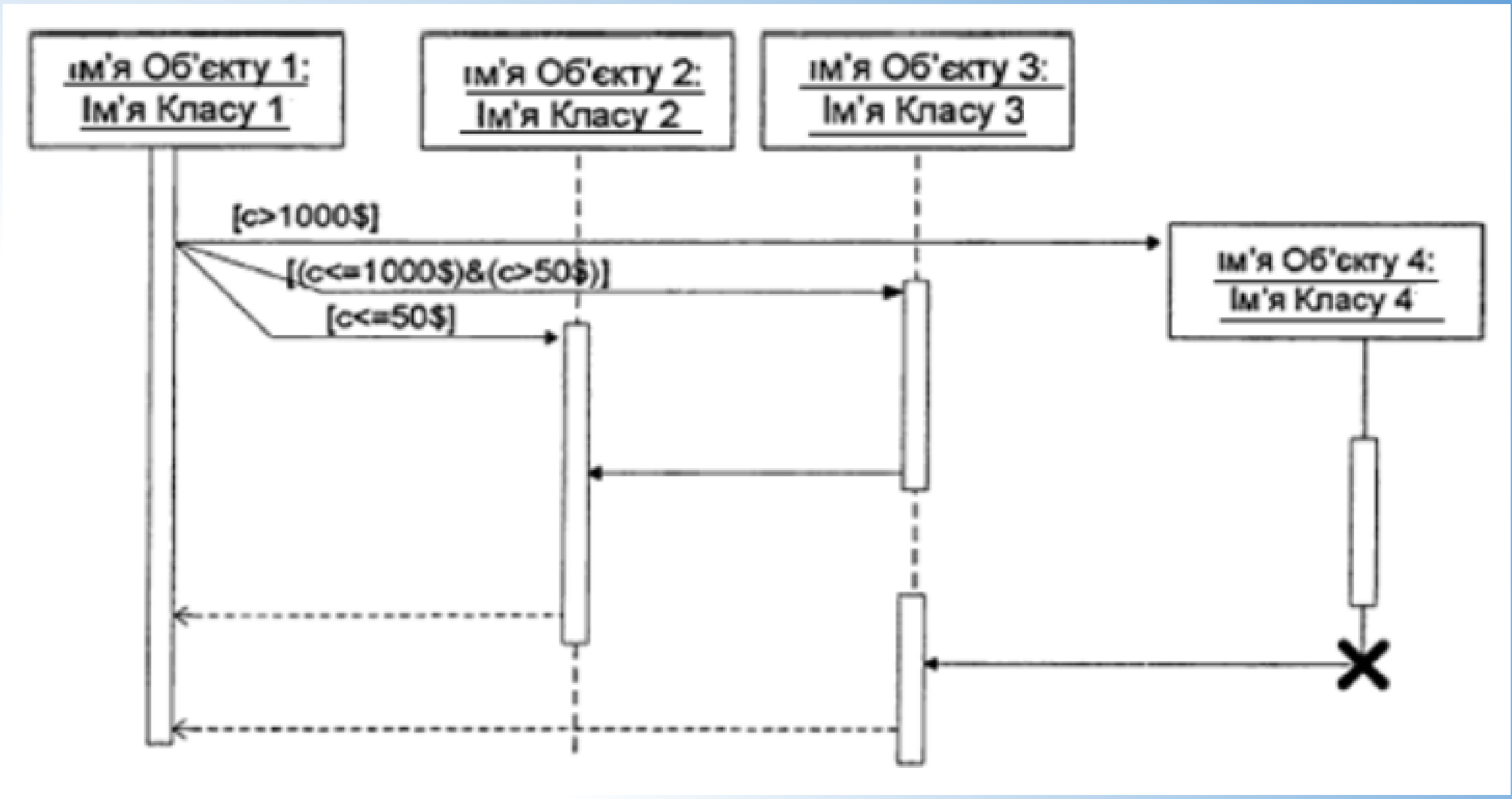


Рисунок 12.22 Графічне зображення тернарного розгалуження потоку керування на діаграмі послідовності

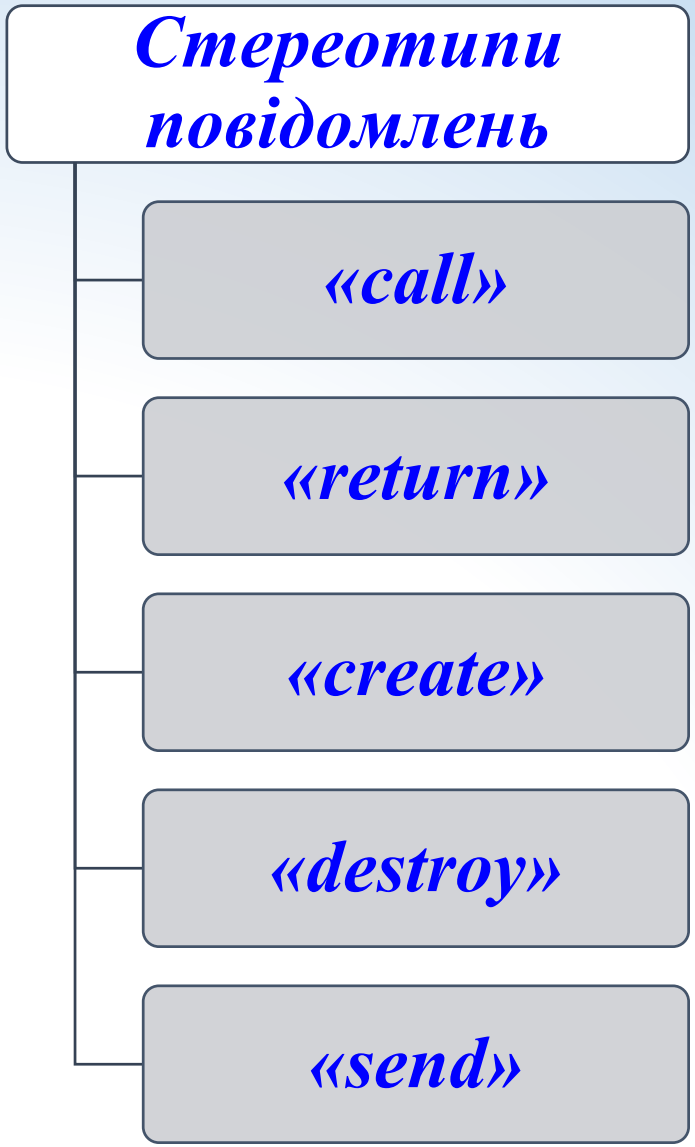


Рисунок 12.23

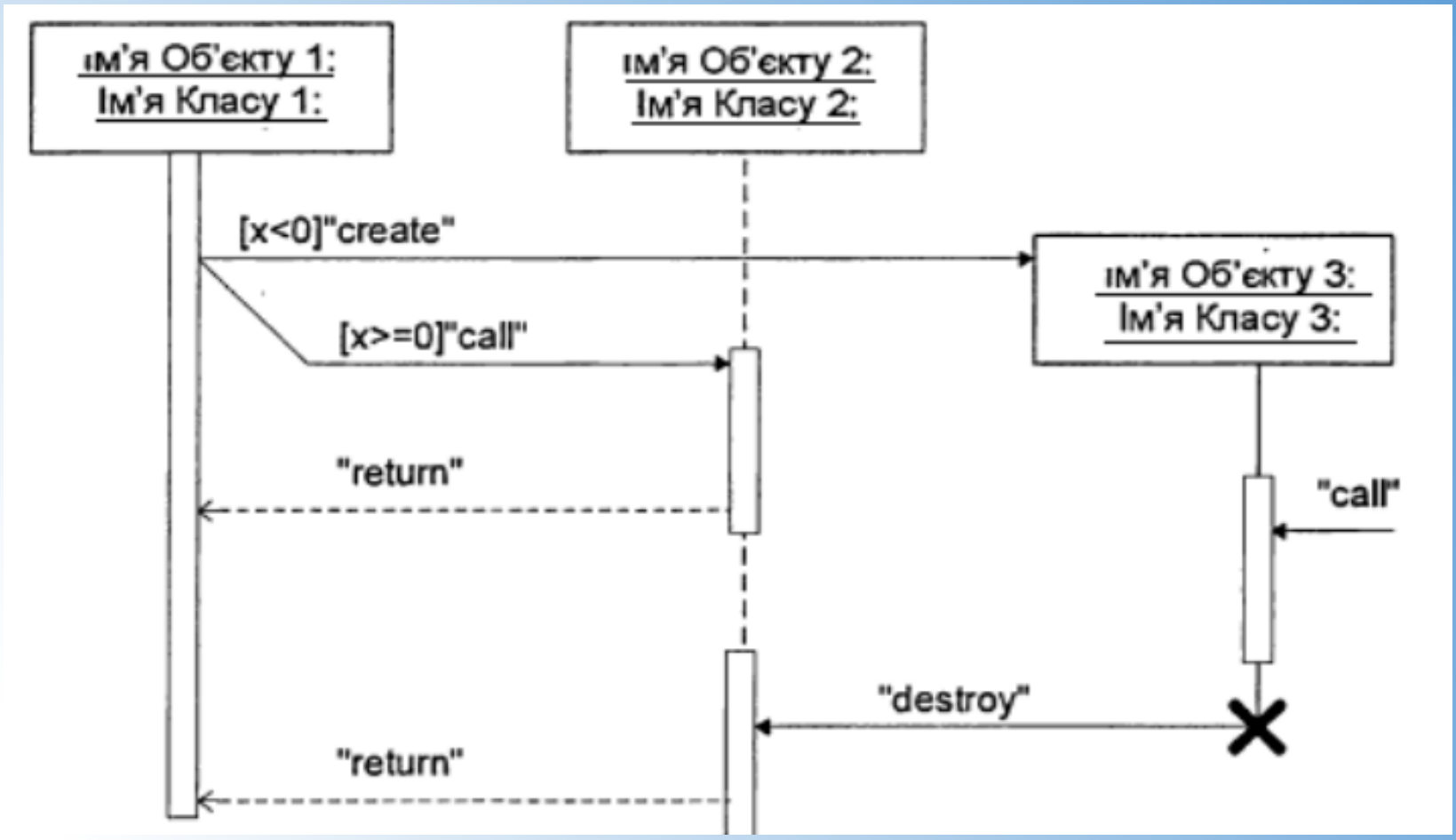


Рисунок 12.24 Діаграма послідовності зі стереотипними значеннями повідомлень

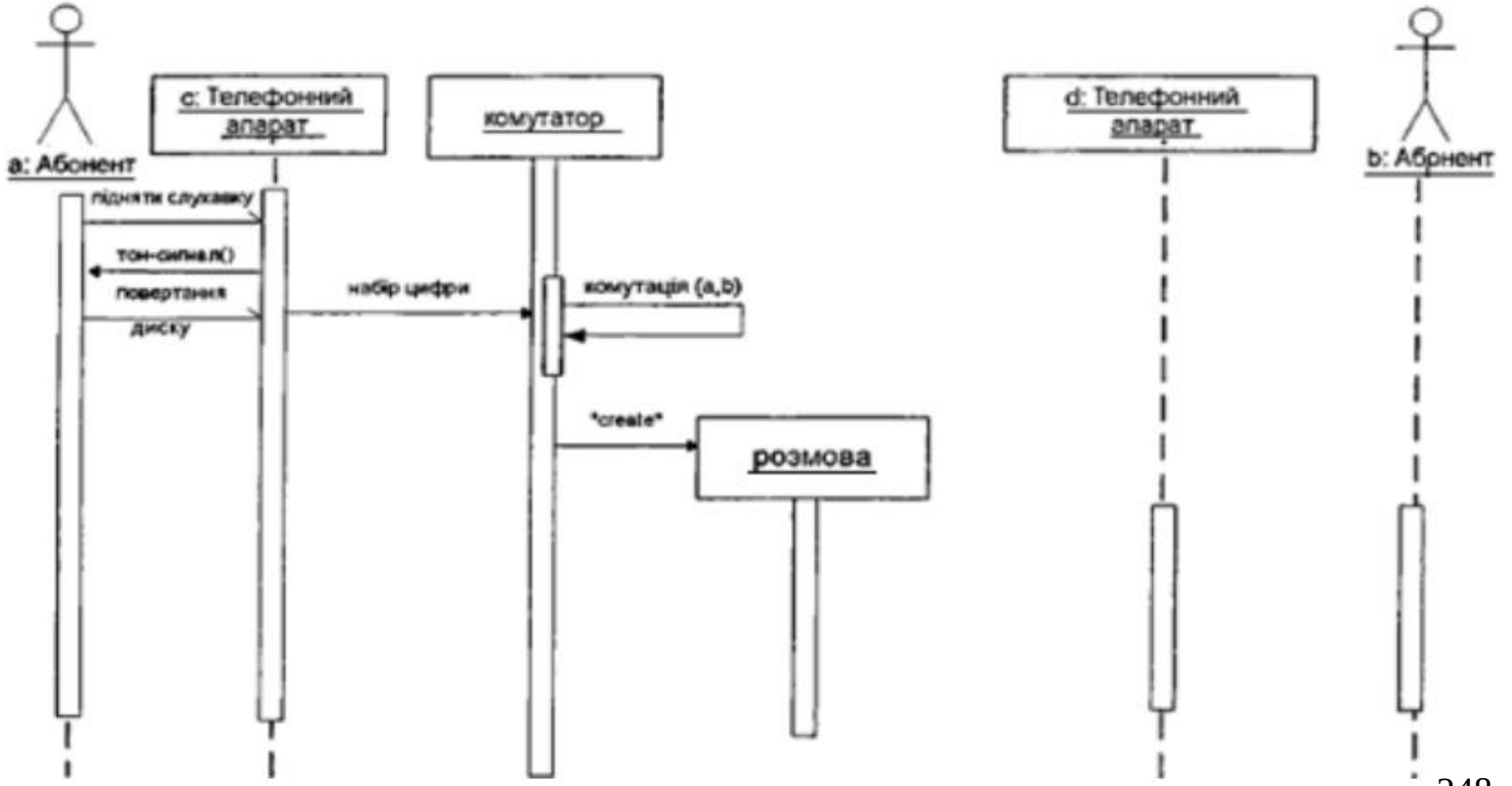
Тимчасові обмеження можуть ставитися як до виконання певних дій об’єктами, так і до самих повідомлень, явно специфікуючи умови їх передачі або прийому.

{час_прийому_повідомлення - час_відправки_повідомлення<1 сек.}
{час_очікування_відповіді<5 сек.}
{час_передачі_пакету<10 сек.}
{об’єкт_1.час_подачі_сигналу_тревоги>30сек.}

Коментарі або примітки можуть включатися в діаграми послідовності, асоціюючись з окремими об’єктами або повідомленнями.

Рисунок 12.25 Початковий фрагмент діаграми послідовності для моделювання телефонної розмови

Рисунок 12.26 Доповнений фрагмент діаграми послідовності для моделювання телефонної розмови



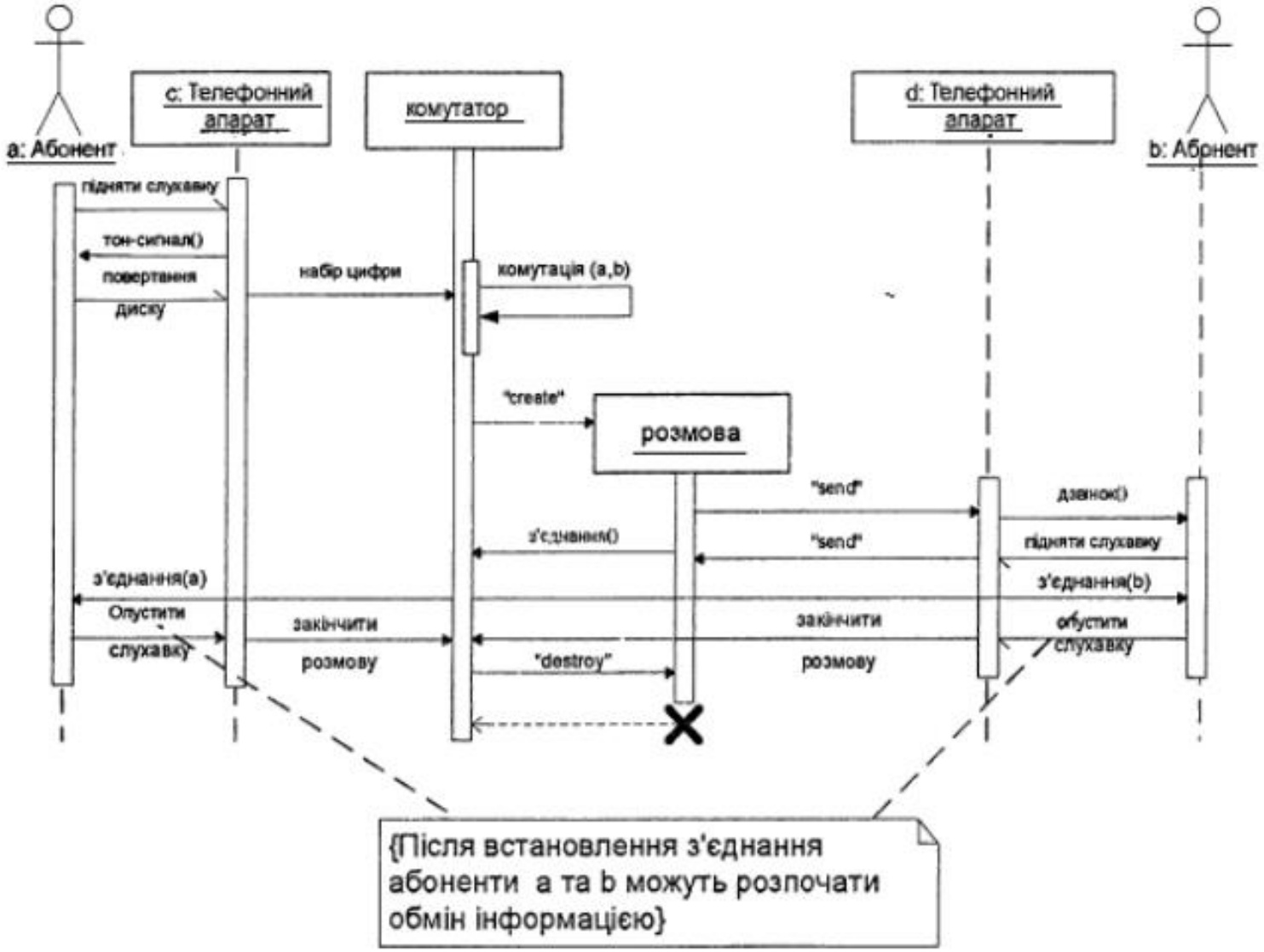


Рисунок 12.27 Остаточний варіант діаграми послідовності для моделювання телефонної розмови

Побудову діаграми послідовності доцільно починати з виділення з усієї сукупності тих і тільки тих класів, об’єкти яких беруть участь в модельованій взаємодії. Після цього всі об’єкти наносяться на діаграму з дотриманням деякого порядку ініціалізації повідомлень.

Коли об’єкти візуалізовані, можна приступати до специфікації повідомлень.

Подальша деталізація діаграми послідовності пов’язана з введенням тимчасових обмежень на виконання окремих дій в системі.

ТЕМА №13. ПРОЄКТУВАННЯ ФІЗИЧНОЇ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

13.1 Діаграма компонентів

13.2 Діаграма розміщення / розгортання

13.1 ДІАГРАМА КОМПОНЕНТІВ

Діаграма компонентів описує особливості фізичного представлення системи.

Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами.

Рисунок 13.1

Цілі розробки діаграми компонентів:

- візуалізації загальної структури вихідного коду програмної системи;
- специфікації виконуваних варіантів програмної системи;
- забезпечення багаторазового використання окремих фрагментів програмного коду;
- представлення концептуальної і фізичної схем баз даних.

Рисунок 13.2

Діаграма компонентів забезпечує узгоджений перехід від логічного представлення до конкретної реалізації проєкту у формі програмного коду; одні компоненти можуть існувати тільки на етапі компіляції програмного коду, інші – на етапі його виконання.

Діаграма компонентів відображає загальні залежності між компонентами, розглядаючи останні як класифікатори.

Компонент (Component) реалізує деякий набір інтерфейсів та служить для загального позначення елементів фізичного представлення моделі.

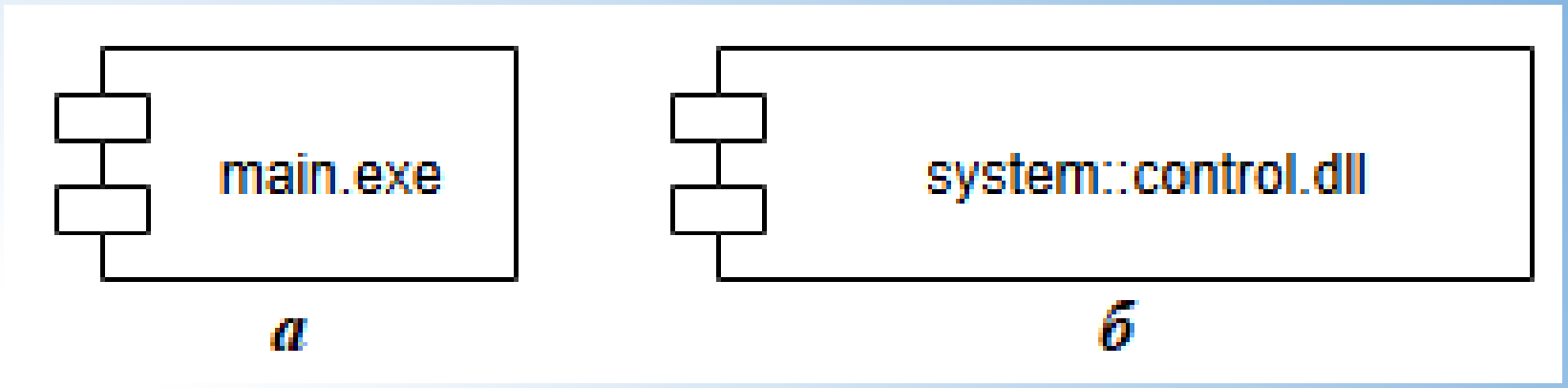


Рисунок 13.3 Графічне зображення компонента в мові UML:
а) з компонентом рівня екземпляра зв'язується тільки його ім'я;
б) додатково є ім'я пакету та позначене значення

Особливості іменування компоненту:

- ім'я компонента підпорядковується загальним правилам іменування елементів моделі в мові UML;
- може складатися з будь-якого числа букв, цифр і деяких розділових знаків.

Рисунок 13.4

Приклади простих імен компоненту:

- імена виконуваних файлів (із зазначенням розширення exe після точки-роздільника);
- імена динамічних бібліотек (розширення dll);
- імена вебсторінок (розширення html);
- імена текстових файлів (розширення txt, doc, docx);
- імена файлів довідки (hlp);
- імена файлів баз даних (ACCDB, DB, MDB, MDF, USR);
- імена файлів з вихідними текстами програм (розширення h, cpp для мови C++ і ін., в залежності від мови; asm, c, cs, java, json, .obj і ін.);
- скрипти (ps1, js, asp, aspx, php, cs, pl і ін.).

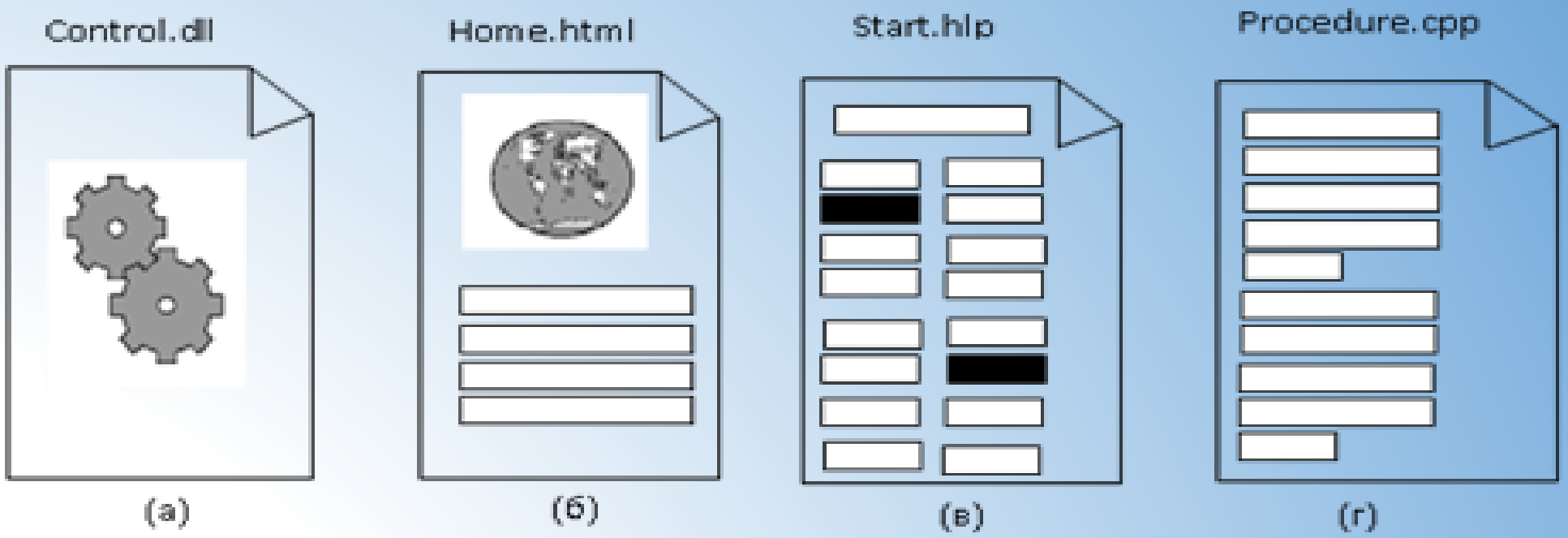


Рисунок 13.5 Варіанти графічного зображення компонентів на діаграмі компонентів

Компоненти розгортання (рис. 13.5, а, б, в) забезпечують безпосереднє виконання системою своїх функцій.

Компоненти-робочі продукти – це файли з вихідними текстами програм (рис. 13.5, г)

Компоненти виконання, що представляють виконувані модулі – файли з розширенням exe.

Спосіб специфікації різних видів компонентів при явній вказівки стереотипу компонента (перед його ім'ям):

- ***бібліотека (≪library≫)*** – визначає різновид компоненту, який представляється в формі динамічної або статичної бібліотеки;
- ***таблиця (≪table≫)*** – визначає різновид компоненту, який представляється в формі таблиці бази даних;
- ***файл (≪file≫)*** – визначає різновид компоненту, який представляється у вигляді файлів з вихідними текстами програм;
- ***документ (≪document≫)*** – визначає різновид компоненту, який представляється у формі документа;
- ***виконуваний компонент (≪executable≫)*** – визначає вид компонента, який може виконуватися в вузлі.

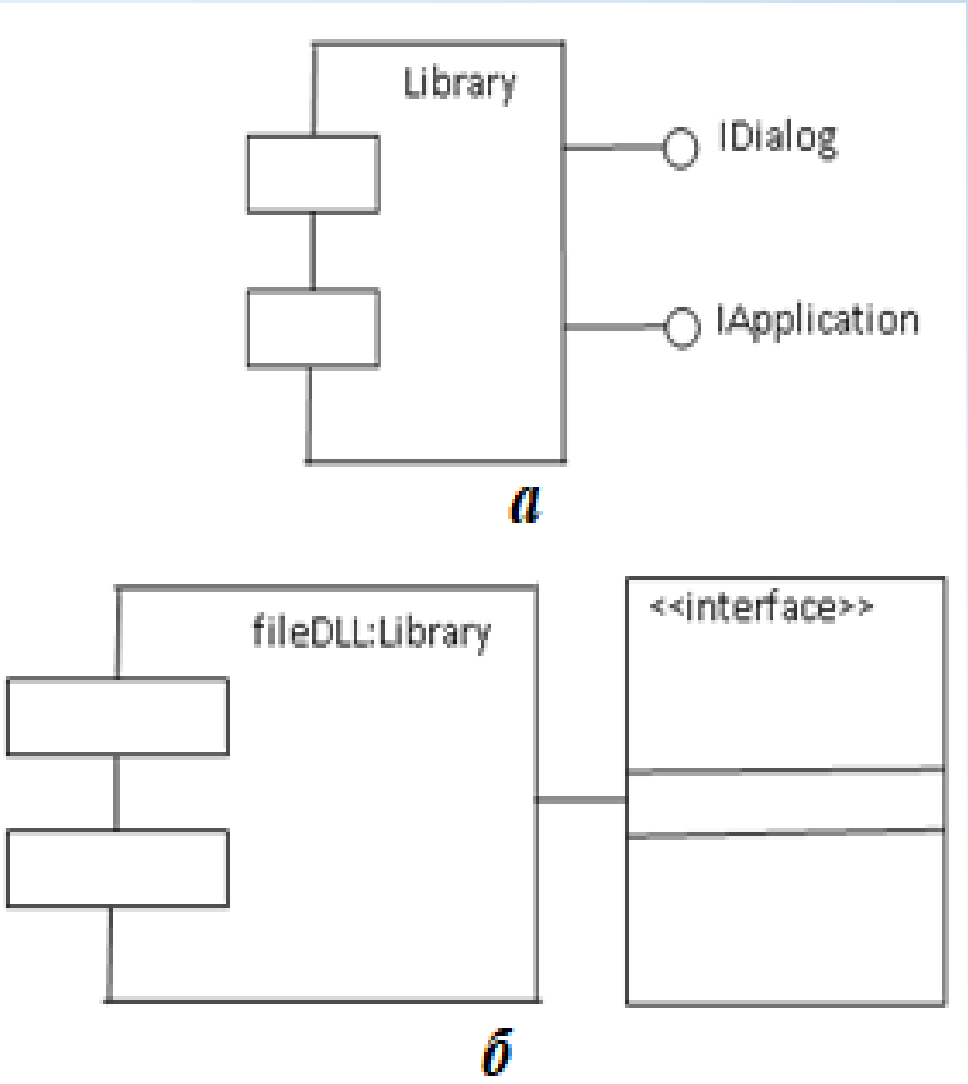


Рисунок 13.6 Графічне зображення інтерфейсів на діаграмі компонентів

Особливості графічного подання інтерфейсу:

- варіант 1: зображується колом, яке з'єднується з компонентом відрізком лінії без стрілок (рис. 13.6, а);
- варіант 2: зображення у вигляді прямокутника класу зі стереотипом «інтерфейс»/«interface» з можливими секціями атрибутів і операцій (рис. 13.6, б).

Призначення інтерфейсів – специфікація взаємодії з користувачами системи.

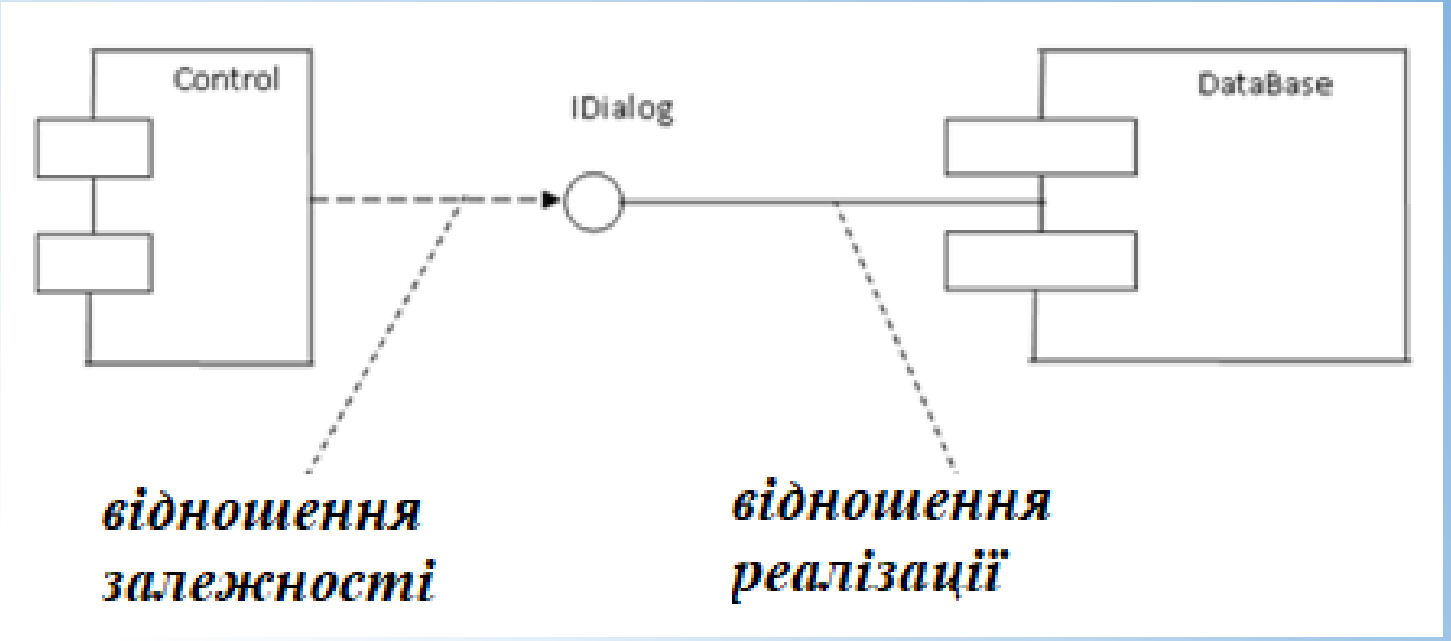
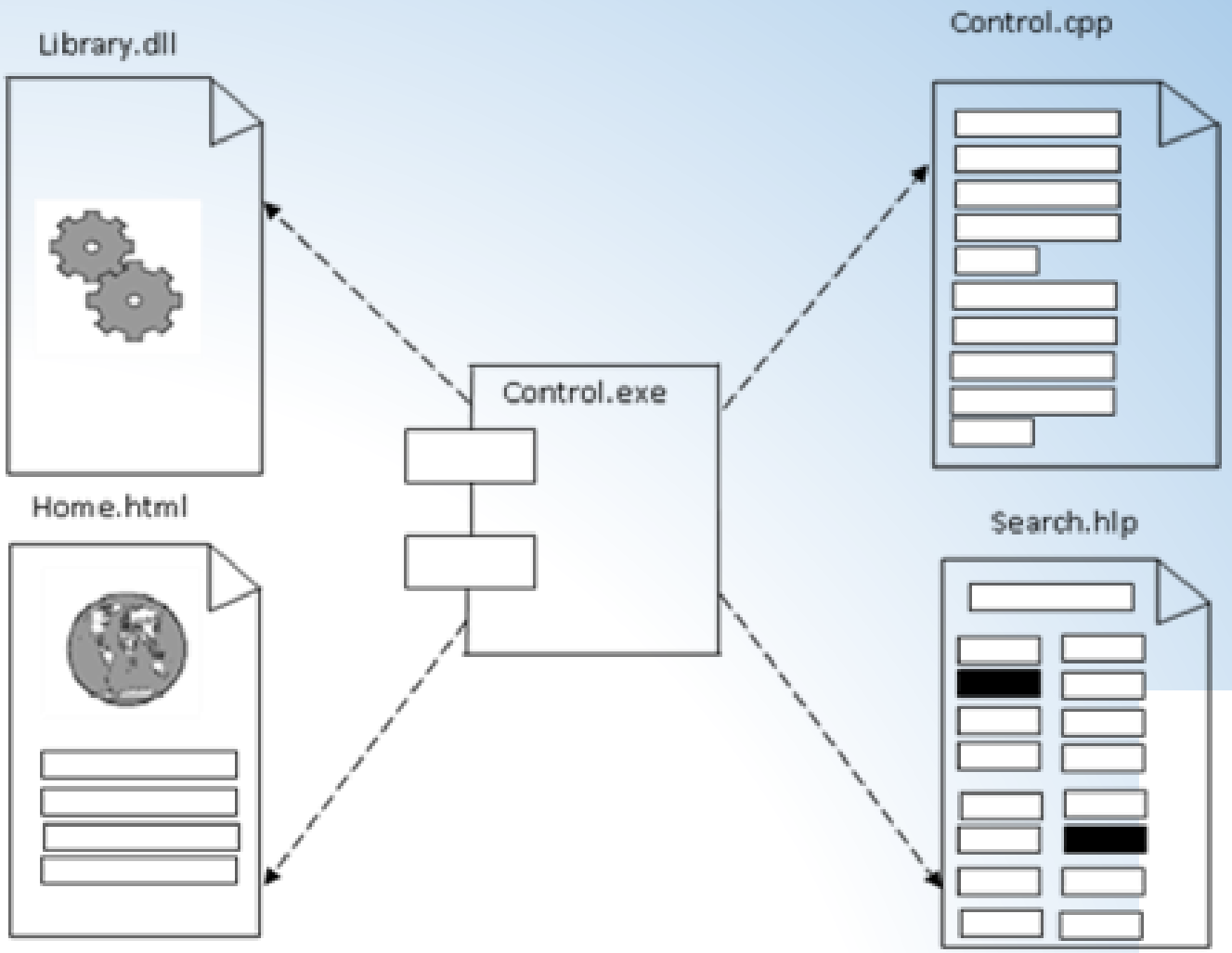


Рисунок 13.7 Фрагмент діаграми компонентів з відношенням залежності

Залежність (відношення залежності) – служить для представлення факту наявності такого зв'язку, коли зміна одного елемента моделі впливає або призводить до зміни іншого елемента моделі.

Залежності можуть пов'язувати компонент і імпортовані цим компонентом інтерфейси, а також різні види компонентів між собою.



Відношення залежності на діаграмі компонентів є відношення між різними видами компонентів (рис. 13.8).

Відносини залежності між компонентами і реалізованими в них класами (рис. 13.9).

Рис. 13.8 Графічне зображення відношення залежності між компонентами

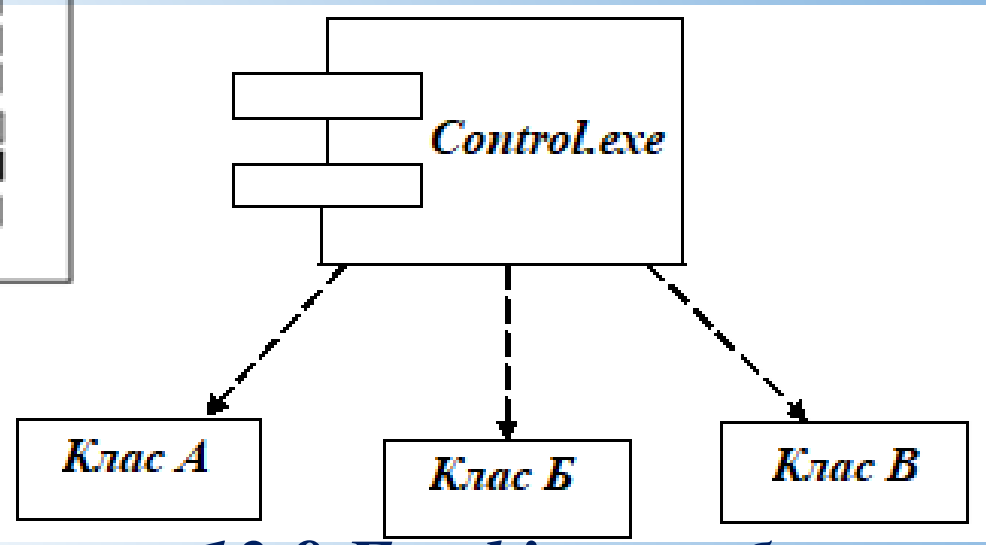


Рисунок 13.9 Графічне зображення залежності між компонентом і класами

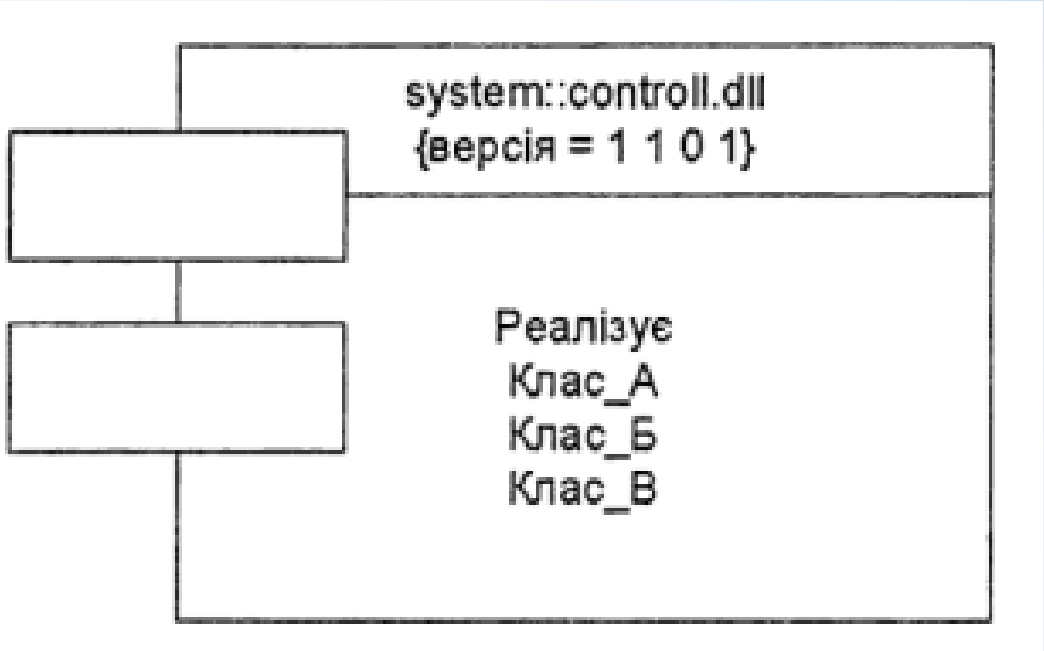


Рисунок 13.10 Графічне зображення компонента з додатковою інформацією про реалізовані їм класах

Якщо потрібно підкреслити, що деякий компонент реалізує окремі класи, то для позначення компонента використовується розширений символ прямокутника. При цьому прямокутник компонента ділиться на дві секції горизонтальною лінією. ***Верхня секція*** служить для запису імені компонента, а ***нижня секція*** – для вказівки додаткової інформації (рис. 13.10).

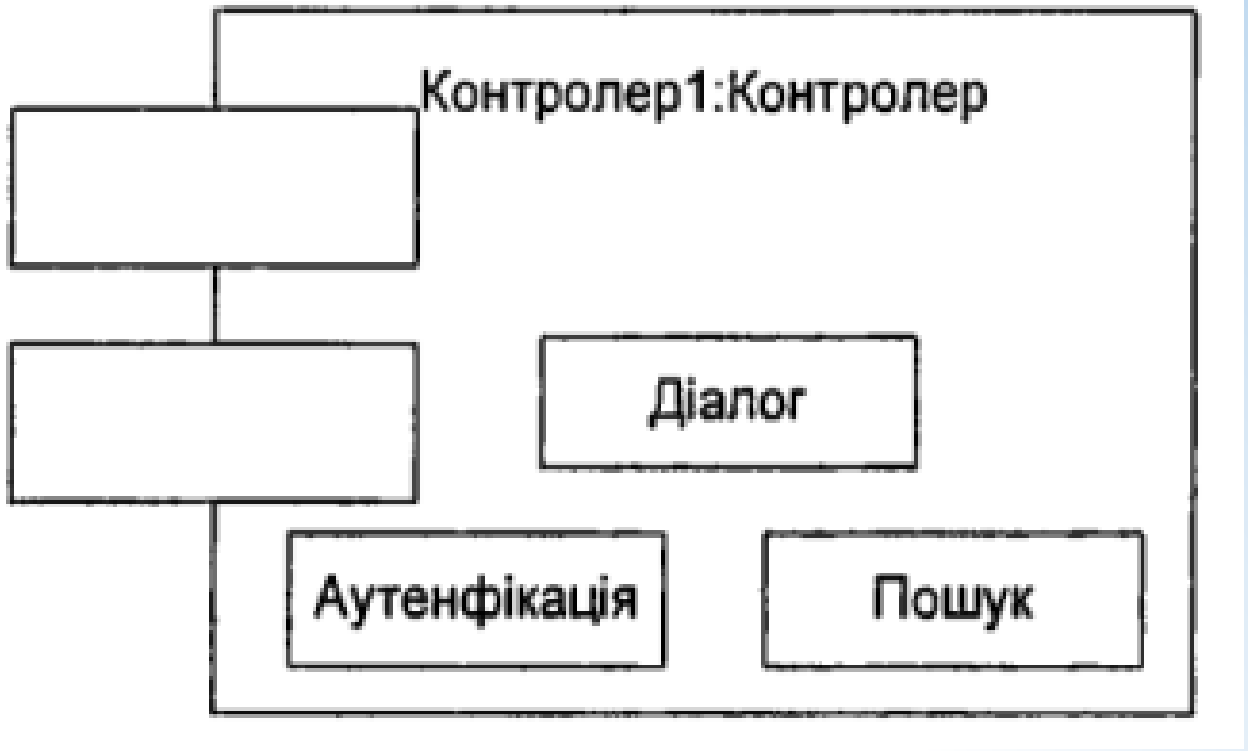


Рисунок 13.11 Графічне зображення компонента рівня екземпляра, що реалізує окремі об'єкти

Всередині символу компоненту можуть зображуватися інші елементи графічної нотації, такі як класи (компонент рівня типу) або об'єкти (компонент рівня екземпляра).

Об'єкти, які знаходяться в окремому компоненті-екземплярі, зображуються вкладеними в символ даного компонента та означає, що виконання компонента тягне виконання відповідних об'єктів.

Рекомендації з побудови діаграми компонентів

- До початку розробки необхідно прийняти рішення про вибір обчислювальних платформ і операційних систем, на яких передбачається реалізовувати систему, а також про вибір конкретних баз даних і мов програмування.
- Вирішити, з яких фізичних частин (файлів) буде складатися програмна система.
- Після загальної структуризації фізичного представлення системи необхідно доповнити модель інтерфейсами і схемами бази даних.
- На завершальному етапі побудови діаграми компонентів відбувається встановлення і нанесення на діаграму взаємозв'язків між компонентами, а також відносин реалізації.
- При розробці діаграми компонентів слід дотримуватися загальних принципів створення моделей на мові UML.
- Діаграма компонентів, як правило, розробляється спільно з діаграмою розгортання, на якій подається інформація про фізичне розміщення компонентів програмної системи по її окремих вузлах.

13.2 ДІАГРАМА РОЗМІЩЕННЯ / РОЗГОРТАННЯ

Діаграма розміщення / розгортання – UML-діаграма, на якій відображаються обчислювальні вузли під час роботи програми, компоненти та об’єкти, що виконуються на цих вузлах.

Функціональні можливості діаграми розміщення / розгортання:

- застосовується для подання загальної структури і топології системи;
- містить зображення розміщення компонентів поміж окремими вузлами системи;
- показує наявність фізичних з’єднань;
- відображає робочі екземпляри компонентів;
- моделює фізичне розгортання артефактів на вузлах.

Компоненти, наведені на діаграмі розгортання відповідають представленню робочих екземплярів одиниць коду.

Вузли представляються як прямокутні паралелепіпеди з артефактами, розташованими в них, та можуть мати підвузли, які представляються як вкладені прямокутні паралелепіпеди.

Типи вузлів:

- вузол пристрою;
- вузол середовища виконання.

Форми подання вузлів:

- на рівні класу (рис. 13.12, а);
- на рівні екземпляра (рис. 13.12, б)

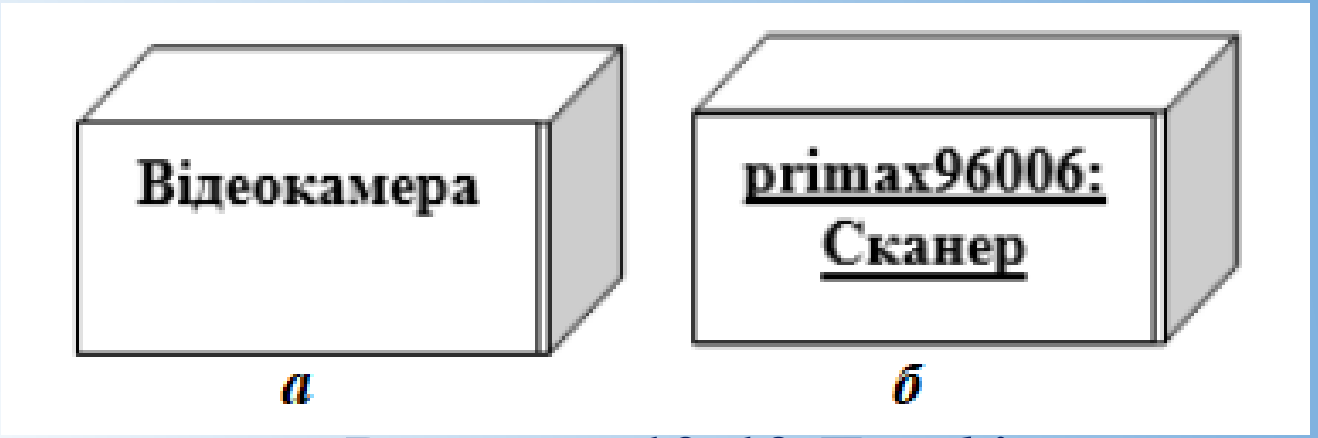


Рисунок 13.12 Графічне зображення вузла на діаграмі розгортання

Зображення вузлів можуть розширюватися, щоб включити додаткову інформацію про вузол (рис. 13.13).

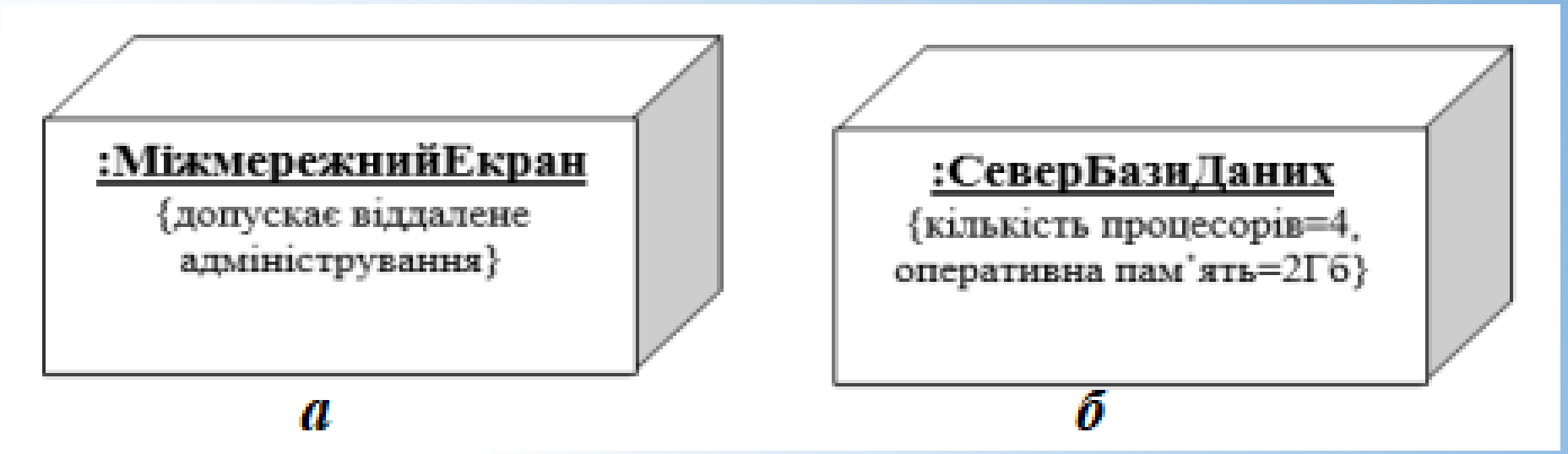
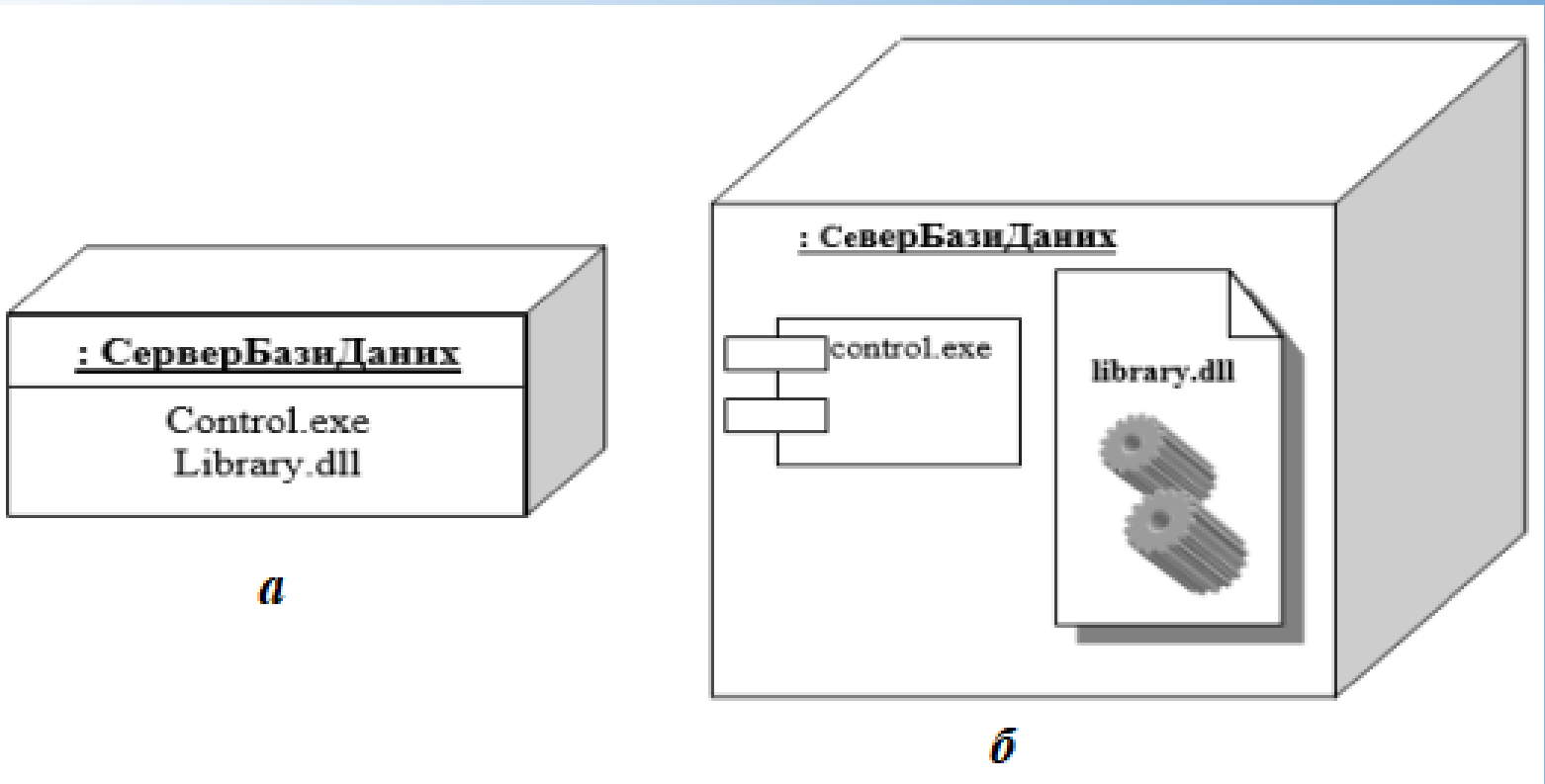


Рисунок 13.13 Графічне зображення вузла-екземпляра з додатковою інформацією у формі поміченого значення

Можна вказати компоненти, які розміщуються або виконуються на окремому вузлі:

- розділити графічний символ вузла на дві секції горизонтальною лінією. У верхній секції – ім’я вузла, а в нижній – розміщені на вузлі компоненти (рис. 13.14, а).
- дозволяє показувати на діаграмі розгортання вузли з вкладеними зображеннями компонентів (рис. 13.14, б).

Рисунок 13.14 Варіанти графічного зображення вузлів-екземплярів з розміщуваними на них компонентами



Спеціальні графічні стереотипи для позначення різновидів вузлів:

- Перший – позначає ресурсоємний вузол, під яким розуміється вузол з процесором і пам’яттю (рис. 13.15, а).
- Другий стереотип у формі звичайного куба позначає пристрій (device), під яким розуміється вузол без процесора і пам’яті (рис. 13.15, б).
- Додаткові графічні стереотипи, які покращують наочність зображення діаграм розгортання (рис. 13.15, в).

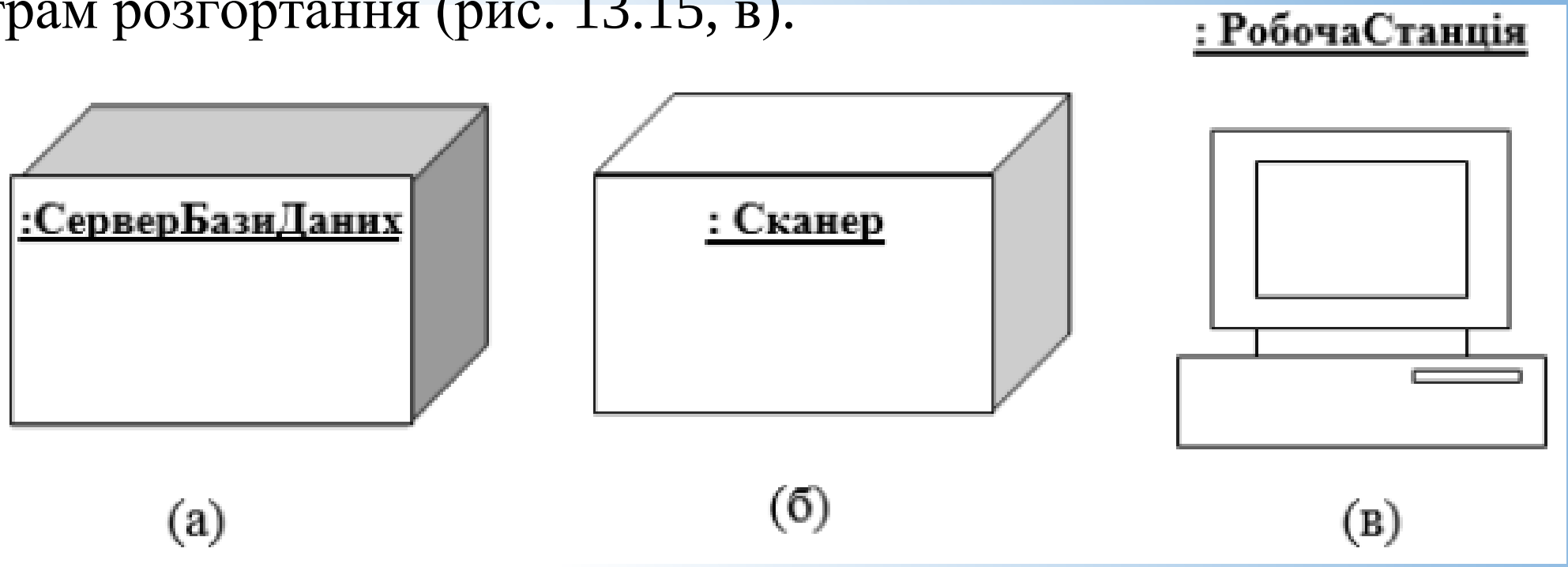


Рисунок 13.15 Варіанти зображення графічних стереотипів вузлів

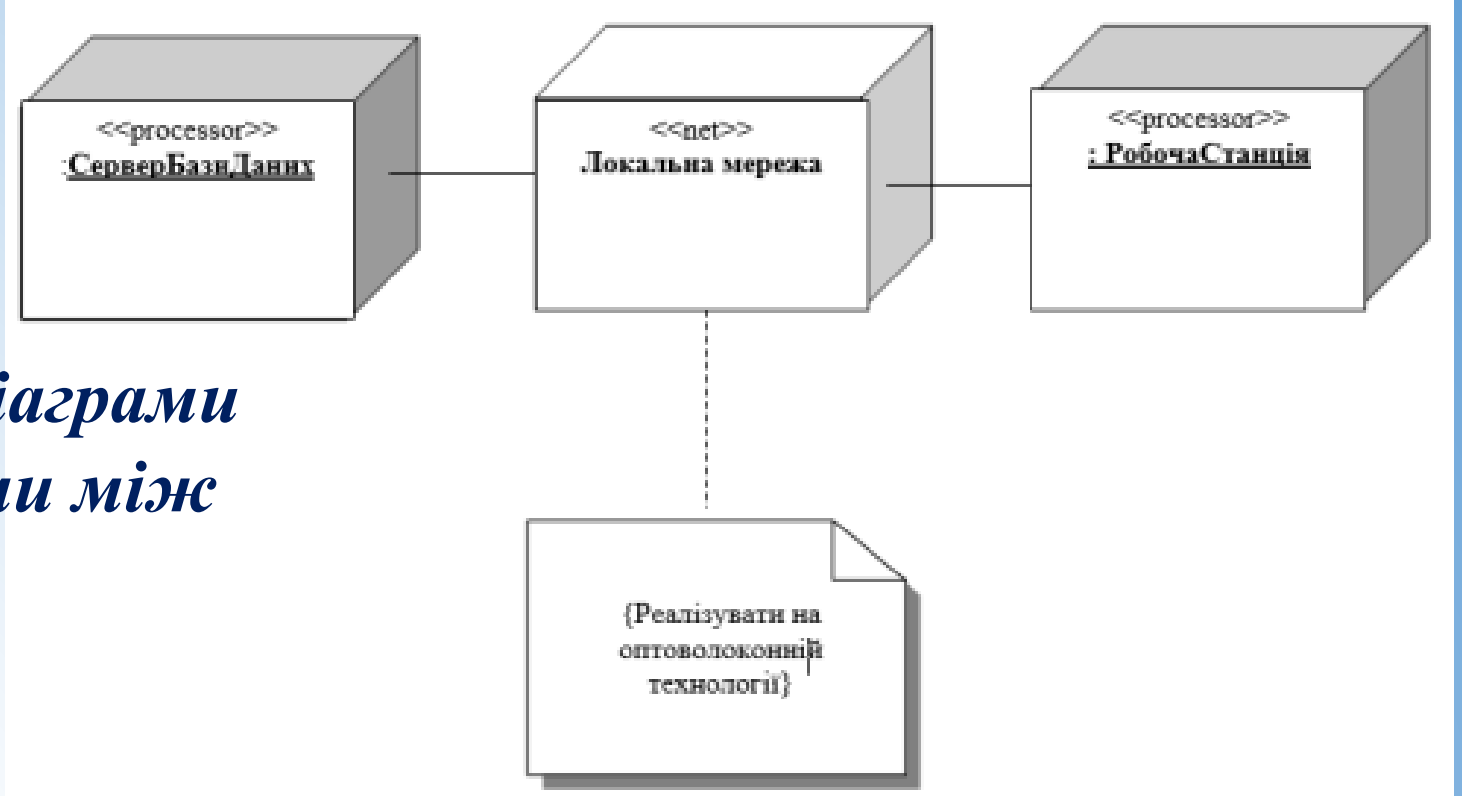


Рисунок 13.16 Фрагмент діаграми розгортання із з'єднаннями між вузлами

Відношення 1:

- фізичні з'єднання між вузлами (є різновидом асоціації), а також залежності між вузлами та компонентами;
- зображуються відрізками ліній без стрілок;
- наявність лінії вказує на необхідність організації фізичного каналу для обміну інформацією між відповідними вузлами;
- характер з'єднання може бути додатково специфікований приміткою, стереотипом, поміченим значенням або обмеженням

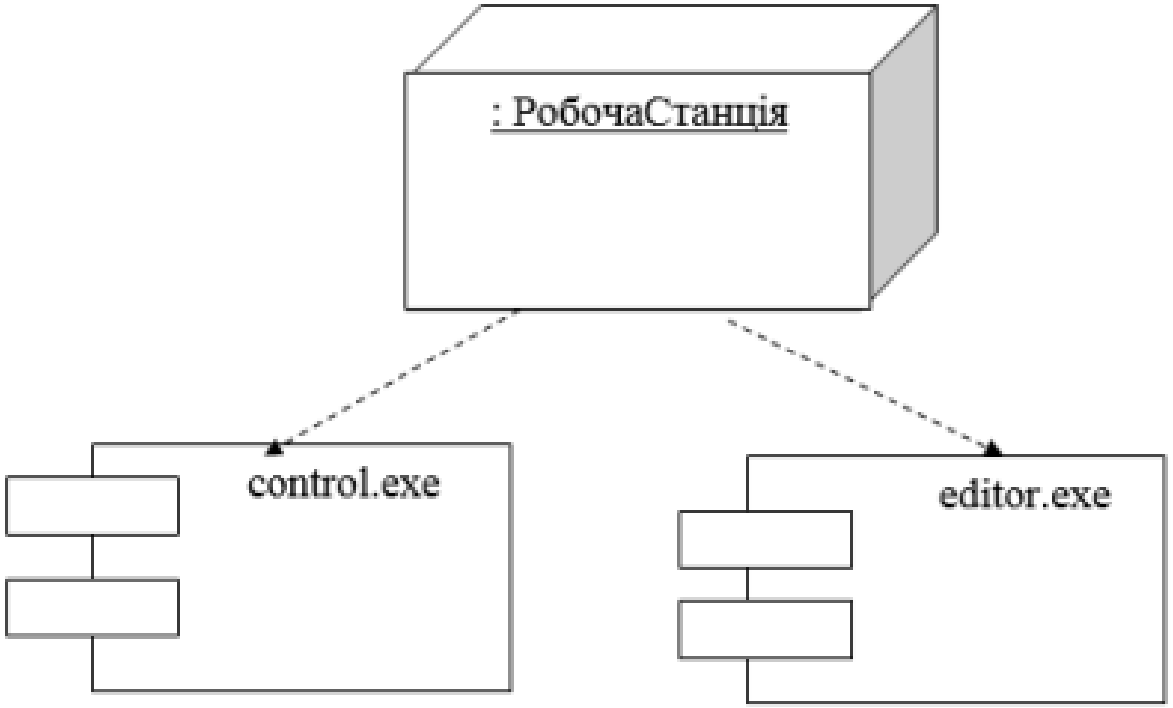


Рисунок 13.17 Діаграма розгортання з відношенням залежності між вузлом і розгорнутими на ньому компонентами

Відношення 2:

- відношення залежності між вузлом і розміщеними на ньому компонентами

Розробка інформаційних систем, що забезпечують доступ в режимі реального часу, передбачає не лише створення програмного коду, але і використання додаткових апаратних засобів.

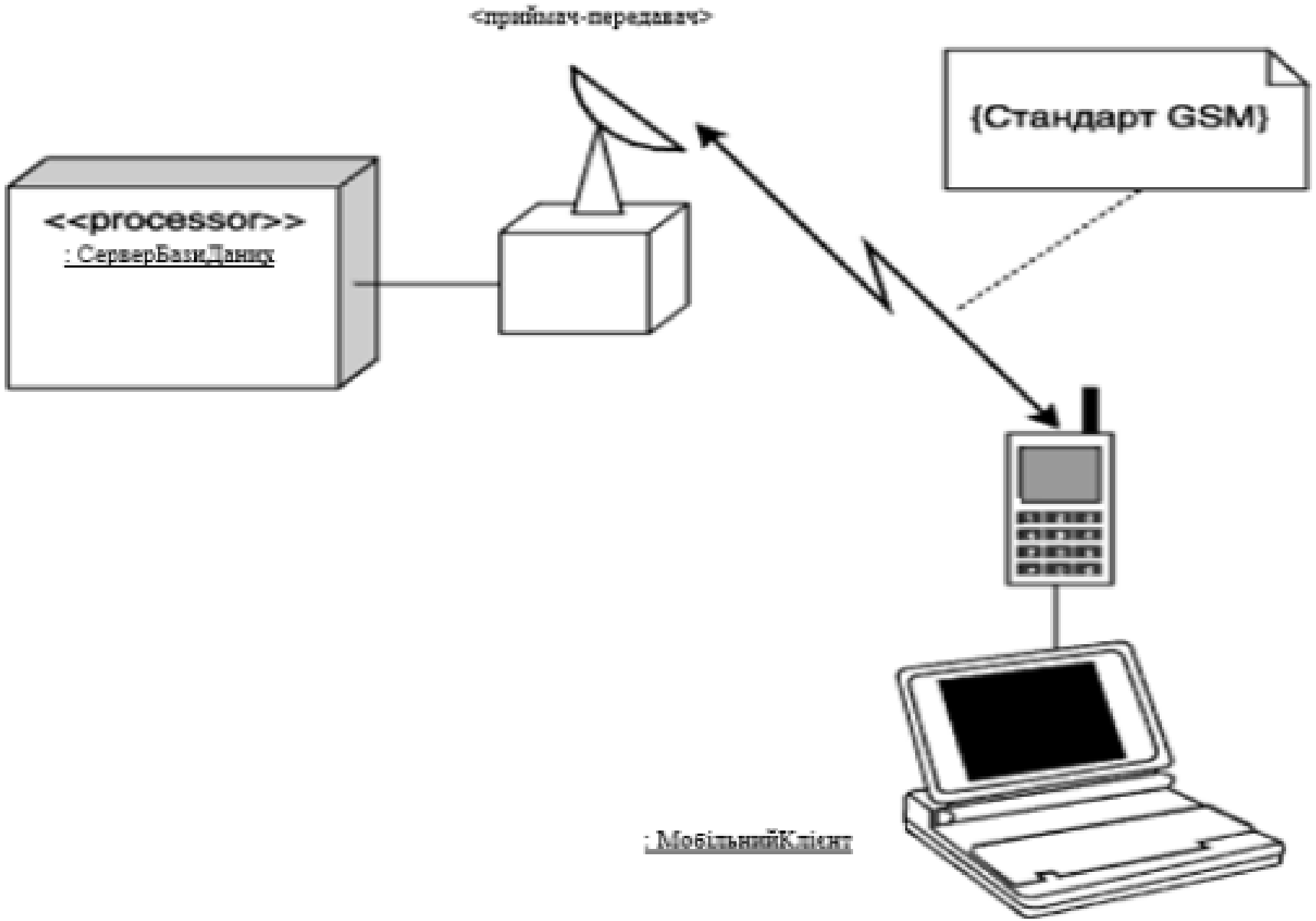


Рисунок 13.18 Діаграма розгортання для системи мобільного доступу до корпоративної бази даних

Рекомендації з побудови діаграми розгортання

- Розробка діаграми розгортання починається з ідентифікації всіх апаратних, механічних і інших типів пристроїв, які необхідні для виконання системою всіх функцій. Спочатку специфікуються ресурсоємні вузли системи, що мають процесор і пам’ять.

- Деталізація діаграми розгортання пов’язана з розміщенням всіх виконуваних компонентів діаграми поміж вузлами системи.

- Розробка діаграми розгортання здійснюється на завершальному етапі ООАП. З іншого боку, діаграма розгортання може будуватися для аналізу існуючої системи з метою її подальшої деталізації і модифікації.

ТЕМА №14. ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ СИСТЕМ

Тематичний план

14.1 Загальні питання інформаційної моделі предметної області

14.2 Огляд методології IDEF1X

14.1 ЗАГАЛЬНІ ПИТАННЯ ІНФОРМАЦІЙНОЇ МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ

Рисунок 14.1

Об'єктна (інформаційна, інфологічна) модель – це модель інфологічного рівня представлення, в якій акцентується інформаційний (структурний) аспект моделювання предметної області.

Сутність відображає матеріальні, фінансові, трудові інформаційні потоки або ресурси.

Різновиди сутностей: статичні, динамічні

Механізми представлення різноманіття об'єктів ПрО в моделі:

- ***Узагальнення*** – це механізм створення сутності шляхом об'єднання її за класифікаційними ознаками;
- ***Агрегація*** – це механізм створення сутності шляхом об'єднання різних характеристик незалежних об'єктів.

Зв'язкам ієрархій агрегації та узагальнення може бути дана наступна інтерпретація:

зв'язок в ієрархії узагальнення можна прочитати як „є" („is-a"), а зв'язок в ієрархії агрегації – як „є частиною" („is-part-of").

Характеристика сутностей (атрибут, властивість) – це опис сутності в термінах ПрО.

Типи атрибутів сутності:

- ознакові атрибути
- кількісні атрибути

Вибір сутностей та їх атрибутів з ПрО здійснює тільки в залежності від поставленої задачі.

Характеристика зв'язків між об'єктами – це опис структурної залежності, що виникає в ПрО між різними об'єктами.

Функціональні зв'язки між об'єктами описуються смисловими дієсловами ПрО і не відображаються в об'єктній моделі ПрО.

Структурний зв'язок між сутностями показує, скільки об'єктів однієї сутності залежать від якої кількості об'єктів іншої сутності.

Рисунок 14.2

Засоби створення об’єктної моделі предметної області:

- діаграми «сутність-зв’язок» (ERD);
- діаграми класів UML;
- діаграми класифікацій та композиційні схеми системно-онтологічного аналізу.

Діаграми «сутність-зв'язок» (ERD)

Діаграма «сутність-зв'язок» була ***запропонована*** в 1976 р. Пітером Пін-Шен Ченом.

Методологія моделювання сутностей та зв'язків ***використовує*** діаграмну техніку для подання об'єктної моделі предметної області – структури інформації в термінах об'єктів (сутностей), їх властивостей (атрибутів) і відносин (зв'язків).

Діаграма «сутність-зв'язок» ***ґрунтується*** на важливій семантичній інформації про навколишній світ і призначена для логічного представлення даних. Вона визначає значення даних в контексті їх взаємозв'язку з іншими даними.

При розробці ER-моделей необхідно отримати наступну інформацію про предметну область:

- Специфікація сутностей предметної області.
- Специфікація атрибутів сутностей.
- Специфікація взаємозв'язків між сутностями.

Графічні елементи діаграми «сутність-зв’язок» (нотація Чена)

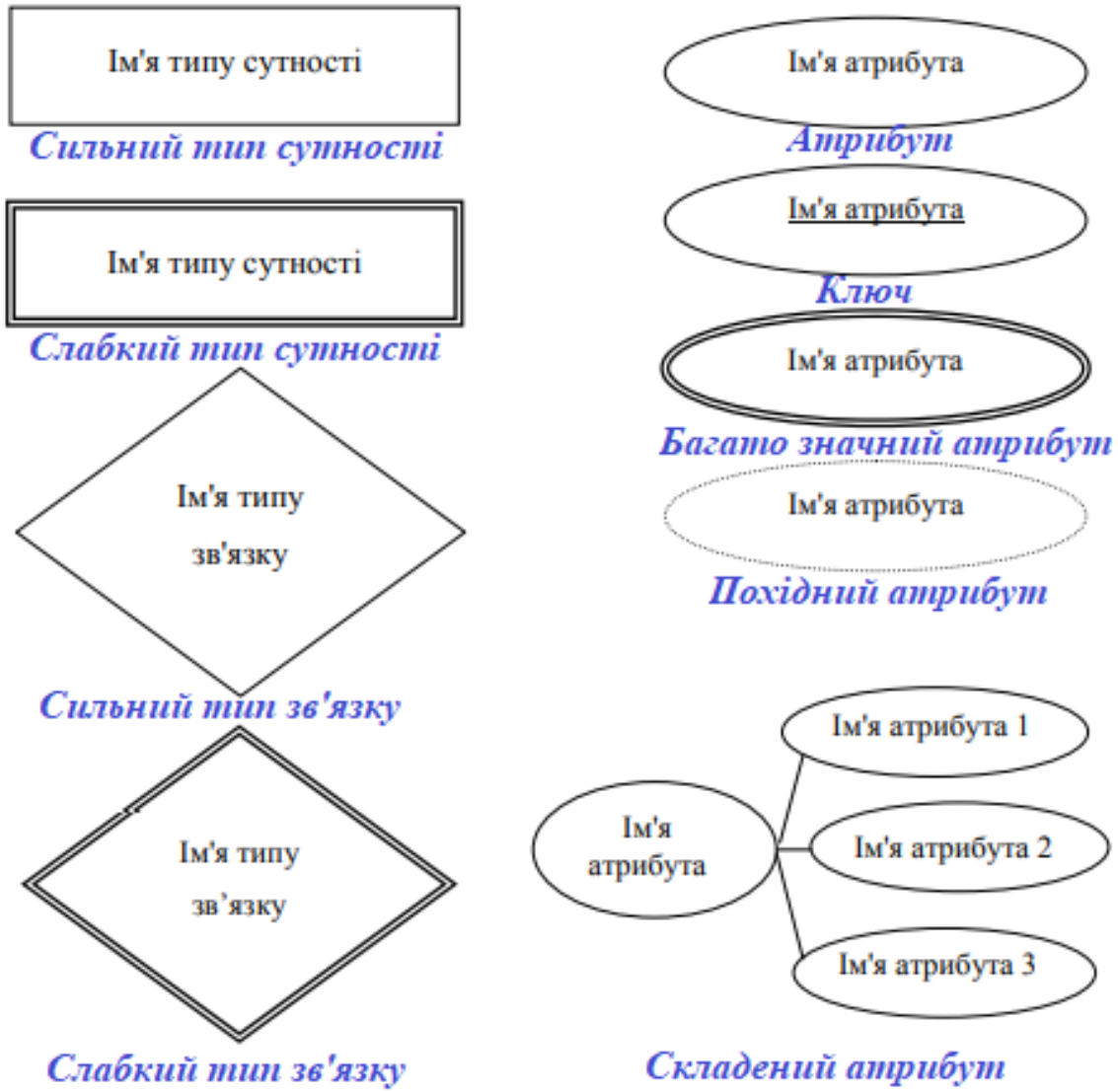
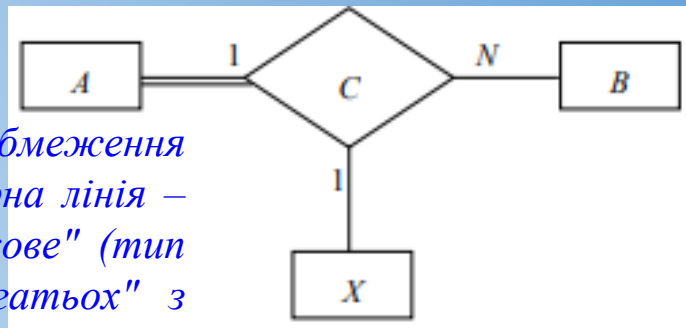


Рисунок 14.3

Позначення значень тах обмеження кардинальності типу зв'язку: 1 – "один", N – "багато"

Позначення значень тор обмеження кардинальності типу зв'язку: одинарна лінія – "необов'язкове", подвійна – "обов'язкове" (тип зв'язку С "один до одного до багатьох" з обов'язковою участю типу сутності А і необов'язковою – типів сутностей В та Х)



Тип зв'язку суперклас/підклас (концепція уточнення/узагальнення). Літера в колі позначає значення обмеження неперетину, що застосовують до типу сутності суперклас і його типів сутностей підклас: "d" позначає неперетинне обмеження, "o" – перетинне. За допомогою лінії, що з'єднує коло й тип сутності суперклас, показують обмеження участі: подвійна лінія – обов'язкова участь, одинарна – необов'язкова

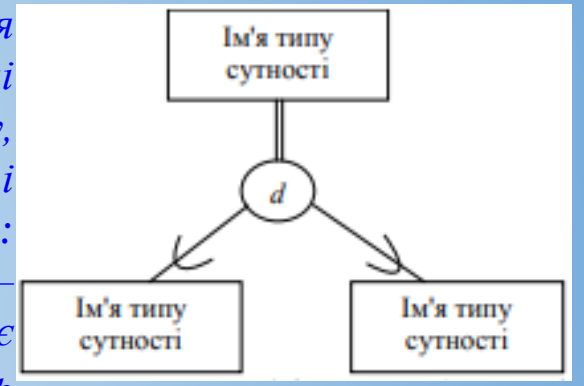
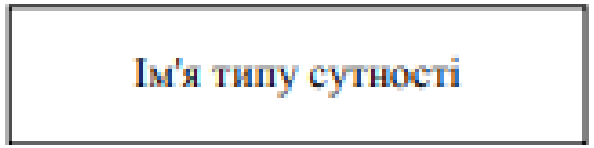


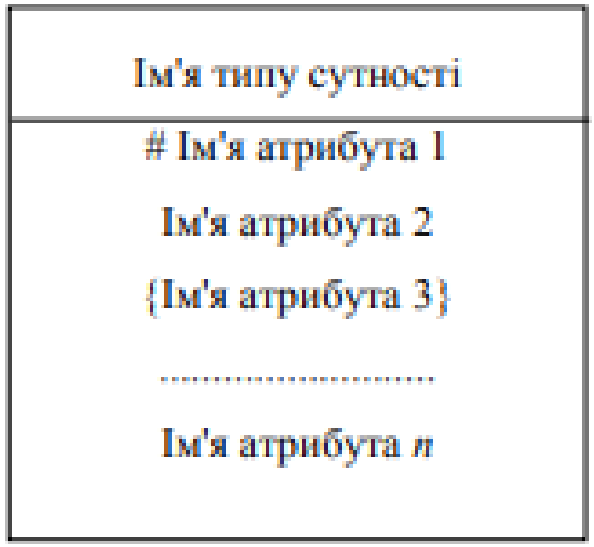
Рисунок 14.4

Графічні елементи діаграми «сутність-зв’язок» (нотація Баркера)



Тип сутності

Тип зв’язку



Атрибути перераховують у нижній частині позначення типу сутності. Первинний ключ позначають #. Багатозначні атрибути поміщають у фігурні дужки

Рисунок 14.5

Позначення значень тах обмеження кардинальності типу зв'язку: одинарна лінія – "один", лінія з лапкою – "багато"; позначення значень тор обмеження кардинальності типу зв'язку: пунктирна лінія – "необов'язкове", суцільна – "обов'язкове" (тип зв'язку С "один до багатьох" з обов'язковою участю типу сутності В та необов'язковою – типу сутності А)

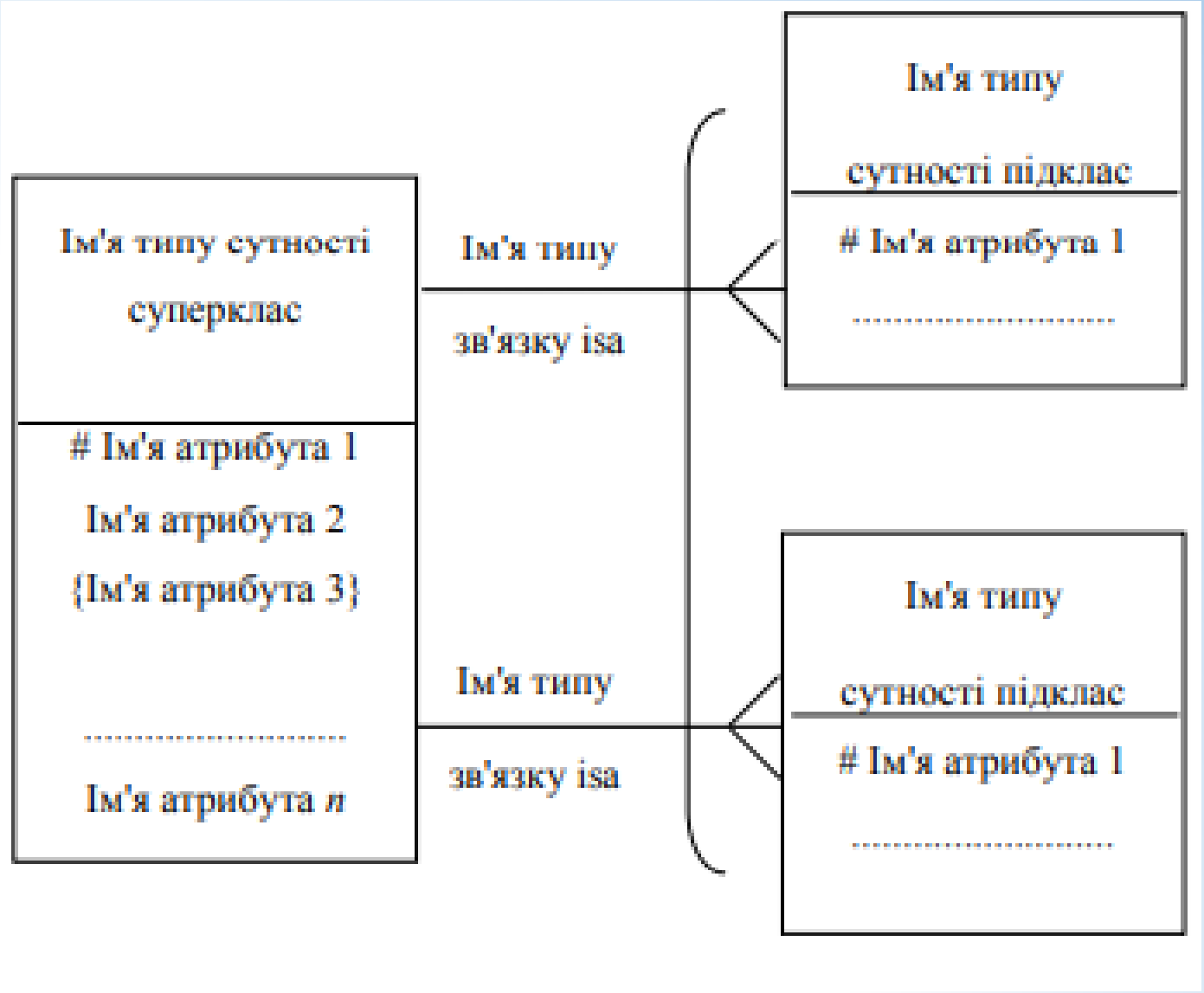
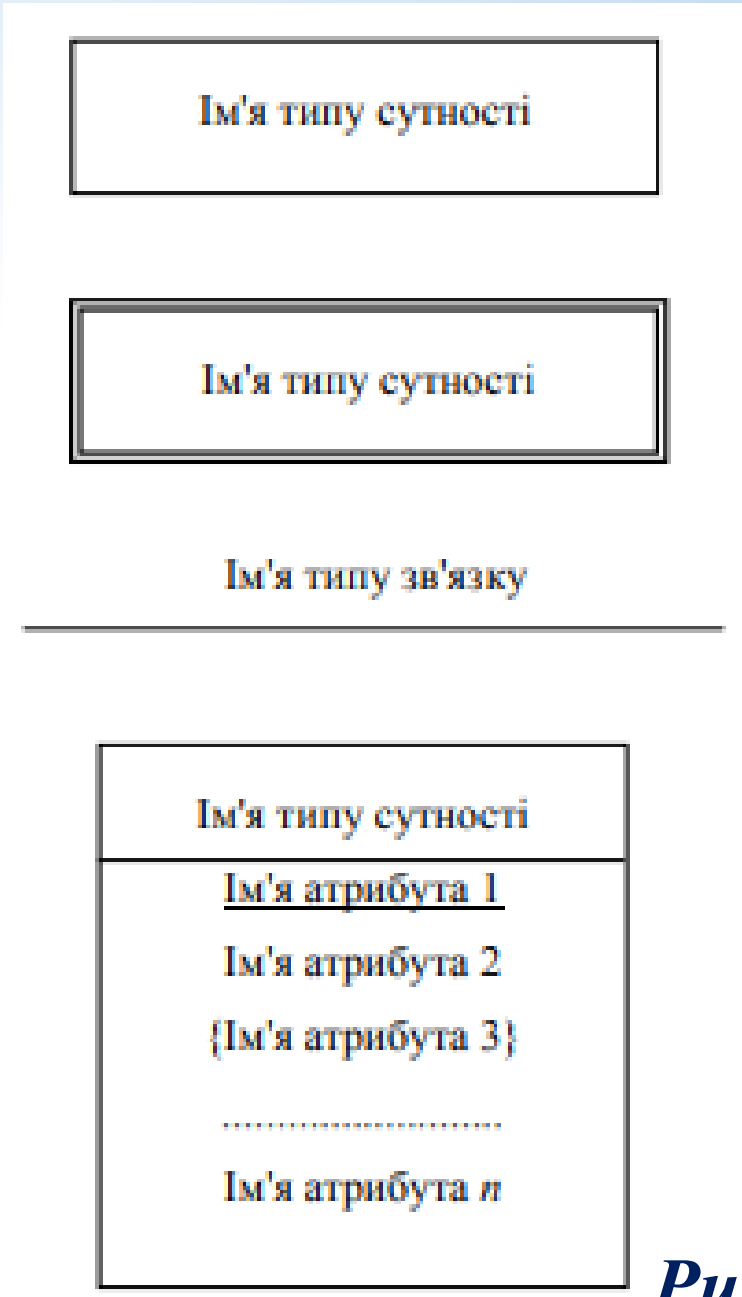


Рисунок 14.6

Тип зв'язку суперклас/підклас (концепція уточнення/узагальнення): використовується елемент "дуга" для об'єднання всіх типів зв'язку isa, що пов'язують єдиний тип сутності суперклас і його типи сутностей підклас

Графічні елементи діаграми «сутність-зв'язок» (нотація Мартіна)



Сильний тип сутності

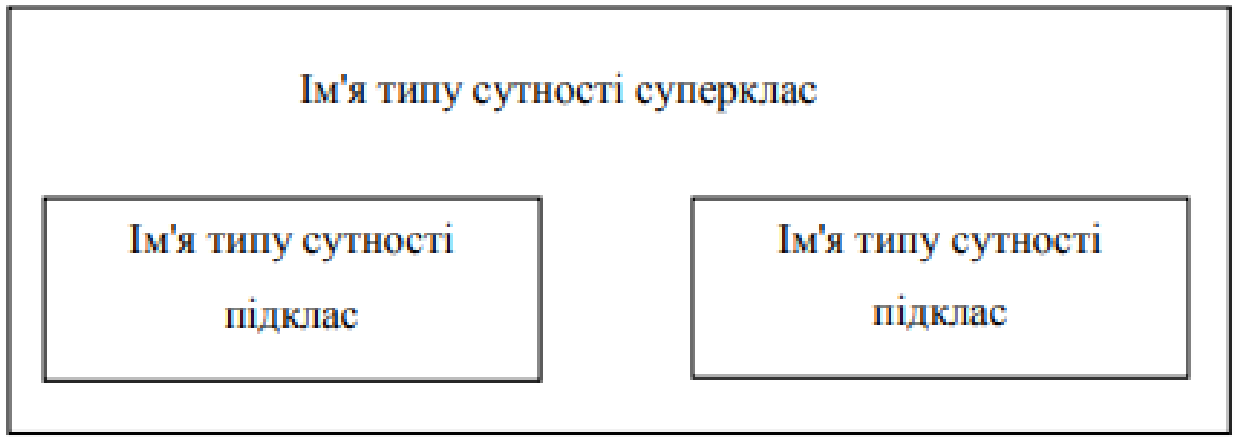
Слабкий тип зв'язку

Тип зв'язку

Атрибути перераховують у нижній частині позначення типу сутності. Первинний ключ підкреслюють. Багатозначні атрибути поміщають у фігурні дужки

Рисунок 14.7

Позначення значень min та max обмежень кардинальності типу зв'язку: коло – "нуль", одна риска – "один", лапка – "багато" (значення min та max обмежень кардинальності типу зв'язку C для типу сутності A – (1,1), а для типу сутності B – (0,M)



Тип зв'язку суперклас/підклас (концепція уточнення/узагалінення)

Рисунок 14.8

Сутність (Entity) – безліч екземплярів реальних або абстрактних об’єктів предметної області, які володіють загальними властивостями.

Сутність повинна володіти наступними властивостями:

- мати унікальне ім’я;
- до одного і того ж імені повинна завжди застосовуватися одна й та ж інтерпретація;
- одна і та ж інтерпретація не може застосовуватися до різних імен, якщо тільки вони не є псевдонімами;

Рисунок 14.9

- мати один або кілька атрибутів, які або належать сутності, або наслідуються через зв’язок;
- мати один або кілька атрибутів, які однозначно ідентифікують кожен екземпляр сутності.

Зв'язок (Relationship) – це асоціація між сутностями, при якій кожен екземпляр однієї сутності асоційований з довільною (у тому числі нульовою) кількістю екземплярів іншої сутності.
Зв'язок показує, як в предметній області інформаційні об'єкти взаємодіють один з одним. Між двома сутностями може бути визначено декілька зв'язків, кожен з яких володіє своєю семантикою. Наявність множини зв'язків і визначає складність інфологічних моделей.

Атрибут (Attribute) – будь-яка характеристика сутності, значима для розглядуваної предметної області і призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або вираження стану сутності.
Екземпляр атрибута – це певна характеристика окремого елемента множини та визначається типом характеристики і її значенням (значенням атрибута).

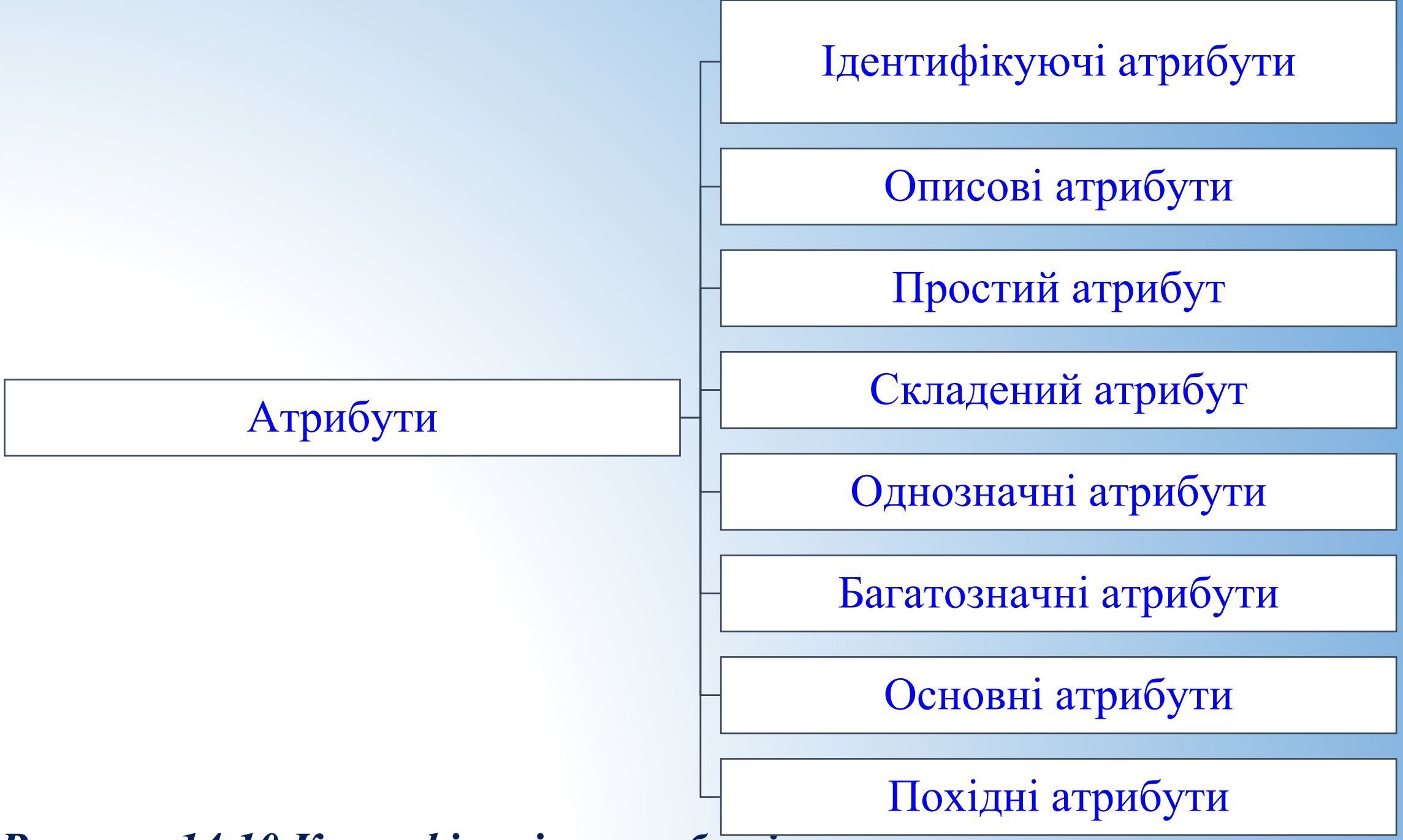


Рисунок 14.10 Класифікація атрибутів

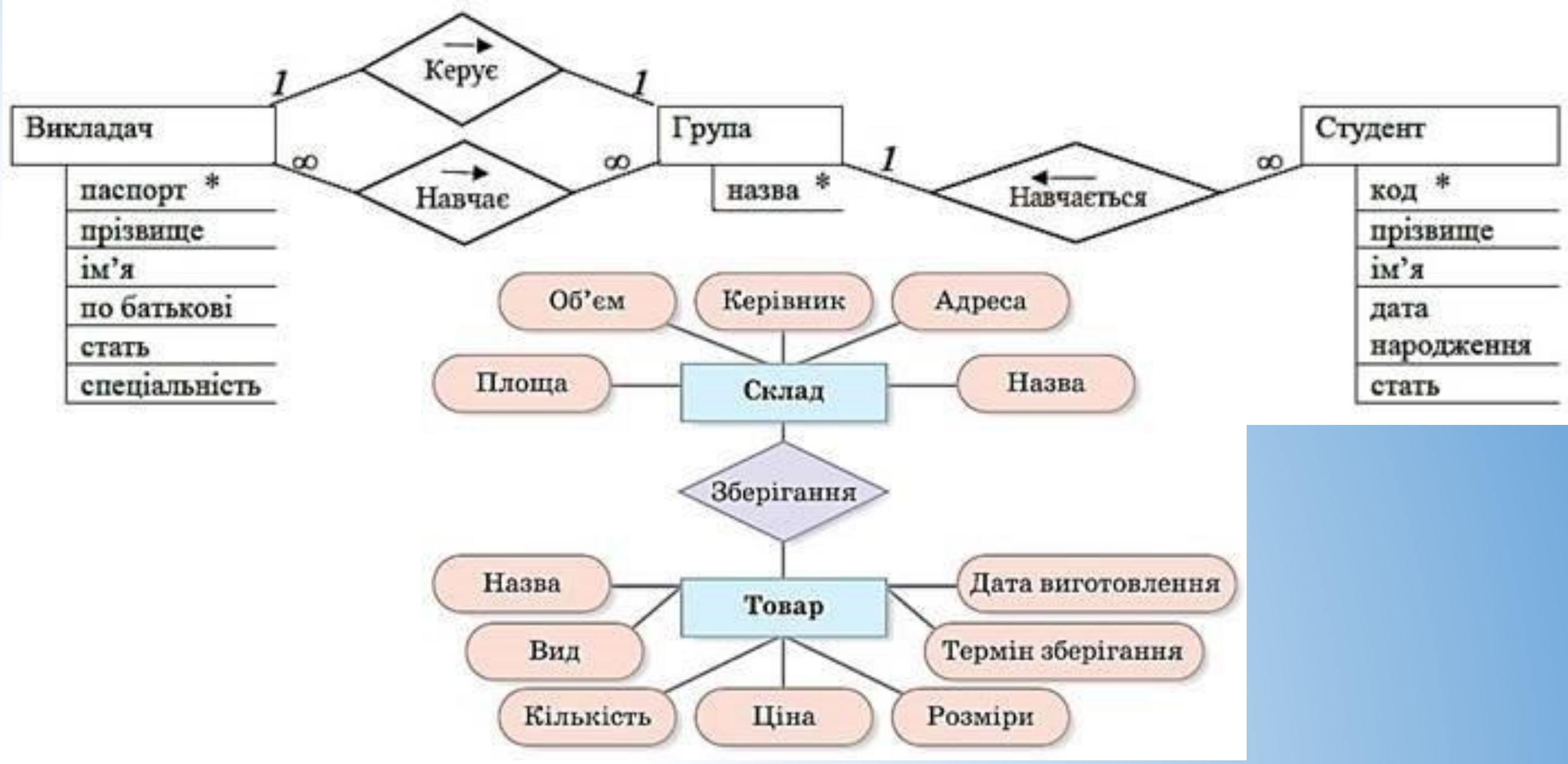


Рисунок 14.11 Готові об'єктні моделі ПрО, побудована засобами діаграми «сутність-зв'язок»

Об’єктна модель може бути представлена засобами ***діаграми класів UML***. Діаграма класів ***включає*** основні класи, необхідні для реалізації виділених варіантів використання, а також можливі зв’язки між класами.

Клас (Class) – опис сукупності однорідних об’єктів з їх атрибутами, операціями, відносинами і семантикою та в мові UML служить для позначення множини об’єктів, які володіють однаковою структурою, поведінкою і відносинами з об’єктами інших класів.

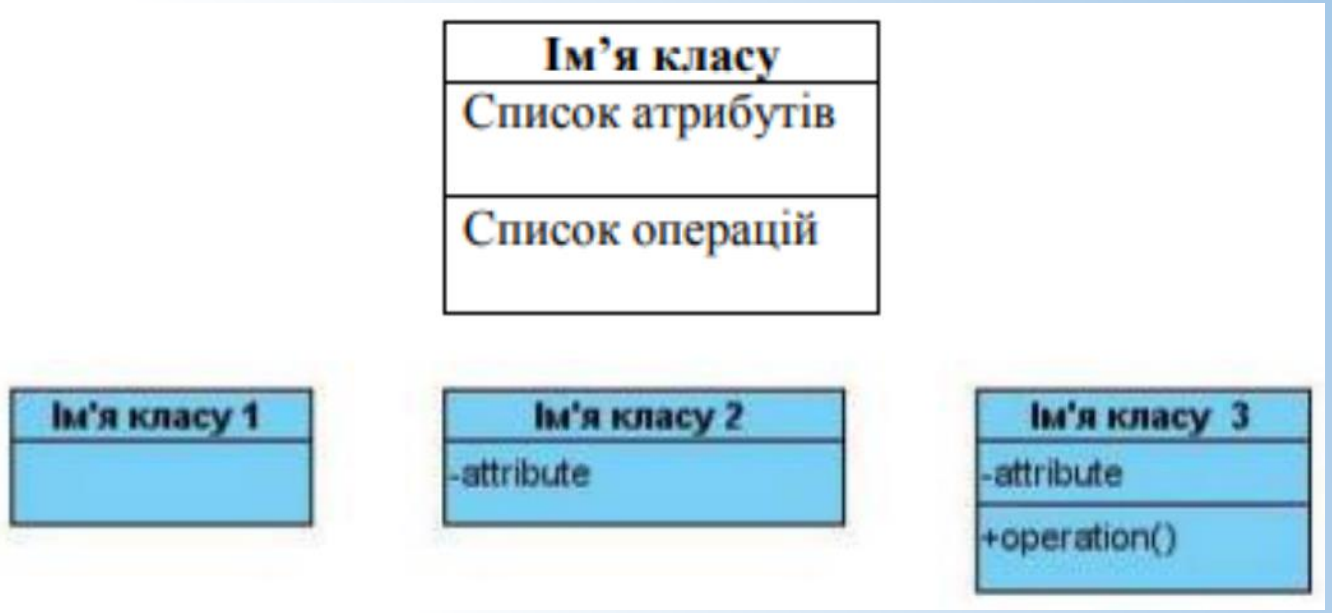


Рисунок 14.12 Узагальнений вигляд класу на діаграмі

Класи являють собою набір сутностей, в термінах яких робота системи повинна представлятися користувачам.

Узагальнення – зв'язок між двома елементами моделі, коли один елемент (підклас) є окремим випадком іншого елементу (суперкласу).

Агрегація – відношення між елементами моделі, коли один елемент є частиною іншого елементу (агрегату) та представляється стрілкою з ромбоподібним кінцем .

Рисунок 14.13

Атрибути класу або ***властивості*** записуються в другій зверху секції прямокутника класу.

<квантор видимості><ім'я атрибута>[кратність]:
<тип атрибута>=<початкове значення>{рядок-властивість}

Квантор видимості може приймати одне з трьох можливих значень і відображається за допомогою відповідних спеціальних символів:

- «+» позначає атрибут з областю видимості типу загальнодоступний (public);
- «#» позначає атрибут з областю видимості типу захищений (protected);
- «–» позначає атрибут з областю видимості типу закритий (private);

Квантор видимості для операції може бути опущений.

Операція (operation) – сервіс, що надається кожним екземпляром класу за певною вимогою.
Сукупність операцій характеризує функціональний аспект поведінки класу.

Відношення між класами

- залежності
- асоціації
- узагальнення

Рисунок 14.14

Рисунок 14.15 Приклад UML-діаграми класів

Рекомендації з побудови діаграми класів:

- при виділенні класів слід враховувати той факт, що вони є узагальненими (укрупненими) сутностями, які надалі підлягають уточненню і можливому розбиттю на кілька дрібніших класів;
- для виділення класів сутностей необхідно визначити всі реальні або уявні об’єкти, що мають істотне значення для розглядуваної предметної області, інформація про які підлягає зберіганню;
- для кожної дійової особи слід передбачити, як мінімум, один граничний клас з метою організації інтерфейсу між ним і системою;
- для управління, забезпечення взаємодії та координації роботи об’єктів, що реалізують одну з функцій системи (зазвичай, варіант використання), необхідно передбачити, як мінімум, один керуючий клас;
- рекомендується використовувати відношення агрегації, композиції і узагальнення;
- при розробці діаграми основна увага повинна бути приділена визначенню та деталізації класів сутностей, керуючих і граничних класів, що забезпечують взаємодію із зовнішніми системами.

Мета моделювання даних – забезпечення розробника ІС концептуальною схемою бази даних у формі однієї моделі або декількох локальних моделей, які відносно легко можуть бути відображені в будь-якій системі баз даних.

ERD безпосередньо використовуються для проєктування реляційних баз даних.

Функціональна модель системи дозволяє концептуально визначити набори даних, що використовутимуться в системі.

Для відображення організації даних в системі будується ***інформаційна модель***.

Процедура проєктування розбита на три етапи:

Етап 1-й. Концептуальне проєктування

Етап 2-й. Логічне проєктування

Етап 3-й. Фізичне проєктування

14.2 ОГЛЯД МЕТОДОЛОГІЇ IDEF1X

Метод IDEF1 – метод, що дозволяє будувати модель даних, еквівалентну реляційній моделі в третій нормальній формі (вдосконалена методологія IDEF1X).

IDEF1X (IDEF1 Extended) – методологія побудови реляційних структур (баз даних, БД), що відноситься до типу методологій «сутність-взаємозв'язок» (ER – Entity-Relationship) і, як правило, використовується для моделювання реляційних БД, що мають відношення до аналізованої системи (1981 р.).

Мета концептуального проєктування – створення концептуальної моделі даних на основі представлень про предметну область кожного окремого типу користувачів.

Концептуальна модель представляє собою опис основних сутностей (таблиць) і зв'язків між ними без урахування прийнятої моделі БД і синтаксису цільової СУБД.

Послідовність кроків при концептуальному проєктуванні.

Етап 1. Виділення сутностей.

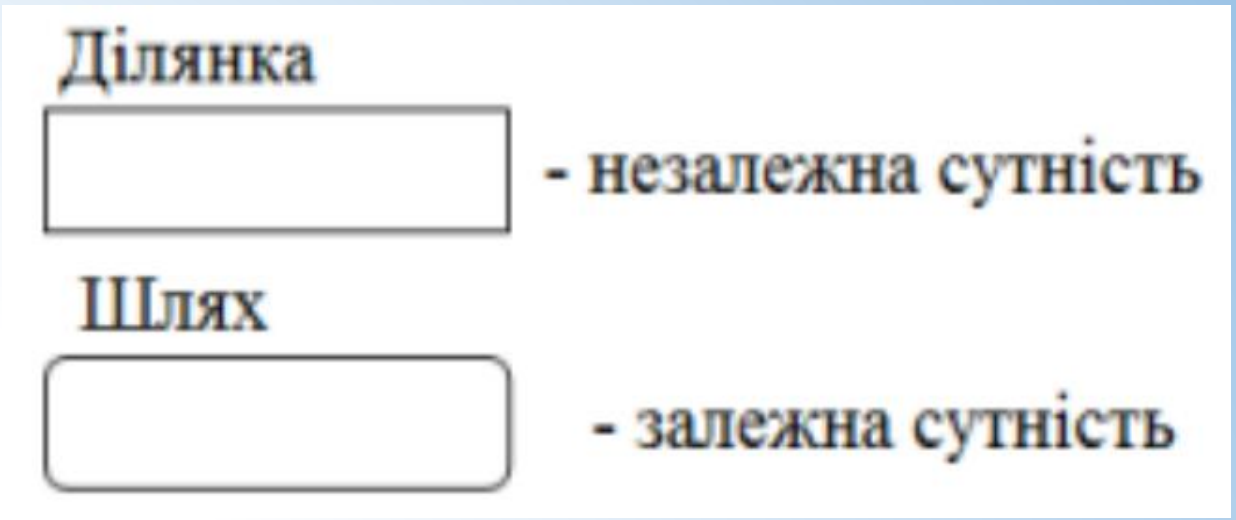


Рисунок 14.16 Відображення залежної та незалежної сутностей

Сутність в методології IDEF1X є ***незалежною***, якщо сутність не залежить від існування іншої сутності.

Сутність називається ***залежною***, якщо її існування залежить від існування інших сутностей.

Етап 2. Виділення атрибутів.

Етап 3. Задавання доменів атрибутів.

Завдання доменів визначає набір допустимих значень для атрибута (декількох атрибутів), а також тип, розмір і формат атрибута (атрибутів).

Етап 4. Визначення набору ключів.

Ключ – один або кілька атрибутів сутності, що служать для однозначної ідентифікації її примірників або для їх швидкого пошуку.

Типи ключів:

- суперключ (superkey);
- потенційний ключ (potential key);
- первинний ключ (primary key);
- альтернативні ключі (alternative key)

В нотації IDEF1X атрибути зображуються у вигляді списку імен усередині блоку сутності.

Рисунок 14.17 Незалежна сутність і приклад первинного та альтернативного ключів

Етап 5. Визначення зв'язків.

Зв'язок (Relationship) – засіб представлення відносин між сутностями.

Типи зв'язків між сутностями:

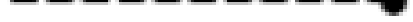
- Зв'язки типу «частина-ціле», які визначаються зазвичай дієсловами «складається з», «включає»
- Класифікаційні зв'язки
- Виробничі зв'язки (наприклад, «начальник-підлеглий»)
- Функціональні зв'язки, які визначаються зазвичай дієсловами «виробляє», «впливає», «залежить від», «обчислюється за»

Зв'язок характеризується наступним набором параметрів:

- ім'ям – вказується у вигляді дієслова і визначає семантику зв'язку;
- кратністю (потужністю) (один-до-одного (1:1); один-до-багатьох (1:N); багато-до-багатьох (N: M, N = M або N <> M);
- типом (ідентифікуючий, неідентифікуючий тип);
- обов'язковістю (обов'язковий, необов'язковий тип зв'язку);
- ступенем участі (кількістю сутностей, що беруть участь в зв'язку: унарний (рекурсивний), тернарний (пов'язує три сутності), кватернарний)

У методології IDEF1X ступінь участі може бути тільки унарною або бінарною.

Таблиця 14.1 – Типи зв'язків

Відображення	Тип обов'язковості зв'язку	Потужність зв'язку праворуч
	обов'язковий, ідентифікуючий	1
		0 .. ∞
 Z		0 або 1
 P		1 .. ∞
 <число>		<число>
	необов'язковий, неідентифікуючий	0 .. ∞
	обов'язковий, неідентифікуючий	0 .. ∞

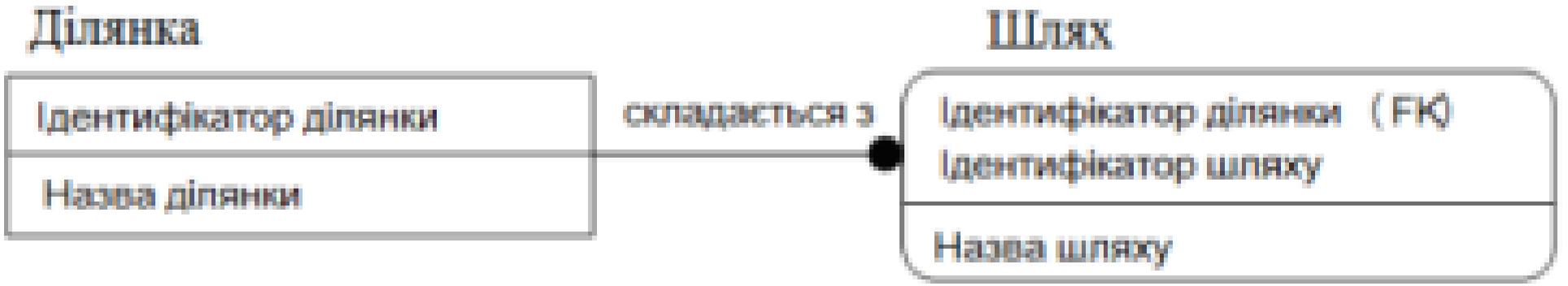


Рисунок 14.18 Ідентифікуючий зв'язок

Рисунок 14.19 Неідентифікуючий зв'язок

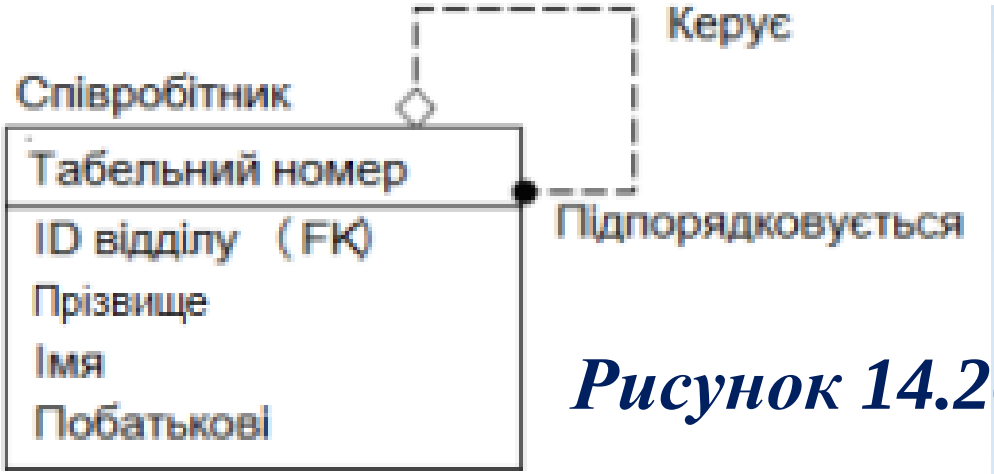


Рисунок 14.20 Рекурсивний зв'язок

Етап 6. Визначення суперкласів і підкласів.

Коли дві і більше сутностей за набором атрибутів незначно відрізняються одна від одної, то можна застосовувати в моделі конструкцію – ***ієрархію спадкування (категорій),*** що включає в себе суперклас і підкласи.

Суперклас – сутність, що включає в себе підкласи.

Ієрархія спадкування являє собою особливий тип об'єднання сутностей, які поділяють спільні характеристики.

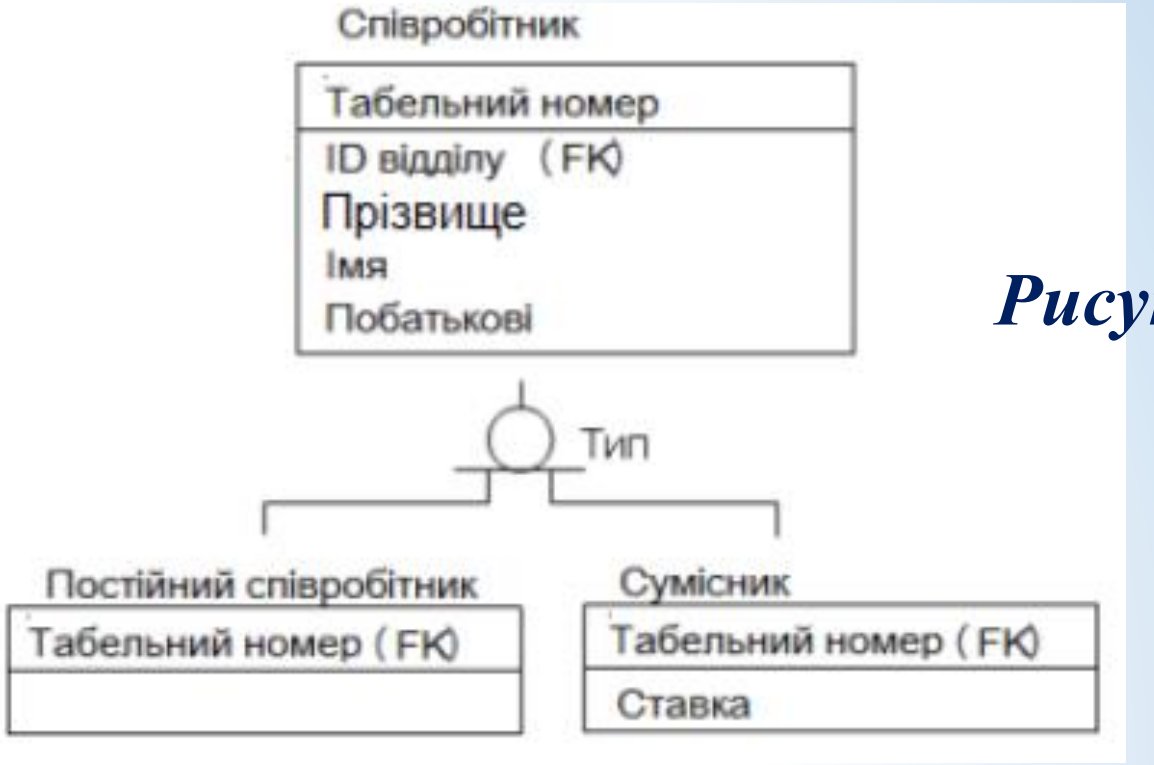


Рисунок 14.21 Ієрархія спадкування (неповна категорія)

Ієрархію спадкування створюють, коли кілька сутностей мають загальні за змістом атрибути або коли сутності мають загальні за змістом зв'язки.

Дискримінатор – атрибут родового предка, який показує, як відрізнити одну сутність від іншої.

Ієрархії категорій діляться на два типи: неповні (рис. 14.21) і повні (рис 14.22)

Рисунок 14.22 Ієрархія спадкування (повна категорія)

Логічне проєктування з використанням методології IDEF1X

Мета логічного проєктування – розвинути концептуальне представлення системи з урахуванням прийнятої моделі БД (ієрархічної, мережевої, реляційної).

Приклад 14.1 В якості моделі прийняти реляційну БД в третій нормальній формі (набір нормалізованих відносин з кратністю зв’язків 1: N).

Транзакція – одна дія або їх послідовність, виконуваних як єдине ціле одним або декількома користувачами (прикладними програмами) з метою здійснення доступу до БД і зміни її вмісту.

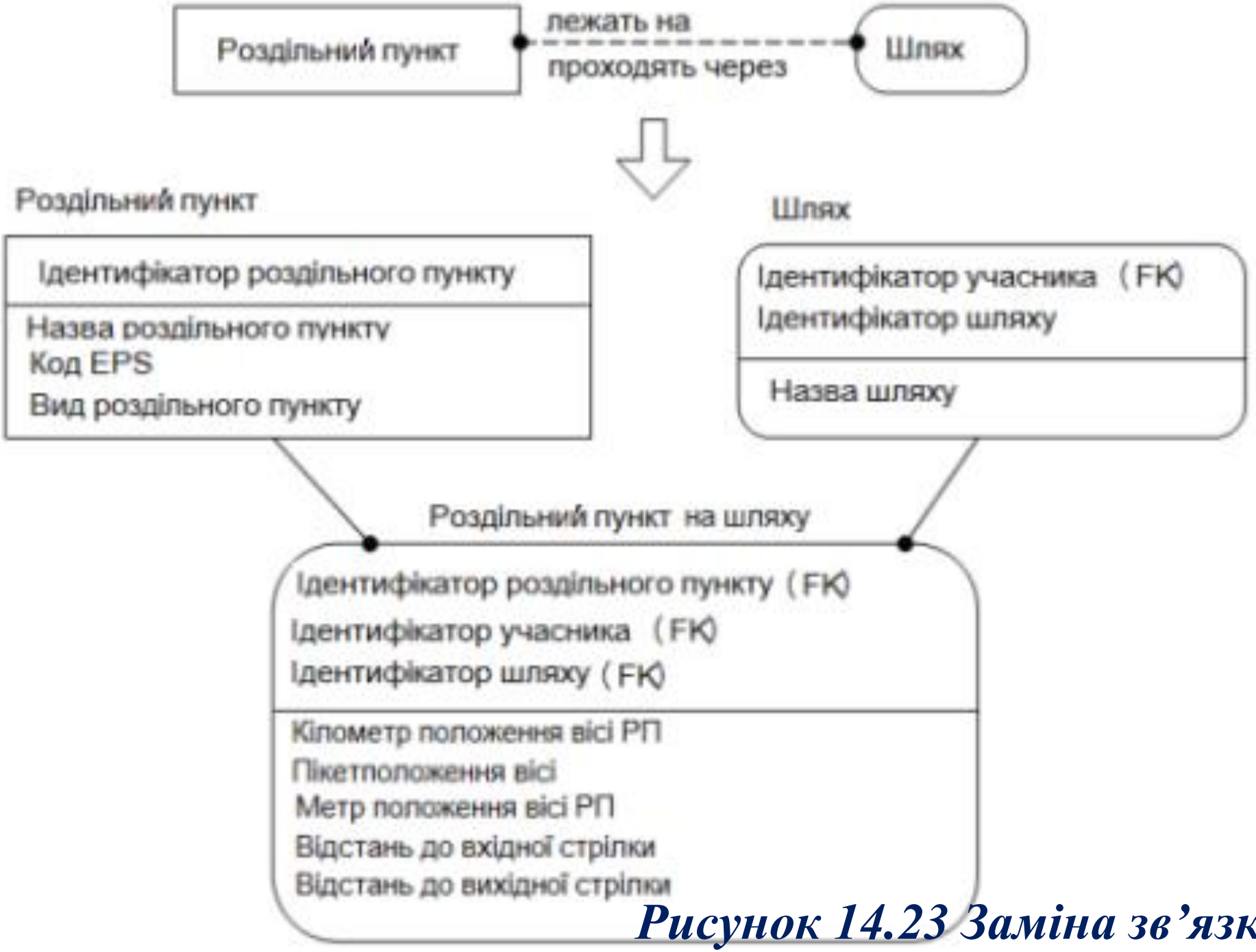
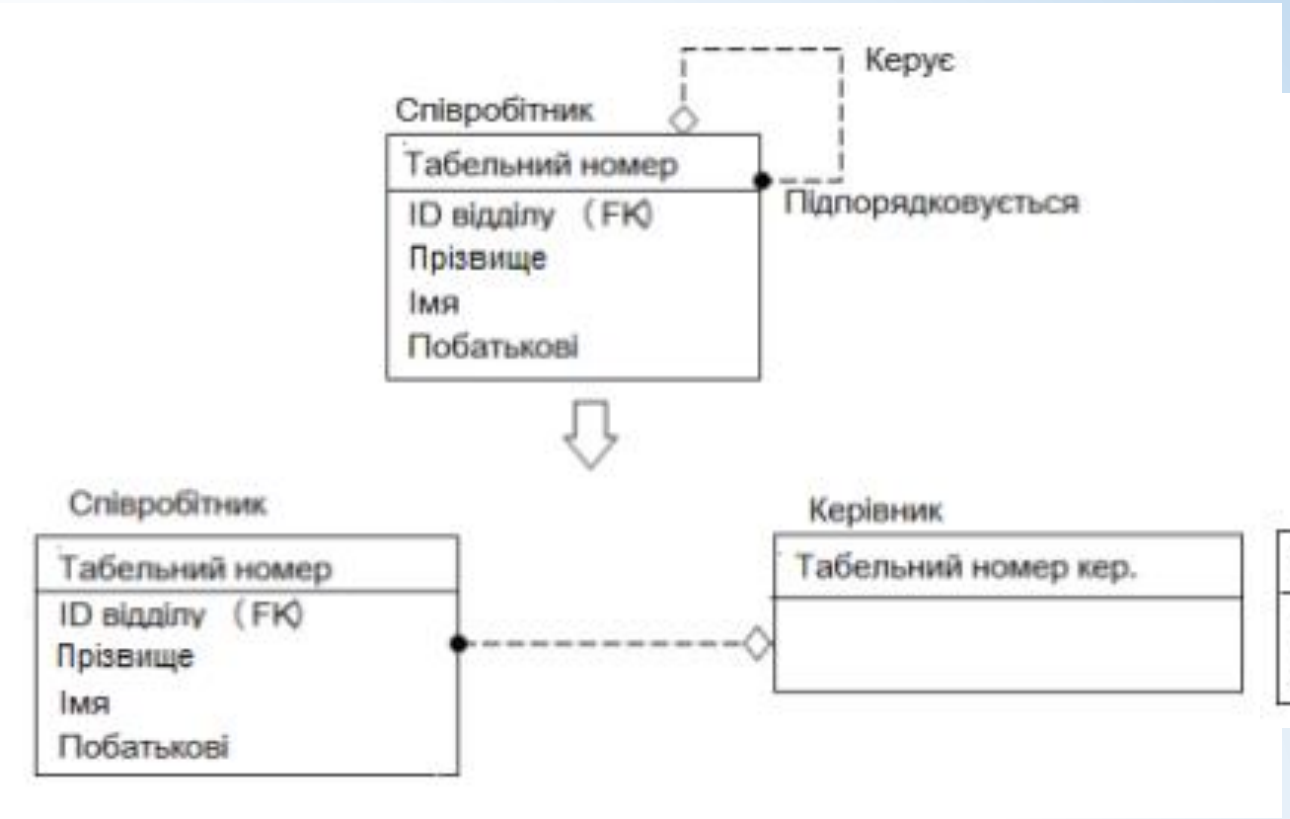


Рисунок 14.23 Заміна зв'язку N: M

Рекурсивний зв’язок замінюють, визначивши додаткову сутність і необхідну кількість зв’язків (рис. 14.24).

Видалення багатозначних атрибутів (рис. 14.25).



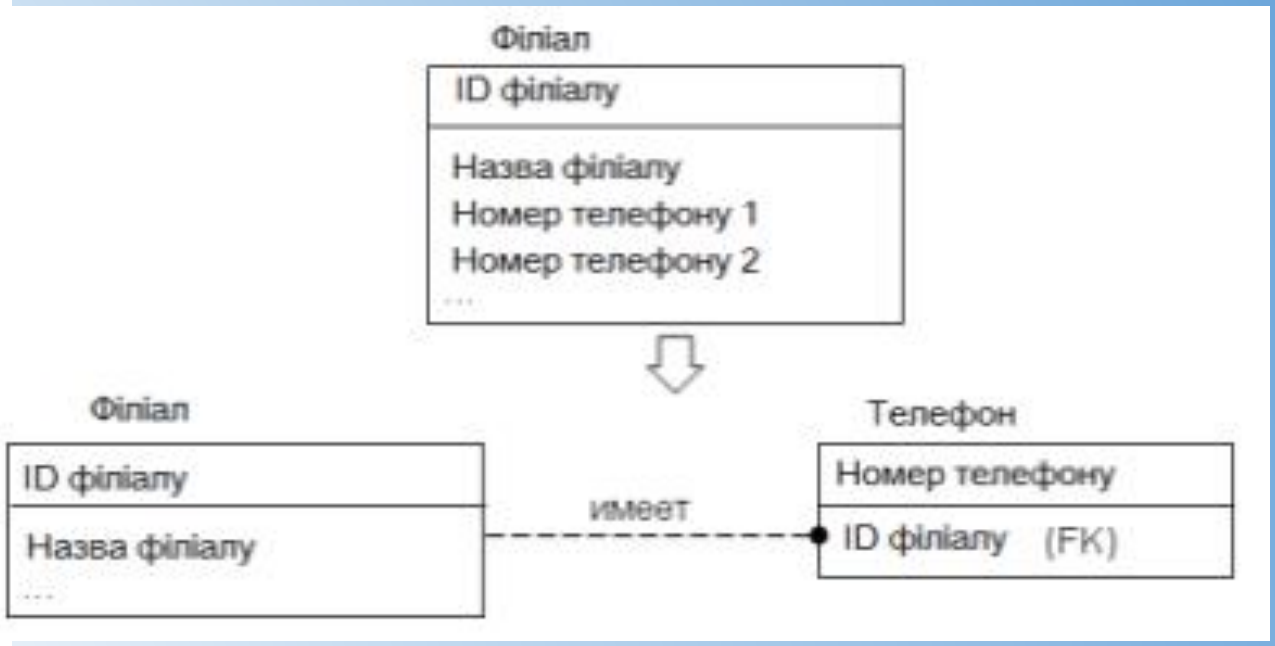


Рисунок 14.24 Заміна рекурсивного зв’язку

Рисунок 14.25 Видалення багатозначних атрибутів

Зв’язок є надлишковим, якщо одна і та ж інформація може бути отримана не тільки через нього, а й за допомогою іншого зв’язку (рис. 14.26).

У процесі визначення сутностей могли бути створені сутності, які насправді є однією сутністю (рис. 14.27). У цьому випадку їх слід об’єднати.

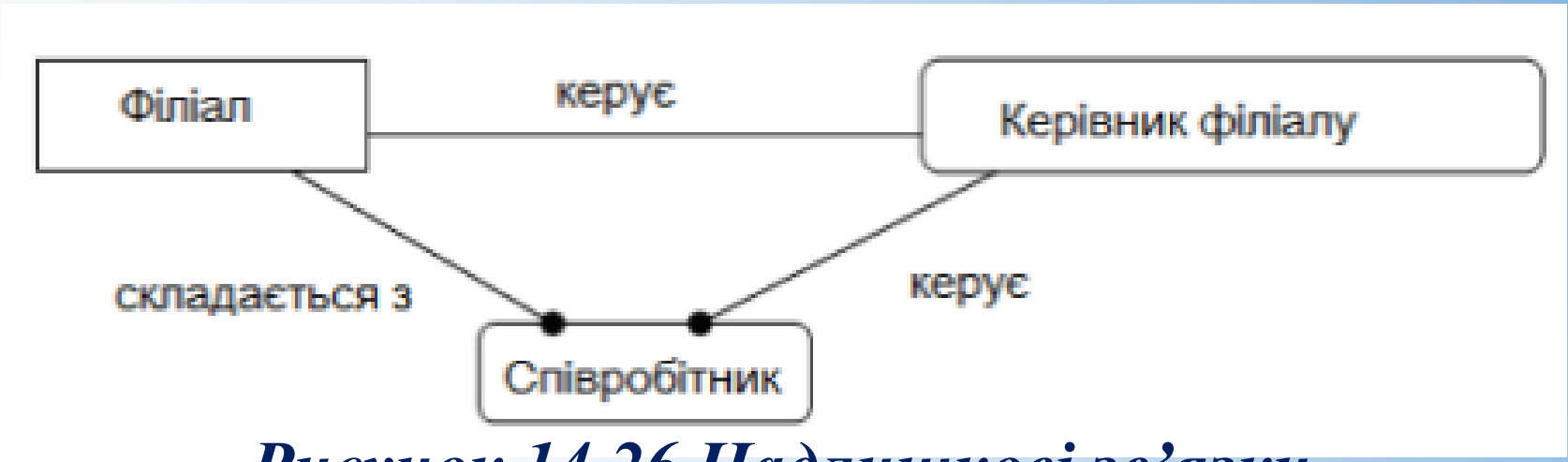


Рисунок 14.26 Надлишкові зв’язки

Рисунок 14.27 Зв’язки 1:1

Фізичне проектування з використанням методології IDEF1X

Мета фізичного проєктування – перетворення логічної моделі з урахуванням синтаксису, семантики і можливостей обраної цільової СУБД.

При реалізації проєкту часто для досягнення більшої ефективності системи потрібно знизити вимоги до рівня нормалізації відносин, тобто внести деяку надмірність даних. Процес внесення таких змін до БД називається ***денормалізацією***.

Основна ідея ***нормалізації*** полягає в тому, щоб кожен факт зберігався в одному місці, тобто щоб не було дублювання даних.

Проєктування реляційної БД являє собою покроковий процес створення набору відносин (таблиць, сутностей), в яких відсутні небажані функціональні залежності.

Функціональна залежність:

Нехай A і B – довільні набори атрибутів відносини. Тоді B функціонально залежить від A (A→B), в тому і тільки в тому випадку, якщо кожному значенню A відповідає в точності одне значення B. Ліва частина функціональної залежності (A) називається ***детермінантою***, а права (B) ***– залежною частиною***.

A може бути первинним ключем, а B – набором неключових атрибутів, так як одному значенню первинного ключа в точності відповідає одне значення набору неключових атрибутів. Якщо в БД відсутні небажані функціональні залежності, то це забезпечує мінімальну надмірність даних, що в свою чергу веде до зменшення обсягу пам'яті, необхідної для зберігання даних. Процес усунення таких залежностей отримав назву ***нормалізація***.

Процес нормалізації вперше був запропонований Е.Ф. Коддом в ***1972 р.*** Спочатку було запропоновано три види нормальних форм (1NF, 2NF і 3NF).

Потім Р. Бойсом і Е.Ф. Коддом (***1974 р.***) було сформульовано більш суворе визначення третьої нормальної форми, яке отримало назву нормальна форма Бойса-Кодда (BCNF).

Слідом за BCNF з'явилися визначення четвертої (4NF) і п'ятої (5NF або PJNF) нормальних форм (Р. Фагин***, 1977 р. і 1979 р.).***

1NF. Відношення знаходиться в 1NF, якщо на перетині кожного стовпця і рядка знаходяться тільки елементарні (атомарні, неподільні) значення атрибутів.

2NF. Відношення знаходиться в 2NF, якщо воно знаходиться в 1NF, і кожен неключовий атрибут характеризується повною функціональною залежністю від первинного ключа (рис. 14.28).

Рисунок 14.28 Забезпечення повної функціональної залежності

3NF. Відношення знаходиться в 3NF, якщо воно знаходиться у 2NF і ніякий неключовий атрибут функціонально не залежить від іншого неключового атрибута, тобто немає транзитивних залежностей.

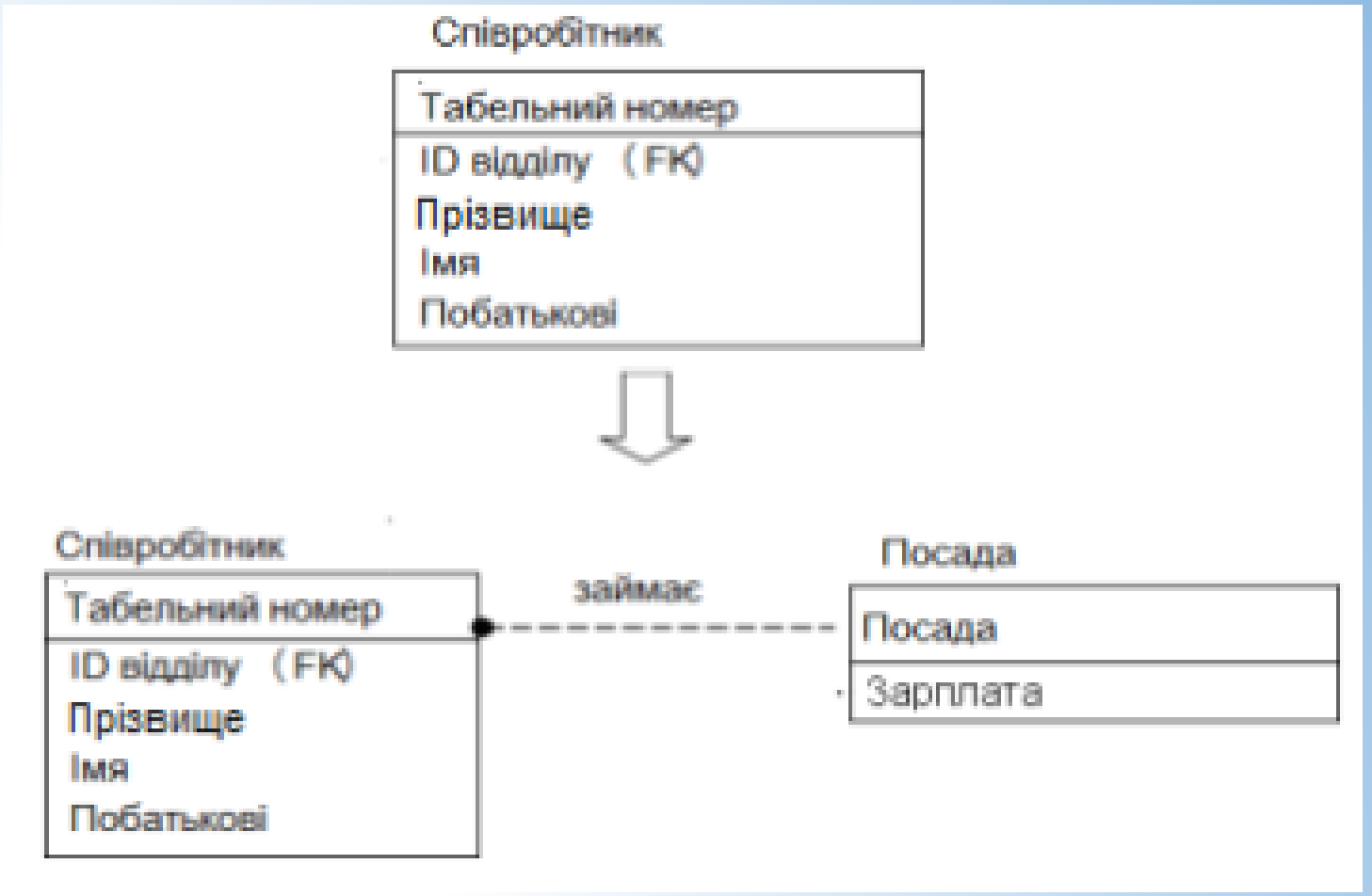


Рисунок 14.29 Усунення транзитивної залежності

BCNF. Відношення знаходиться в BCNF, якщо воно знаходиться в 3NF і кожен детермінант відносини є його можливим ключем. Відступ від BCNF можливий, коли в таблиці є кілька складових потенційних ключів (детермінантів), причому частина атрибутів одного потенційного ключа характеризується повною функціональною залежністю від частини іншого потенційного ключа.

Рисунок 14.30 Усунення транзитивної

4NF. Відношення знаходиться в 4NF в тому і тільки в тому випадку, якщо в ньому відсутні нетривіальні багатозначні залежності. ***Нетривіальна багатозначна залежність:*** у відношенні з атрибутами A, B і C існує нетривіальна багатозначна залежність, якщо для кожного значення атрибута A є набір значень атрибута B (A - >> B) і набір значень атрибута C (A - >> C), але між атрибутами B і C немає залежностей.

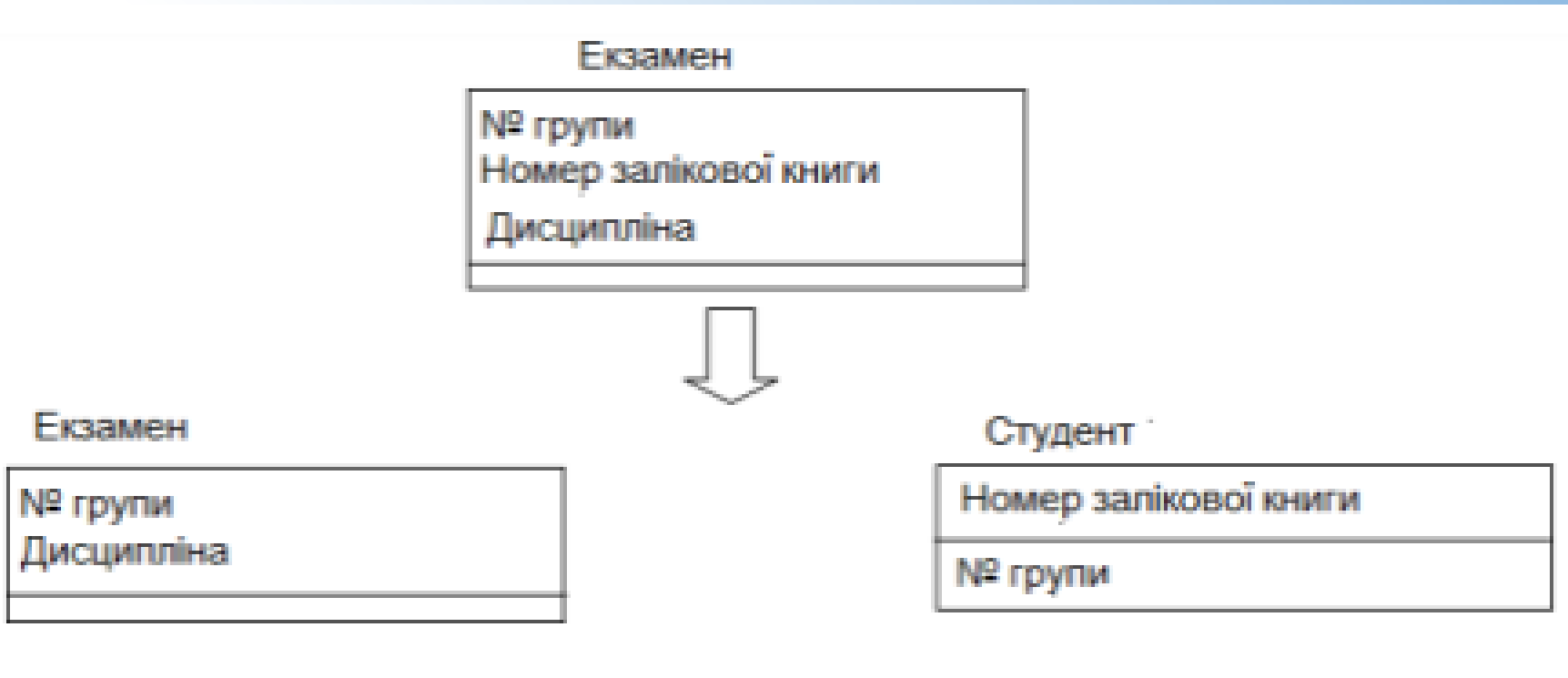


Рисунок 14.31 Приведення відношення до 4NF

5NF (нормальна форма проєкції з'єднання, PJNF). Відношення знаходиться в 5NF, якщо в ньому немає залежностей з'єднання. ***Залежність з'єднання:*** у відношенні з атрибутами A, B і C існує залежність з'єднання, якщо для кожного значення атрибута A є набір значень атрибута B (A - >> B) і набір значень атрибута C (A - >> C), а також існує багатозначна залежність між атрибутами B і C (B - >> C або C - >> B).

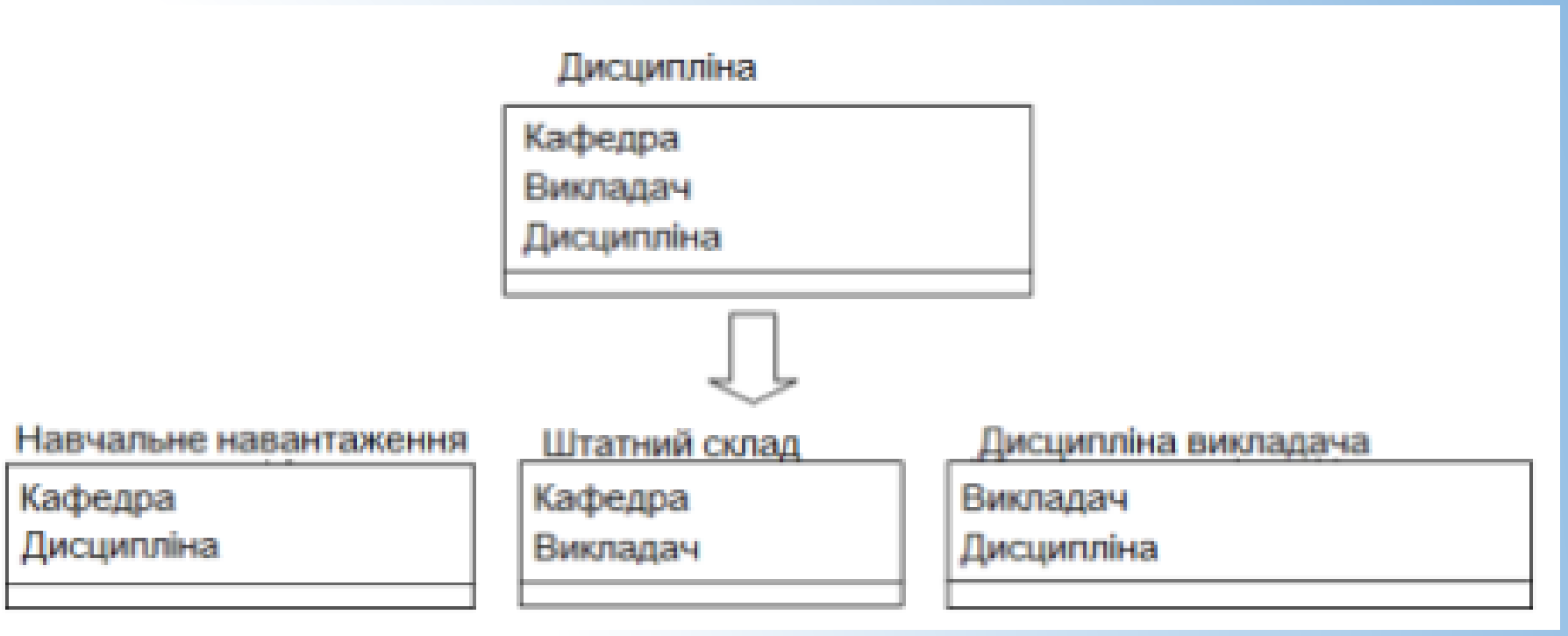


Рисунок 14.32 Приведення відношення до 5NF

Перевірка виконання транзакцій.

Найбільш частими операціями по роботі з БД є введення, видалення і модифікація записів, а також вибірка даних.

На основі поточної ERD необхідно перевірити здійсненність і коректність виконання всіх необхідних операцій по роботі з БД.

Визначення вимог підтримки цілісності даних.

Обмеження цілісності даних є обмеження, які вводяться з метою запобігання розміщення в базі суперечливих даних, до яких відносять:

Обов'язкові дані – атрибути, які завжди повинні містити одне з допустимих значень (NOT NULL).

Домени – набори допустимих значень для атрибута.

Бізнес-правила (бізнес-обмеження) – обмеження, прийняті в даній предметній області.

Посилальна цілісність – набір обмежень, що визначають дії при вставці, оновленні та видаленні записів.

Тригер – це збережена в БД процедура, що викликається автоматично при виконанні видалення (DELETE), вставки (INSERT) або відновлення (UPDATE) запису.

ТЕМА №15. ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

15.1 Загальні поняття та характеристика інтерфейсу
15.2 Визначення принципів формування та особливостей проєктування інтерфейсу
15.3 Загальні вимоги до інтерфейсів інформаційних систем
15.4 Визначення взаємодії між користувачем і комп'ютером
15.5 Розміщення інформації на екрані
15.6 Тональність діалогу та термінологія
15.7 Використання кольорів, мигання і клавіатури
15.8 Запобігання, виявлення і виправлення помилок

15.1 ЗАГАЛЬНІ ПОНЯТТЯ ТА ХАРАКТЕРИСТИКА ІНТЕРФЕЙСУ

Інтерфейс – це сукупність засобів і правил, які забезпечують взаємодію між користувачем, ЕОМ і програмами.

Мета створення ергономічного інтерфейсу: відобразити інформацію настільки ефективно, наскільки це можливо для людського сприйняття, і структурувати відображення на дисплеї так, щоб притягнути увагу до найбільш важливих одиниць інформації та мінімізувати загальну інформацію на екрані та представити тільки те, що є необхідним для користувача.

При розробці інтерфейсу необхідно пам'ятати, що:

- інтерфейс – сама важлива частина програмної системи;
- інтерфейс впливає на характер рішень, які приймає ОПР;
- який саме конкретний тип інтерфейсу можна створити за допомогою вибраних інструментальних засобів і які принципові можливості може надати інструментальна система.

Обов'язкові операції, що здійснюються за допомогою інтерфейсу:

- вхід, вихід ІС;
- доступ до функцій меню ІС;
- ввід даних в ІС;
- одержання результату у заданій формі представлення;
- виведення підказок, коментарів команд;
- аналіз ситуацій;
- повідомлення про помилки;
- вивід застережень;
- запис історії роботи з ІС;
- збереження результатів.

Характеристики інтерфейсу користувача та принципи його формування

Адаптивний інтерфейс (АІ) – сукупність програмних та технічних засобів, які дозволяють користувачу ефективно використовувати всі можливості, які надає система, та задаються конкретні налагодження для кожного користувача.

Головний критерій ефективності функціонування програмних продуктів: максимальне використання для роботи з ними людських ресурсів.

Адаптивний інтерфейс ***повинен враховувати*** апріорну інформацію про психофізичні, професійні, особисті характеристики користувача.

Додаткова інформація про користувача ***повинна бути отримана*** з аналізу дій користувача безпосередньо в процесі роботи.

Адаптивний інтерфейс ***повинен забезпечувати*** користувачу полегшений режим взаємодії.

Адаптивна система ***повинна комбінувати*** в собі особливості адаптивних і адаптованих компонентів.

Адаптована система – система, яка дозволяє користувачу змінювати певні системні параметри і відповідно змінювати їх поведінку.

Адаптивні системи – системи, які адаптуються до користувача автоматично, ґрунтуючись на припущеннях системи.

Можливість адаптації системи до рівня професіоналізму користувача здійснюється за рахунок здатності сприймати і виконувати запити на внутрішній формальній мові, що забезпечує більш швидкий доступ до інформації.

Природно-мовний інтерфейс – переводить запити, що надходять природною мовою, у формальне представлення, звертається з ним до бази даних, і представляє результат, використовуючи алгоритми і технології, реалізовані в ПС, у вигляді, придатному для перегляду й аналізу.

Користувачі ПМ-інтерфейсів – співробітники підприємства, клієнти та усі, кому потрібна можливість швидкого і прямого доступу до актуальної інформації.

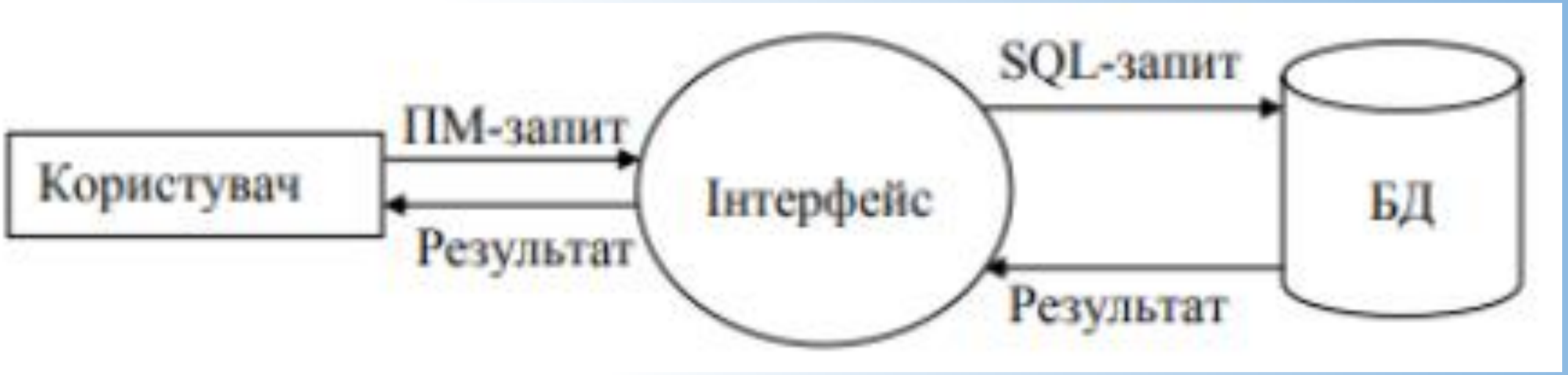


Рисунок 15.1 Загальна схема потоків даних ПМ-інтерфейсу

Використання ПМ-інтерфейсу надає можливість значно скоротити витрати організації на різних рівнях:

- можливість формулювати потреби в інформації найбільш простим і разом з тим самим доступним чином;

- розвантажує фахівців від рутини створення форм і звітів по кожному розрізі інформаційного простору в базі даних та може бути корисний для розробки традиційних додатків на основі баз даних, оскільки рятує від ручного етапу побудови складних sql-запитів;

- при розробці ПМ-інтерфейсів використовуються найсучасніші методи інженерії знань, що дозволяє використовувати технологію розуміння на більш високому рівні абстракції даних.

Побудова природно-мовного інтерфейсу до бази даних і до інформаційної системи надає можливість:

- мати прямий доступ до будь-яких аналітичних розрізів інформації, що зберігається в базі;
- швидко отримувати необхідну інформацію, що потрібна у поточний момент;
- зосереджуватися на тому, що необхідно одержати з бази, а не на тому, як це зробити;
- вчасно контролювати правильність занесення інформації в базу;
- доповнити інформаційну систему інтелектуальною технологією.

Вимоги до ПМ-інтерфейсів повинні визначати зміст і форми інформації, що надається, а також регламент взаємодії користувача і системи.

Базовою формою опису технологічних і бізнес-процесів повинна бути ***схема процесу.***

Сервісними формами представлення інформації можуть бути тексти, таблиці, графіки, гістограми, а також абстрактні образи.

Програмне забезпечення інтерфейсного модуля повинне забезпечити такі можливості:

- введення запитів до бази знань на внутрішній формальній мові системи;
- редагування запитів користувача;
- виконання запитів до бази експертних знань і вивід результатів на екран комп'ютера в зручному для користувача вигляді;
- перегляд проміжних результатів роботи;
- поповнення і коректування бази експертних знань у режимі діалогу;
- верифікацію відповідей на запити.

Основні етапи діяльності користувача в системі, як особи, що приймає рішення:

- сприйняття інформації
- оцінювання інформації, її аналіз і узагальнення;
- прийняття управлінських рішень.

15.2 ВИЗНАЧЕННЯ ПРИНЦИПІВ ФОРМУВАННЯ ТА ОСОБЛИВОСТЕЙ ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ

Загальні принципи побудови адаптивних інтерфейсів:

- принцип відповідності призначення і структури інтерфейсу поставленим цілям і задачам;
- принцип мінімізації витрат ресурсів користувача;
- принцип максимального взаєморозуміння та непротиріччя;
- принцип незбитковості;
- принцип безпосереднього доступу до системи підказок;
- принцип гнучкості;
- принцип максимальної концентрації користувача на задачі, що розв’язується і локалізації повідомлень про помилки;
- принцип врахування професійних навичок конкретного користувача;
- принцип легкості користування і простоти навчання;
- принцип надійності

Особливості проєктування інтерфейсу на принципах людського фактору

Предмет вивчення «людського фактору» – це дослідження факторів, явищ, подій, які впливають на продуктивність праці, ефективність та якість життя всіх тих, хто працює, а також проєктування машин та обладнання, що використовується в процесі виконання роботи.

Людський фактор – це метод проєктування, дослідження та пояснювання інтерфейсу в широкому смислі між людиною та машиною або знаряддями, які створюються людиною.

Користувачів комп'ютерних систем можна розділити на фахівців та тимчасових користувачів.

Загальні принципи людського фактору:

- користувач повинен завжди знати, що робити далі, тобто, система повинна давати йому інструкції;

Декілька ситуацій, які вимагають зворотного зв'язку від системи:

- ✓ користувач повинен знати, що система чекає від нього;
- ✓ користувач повинен знати, що дані введено коректно;
- ✓ користувач повинен знати, що дані не були введені коректно
- ✓ якщо має місце затримка з обчисленнями, то користувач повинен про це знати;
- ✓ користувач повинен знати, що система завершила (не завершила) виконання завдання

- форматування екрану монітора необхідно робити таким чином, щоб різні типи інформації, команди, повідомлення завжди з’являлись в одній і тій же області;

Екран можна розділити на такі зони:

- ✓ вікно з назвою (титулом) сторінки на екрані;
- ✓ вікно з прапорцями;
- ✓ вікно повідомлень;
- ✓ вікно виходу;
- ✓ операційне вікно

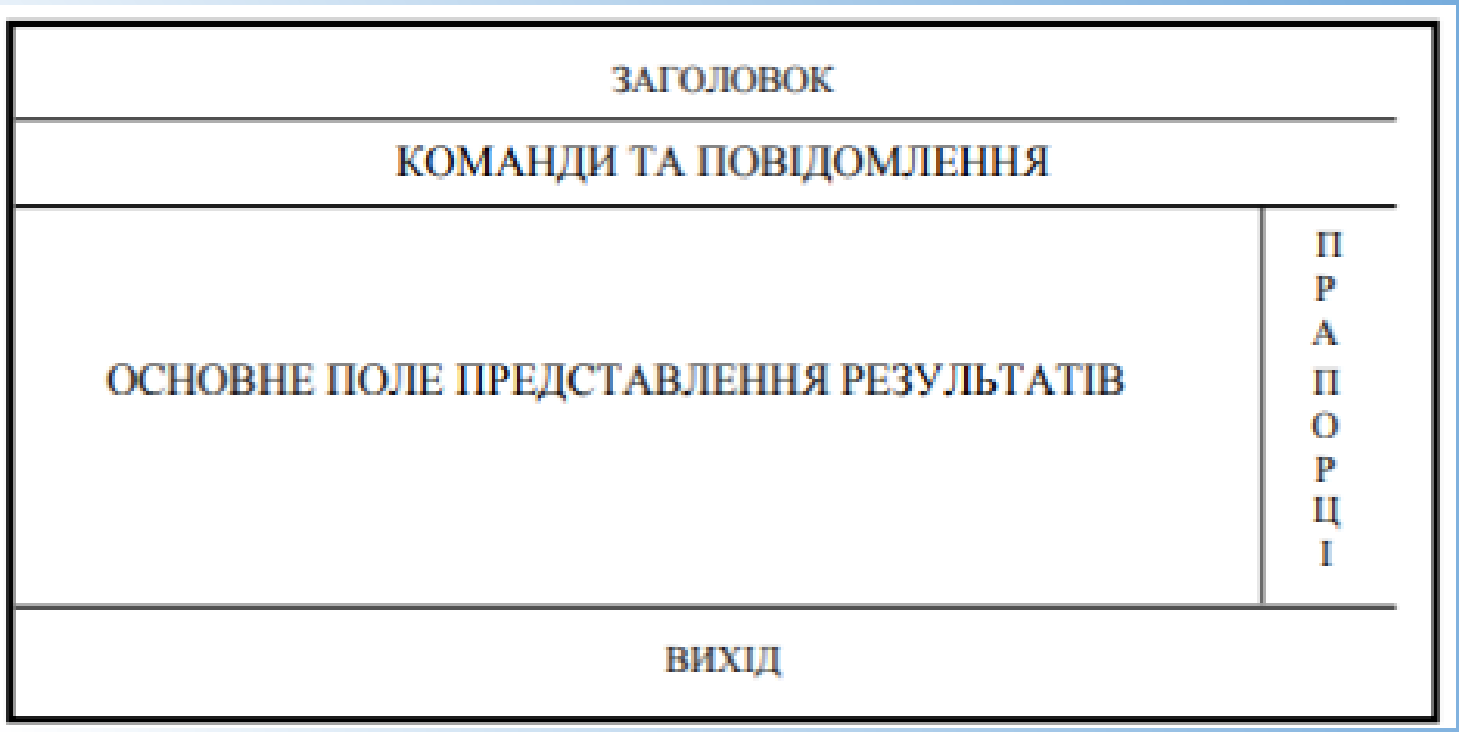


Рисунок 15.2 Приклад розподілу екрану на зони

- при проєктуванні функцій пейджингу чи скролінгу у межах операційного вікна діалог повинен обмежуватись однією ідеєю на фрейм;

- повідомлення, команди або інформація щодо результатів повинні утримуватись на екрані достатньо довго, для того, щоб користувач міг прочитати і сприйняти їх;

- економне використання ресурсів монітора;

- спрощувати складні функції і скорочувати об’єм введення даних з клавіатури за рахунок використання функціональних клавіш

Функціональним клавішам можуть бути присвоєні, наприклад, такі функції:

- ✓ START програми або функції;
- ✓ ДОПОМОГА (HELP);
- ✓ ВИХІД (EXIT) або ЗАВЕРШЕННЯ (TERMINATE) поточного сеансу діалогу;
- ✓ переривання поточної операції (ESCAPE);
- ✓ використання комбінацій клавіш

- ➢ створювати специфікації значень або повідомлень «за замовчуванням», які спрощують введення даних користувач системи в процесі діалогу.

- ➢ передбачати помилки, які можуть мати місце з боку користувача системи.

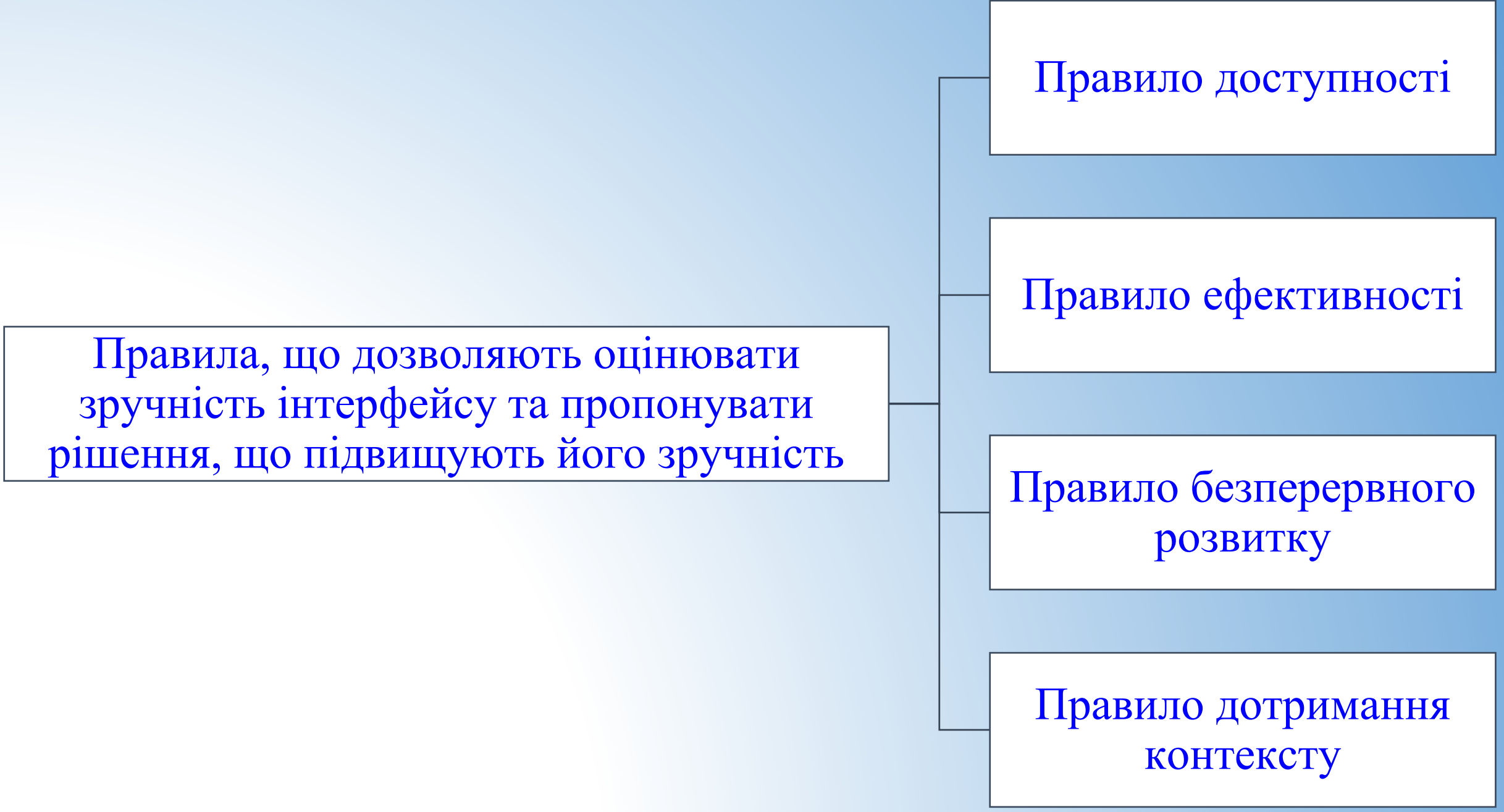

Правила, що дозволяють оцінювати зручність інтерфейсу та пропонувати рішення, що підвищують його зручність
Правило доступності
Правило ефективності
Правило безперервного розвитку
Правило дотримання контексту

Рисунок 15.3


Принципи розробки інтерфейсу користувача:
Принцип структуризації
Принцип простоти
Принцип видимості
Принцип зворотного зв’язку
Принцип толерантності
Принцип повторного використання

Рисунок 15.4

15.3 ЗАГАЛЬНІ ВИМОГИ ДО ІНТЕРФЕЙСІВ ІНФОРМАЦІЙНИХ СИСТЕМ

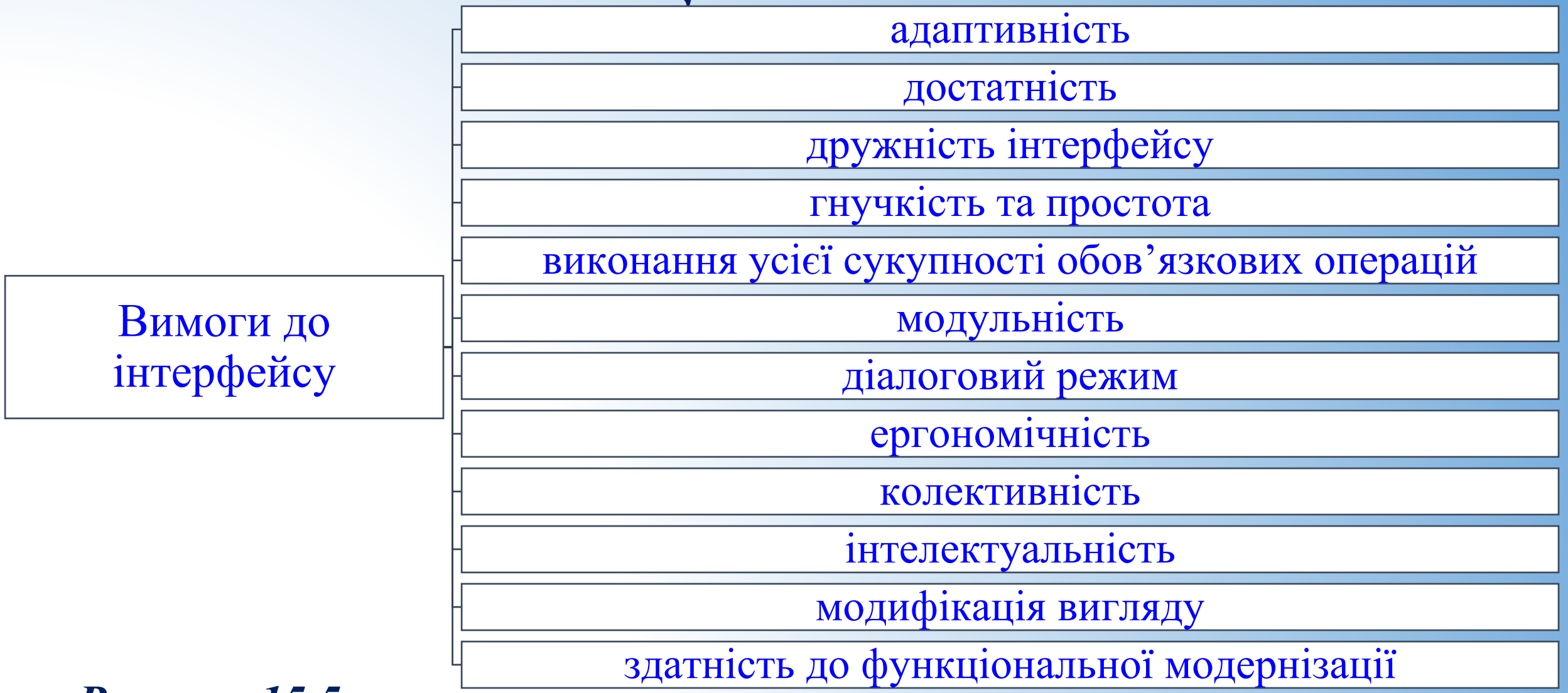


Рисунок 15.5

15.4 ВИЗНАЧЕННЯ ВЗАЄМОДІЇ МІЖ КОРИСТУВАЧЕМ І КОМП'ЮТЕРОМ

Рисунок 15.6

15.5 РОЗМІЩЕННЯ ІНФОРМАЦІЇ НА ЕКРАНІ

Екранна щільність – кількість інформації, що відображається на екрані.

Закономірність – чим менше екранна щільність, тим інформація, що відображається, доступніша та зрозуміліша для користувача.

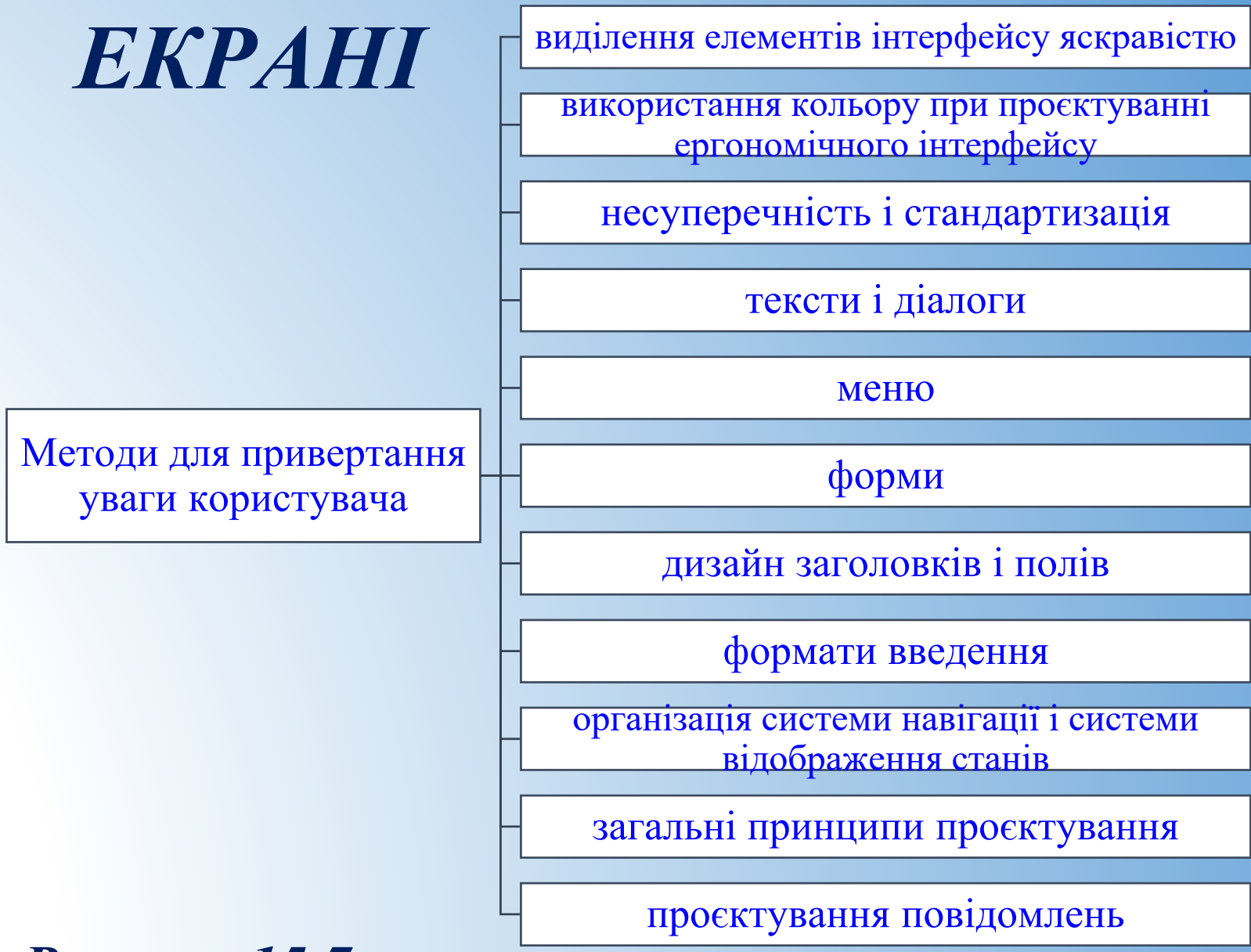


Рисунок 15.7

15.6 ТОНАЛЬНІСТЬ ДІАЛОГУ ТА ТЕРМІНОЛОГІЯ

Діалог повинен бути дружнім по відношенню до користувача.

Тональність діалогу повинна ґрунтуватись на таких загальних рекомендаціях:

- використовувати прості, граматично правильні речення
- не старатися жартувати або вживати гострі фрази;
- не вживати фрази, які звучать як повчання для користувача

Термінологія, яку фахівці з проєктування інформаційних систем рекомендують відносно наступних правил:

- не використовати комп’ютерний жаргон;
- уникати абревіатур;
- уникати символів, які можуть бути невідомими для користувача;
- використовувати просту і зрозумілу термінологію;
- використовувати одні й ті ж терміни для виконання однакових операцій;
- команди і пояснення користувачу повинні бути продумані і коректно сформульовані з правильним вживанням дієслів.

15.7 ВИКОРИСТАННЯ КОЛЬОРІВ, МИГАННЯ І КЛАВІАТУРИ

Кольори використовують для таких цілей:

- «піднімання» конкретних повідомлень;
- для зображення кластерів даних;
- виділення цифрових даних, графіків або областей екрану.

Сучасні монітори забезпечують можливості використання досить широкого набору відео- та аудіо-атрибутів:

- подвійна яскравість вибраних полів екрану або окремих повідомлень;
- мигання вибраних полів або повідомлень; –
- приховування відображення вибраних полів;
- інверсне відображення вибраних полів, повідомлень або областей екрану.

Клавіші можна використати для ініціалізації виконання деяких загальних операцій, які повторюються в процесі взаємодії з СППР.

В деяких стандартах є обов’язковим використання клавіші F1 для реалізації функції отримання допомоги (HELP).

Одні й ті ж функціональні клавіші повинні завжди використовуватись для однієї мети.

Поєднання вказаних атрибутів дозволяє суттєво підвищити швидкість сприйняття та глибину розуміння результатів роботи СППР, прискорити процеси вводу/виводу даних.

15.8 ЗАПОБІГАННЯ, ВИЯВЛЕННЯ І ВИПРАВЛЕННЯ ПОМИЛОК

Помилки користувача можуть бути засновані на неправильному розумінні дії або порядку дій або бути випадковими, неумисними.

Помилки другого виду:

- неточність у виборі опції;
- втрата активності;
- помилка режиму або стану

Техніка захисту включає такі аспекти:

- примусові дії в системі, які запобігають або утрудняють появу помилок;
- забезпечення хороших і інформативних повідомлень про помилки;
- забезпечення нормальної діагностики системи

Основні принципи обробки помилок у формах введення:

- забезпечення можливості посимвольного редагування введених записів для виправлення помилок введення;
- якщо помилка виявлена системою, бажано повернути курсор в поле з помилковими даними і яким-небудь чином виділити це поле;
- виводити значимі повідомлення про помилки, що використовують стиль мови користувача і відповідну термінологію;
- виводити повідомлення про помилки, які пояснюють і пропонують шляхи їх усунення.

ТЕМА №16. РЕІНЖИНІРИНГ ІНФОРМАЦІЙНИХ СИСТЕМ

Тематичний план

16.1 Сутність, призначення, сумісні поняття реінжинірингу ІС

16.2 Основний зміст діяльності з реінжинірингу ІС, місця реінжинірингу в ЖЦ ІС

16.1 СУТНІСТЬ, ПРИЗНАЧЕННЯ, СУМІСНІ ПОНЯТТЯ РЕІНЖИНІРИНГУ ІС

«Реінжиніринг» - планування та проведення операцій перепроєктування та внесення змін до окремих складових функціонально-організаційної структури системи та регламенту функціонування.

Реінжиніринг інформаційної системи (РІС) ***передбачає*** аналіз і перепроєктування ІС з метою реалізації її в новій якості та якісне поліпшення існуючих інформаційних мереж шляхом їх перепроєктування, перегляду процесів їх функціонування.

Мета РІС: встановлення істотного поліпшення якості ІС «в рази»

Особливості РІС:

- реінжиніринг ІС не є концепцією сталого поліпшення;
- першочергове завдання – створити відносно стійку структуру чітко визначеної якості з уже наявної, і, звичайно, з урахуванням можливості майбутніх перетворень

Модернізація ІС – відносно невелике поліпшення ІС, виправлення критичних помилок, при відсутності кардинальних змін системи.

Рефакторинг ІС – повне або часткове перетворення внутрішньої структури ПЗ ІС при збереженні зовнішньої поведінки; переклад на більш сучасну мову програмування.

Редизайн ІС – переробка тільки користувальницьких інтерфейсів ІС без істотного втручання в її пристрій.

Реверс-інжиніринг ІС – дослідження, відновлення (побудова) структурних моделей ІС.

Реінжиніринг бізнес-процесів (РБП) – фундаментальне переосмислення та радикальне перепроєктування бізнес-процесів для досягнення істотних поліпшень в ключових для сучасного бізнесу показниках результативності.

Реінжиніринг ІС – систематична трансформація існуючої системи з метою поліпшення її характеристик якості, підтримуваної нею функціональності, зниження вартості її супроводу, імовірності виникнення значимих для замовника ризиків, зменшення строків робіт із супроводу системи, а також підходи, методи та технології міграції, модернізації, еволюції ІС необхідно вважати частиною методологічного й інструментально-технологічного забезпечення процесу реінжинірингу ІС.

- ***Прямий інжиніринг (forward engineering):*** традиційний процес переходу від високорівневих абстракцій і логічного, незалежного від реалізації проєктування до фізичної реалізації системи.

- ***Редокументування (redocumentation):*** форма реструктуризації, процес аналізу системи з метою створення різного роду супроводжуючої її документації.

- ***Рефакторинг (refactoring):*** спеціальний вид реструктуризації – реструктуризації на рівні програмного коду, та є процесом зміни програмної системи, направленим на поліпшення внутрішньої структури програмного коду

- ***Реструктуризація (restructuring):*** трансформація системи з однієї форми представлення в іншу на одному і тому ж рівні абстракції

- ***Переорієнтація (retargeting):*** процес трансформації і перекладу (перенесення) існуючої системи в нову конфігурацію

- ***Зворотний інжиніринг (зворотне проєктування, reverse engineering):*** процес аналізу початкової системи.

- ***Супровід програмних продуктів (software maintenance):*** модифікація програмного продукту після його постачання з метою виправлення помилок, поліпшення продуктивності і інших атрибутів якості, або адаптації продукту до змін оточення;

- ***Трансляція вихідного коду (source code translation):*** трансляція початкового коду з однієї мови програмування на іншу або з однієї версії в іншу на тій же самій мові програмування;

- ***Інжиніринг систем (systems engineering):*** високорівневий процес інжинірингу систем, направлений на досягнення відповідності системи всім вимогам, що висуваються до неї.

16.2 ОСНОВНИЙ ЗМІСТ ДІЯЛЬНОСТІ З РЕІНЖИНІРИНГУ ІС, МІСЦЯ РЕІНЖИНІРИНГУ В ЖЦ ІС

Реінжиніринг характеризується діяльністю, як із супроводу, так і з розробки ІС.

У контексті досліджень, пов'язаних з еволюцією ІС, виділяються діяльності з ***супроводу, модернізації*** і ***заміщення*** ІС.

Супровід – інкрементальний ітеративний процес, у рамках якого виконуються малі зміни в системі, що не зачіпають структурної організації системи.

Модернізація – характеризується як діяльність, яка передбачає значні зміни існуючої системи, але не її утилізацію або заміщення новою системою.

Заміщення – процес, який полягає у впровадженні нової системи, здатної повністю замінити існуючу ІС.

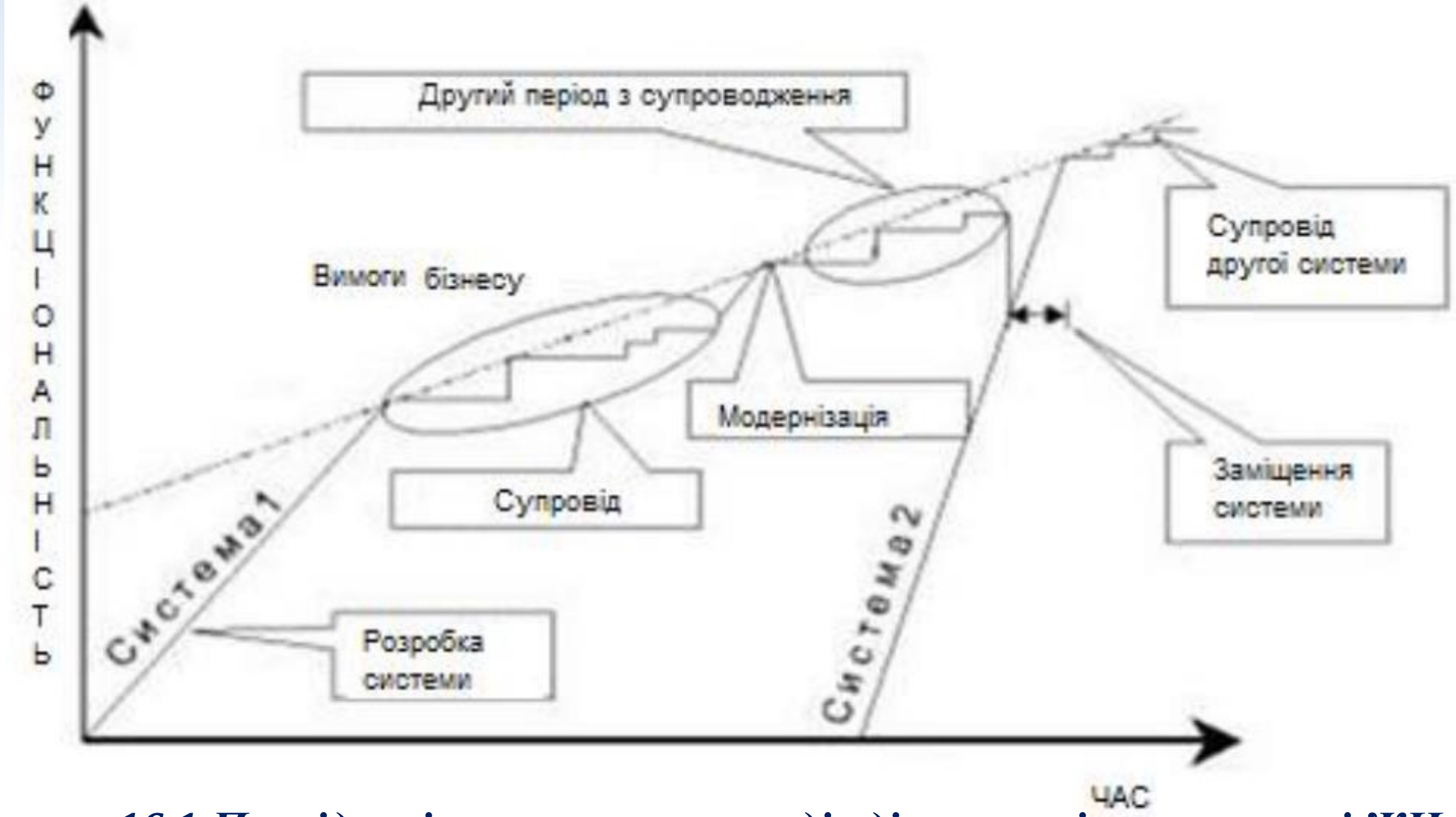


Рисунок 16.1 Послідовність виконання видів діяльності в контексті ЖЦ ІС

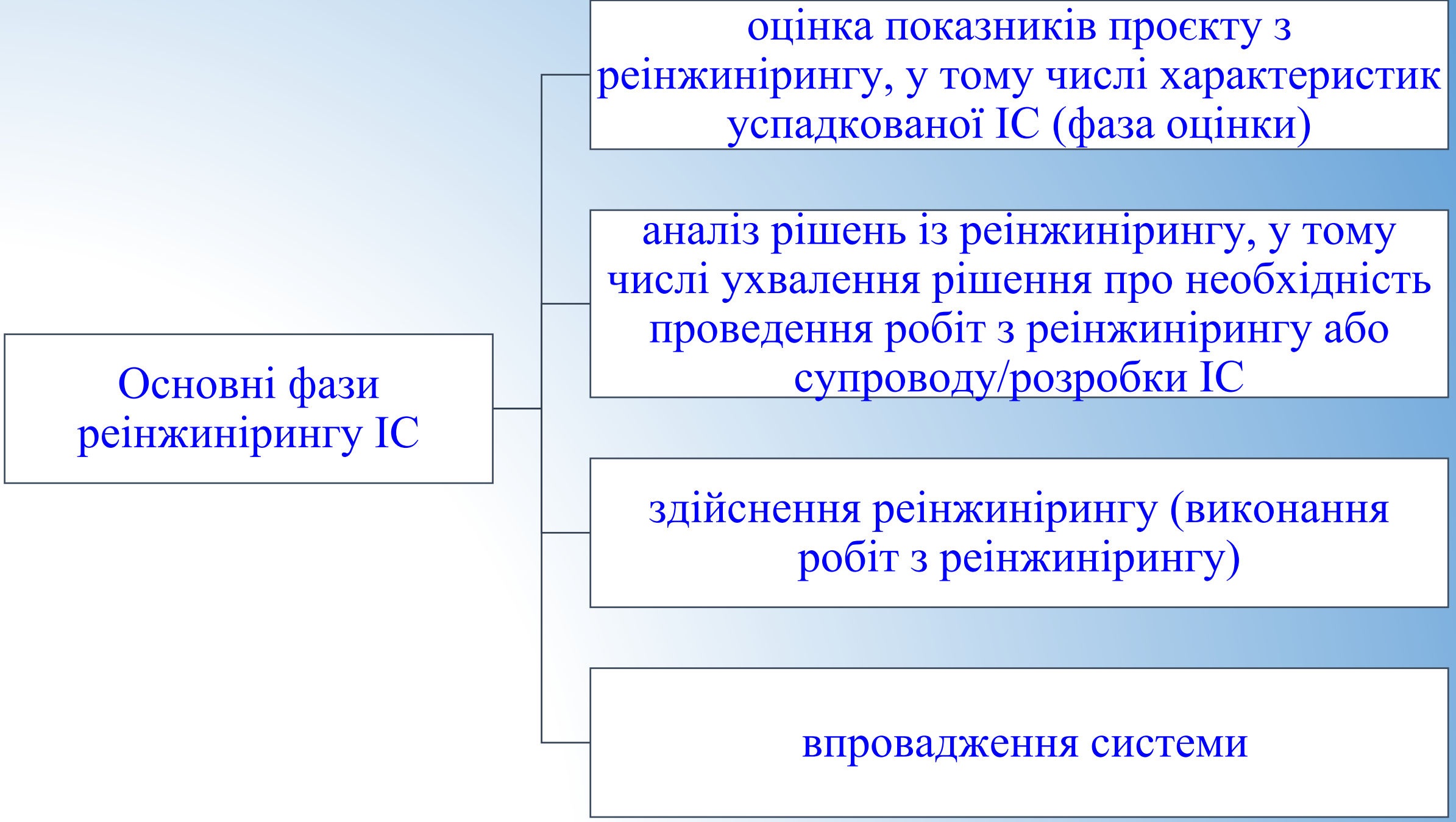

Основні фази реінжинірингу ІС
оцінка показників проєкту з реінжинірингу, у тому числі характеристик успадкованої ІС (фаза оцінки)
аналіз рішень із реінжинірингу, у тому числі ухвалення рішення про необхідність проведення робіт з реінжинірингу або супроводу/розробки ІС
здійснення реінжинірингу (виконання робіт з реінжинірингу)
впровадження системи

Рисунок 16.2


Процеси (види діяльності), що співвідносяться з реінжинірингом ІС
аналіз існуючої системи, заснований на одному або більше її логічних описів
трансформація логічних описів у новий, поліпшений логічний опис системи
розробка нової системи, заснованої на нових логічних описах системи

Рисунок 16.3

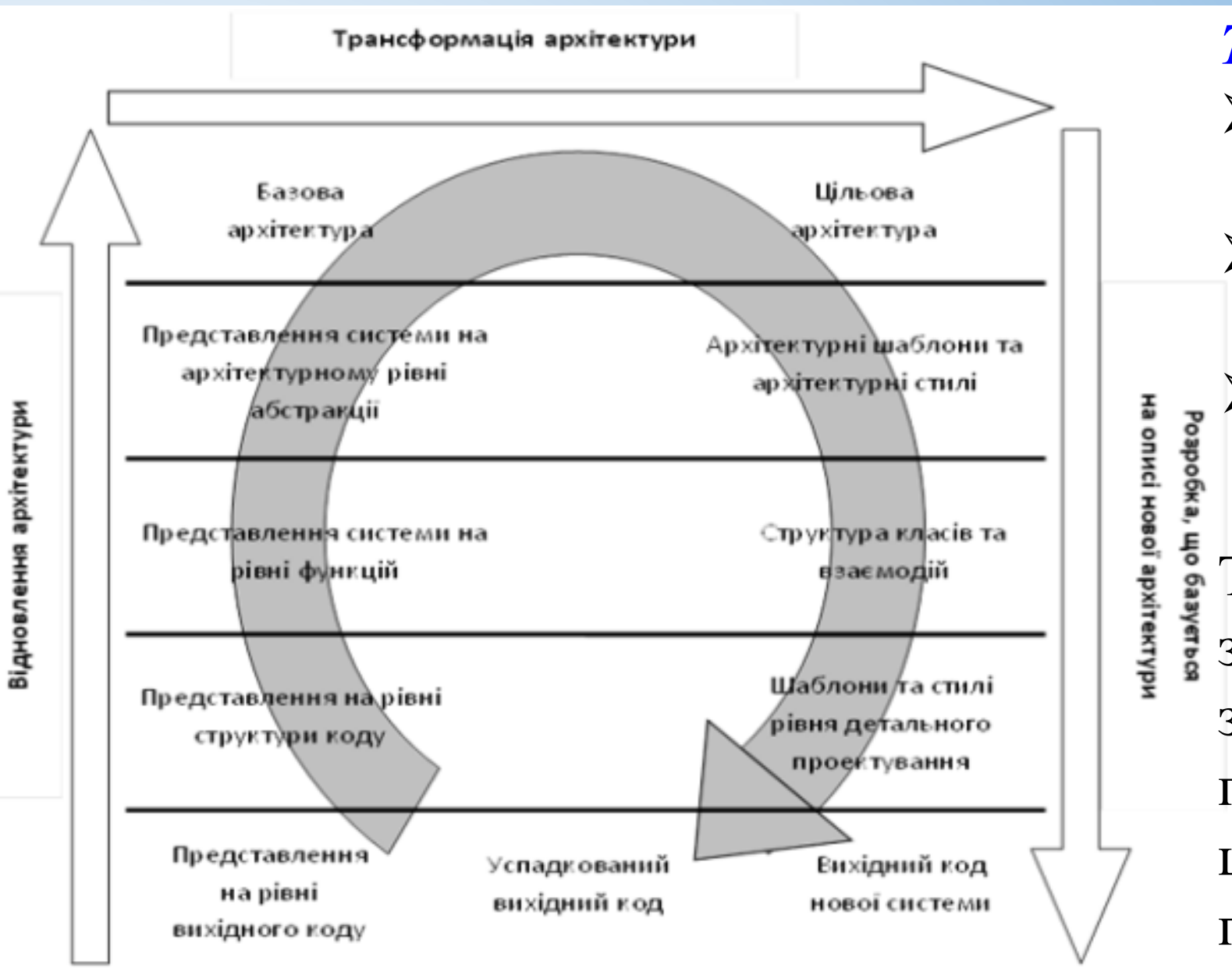


Рисунок 16.4 Модель «Підкови»

Три рівні:

- Представлення на рівні структури коду
- Представлення на рівні функціональності системи
- На архітектурному рівні

Трансформація системи, залежно від ситуації, може здійснюватися на кожному із представлених рівнів, при цьому більш низький рівень підтримує трансформацію на більш високому рівні.

Рисунок 16.5 Кроки процесу реінжинірингу

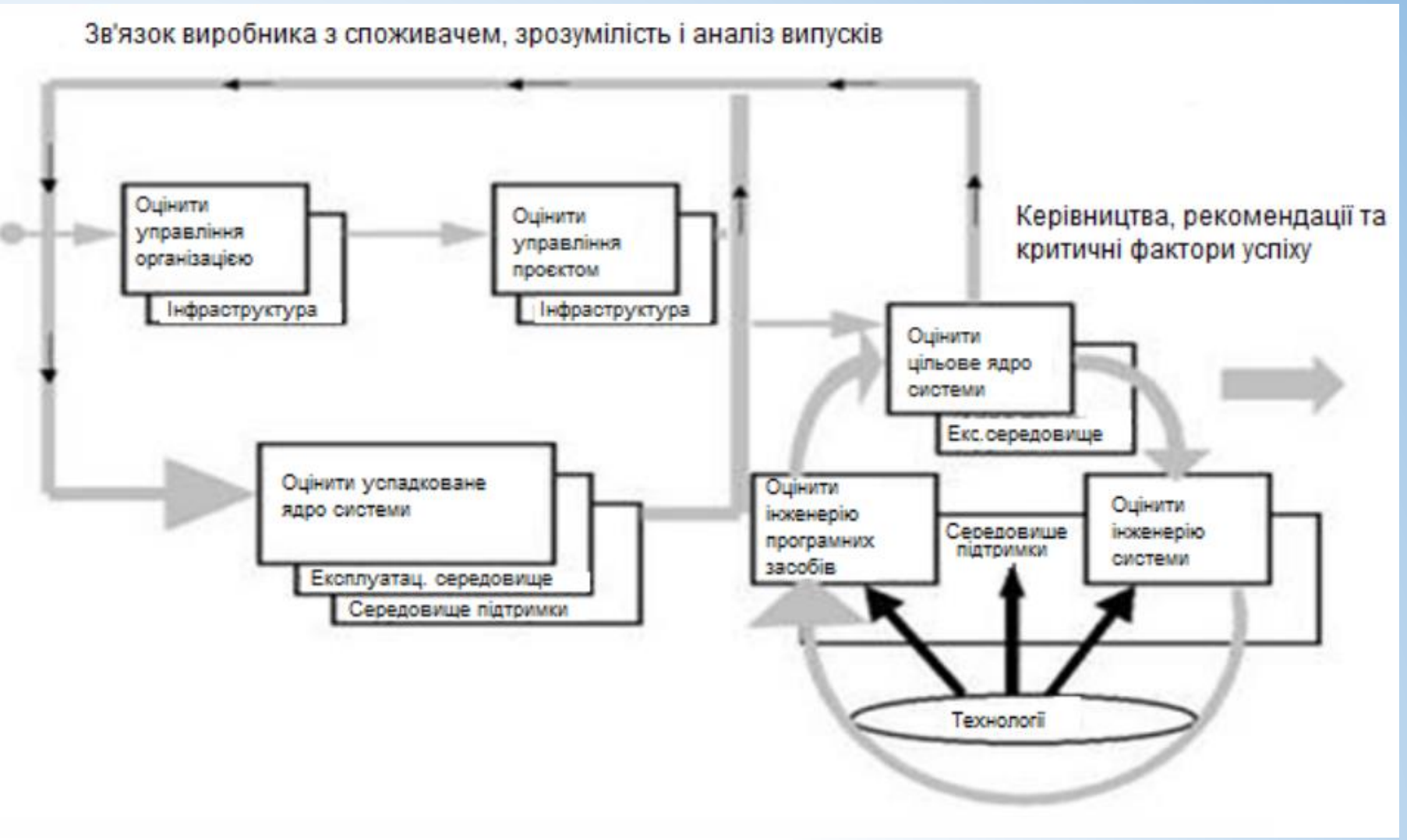
Розглядається ***каркас***, що характеризує глобальне середовище, у якому здійснюється еволюція системи, та дії, процеси й робочі продукти (артефакти), які супроводжують діяльність з еволюції системи.

Основна мета створення каркаса – необхідність оцінки середовища, у рамках якого здійснюється еволюція успадкованої ІС, необхідність забезпечення розуміння широкого спектра питань.

Елементи каркасу: організація, проєкт, успадкована система, інженерія систем і програмних засобів, технології, цільова система.

Каркас може бути використаний для виконання наступних видів діяльності:

- дослідження й аналіз простору проблем і простору рішень;
- розробка опису з складанням стратегічних і тактичних планів;
- виявлення технологічних питань і потенційних проблем;
- рецензування планів, ранжирування (пріоритизація) технічних питань, розробка рекомендацій



У рамках підходу виділяються ***дві значні складові:***

- основна фаза;
- фаза еволюції.

Рисунок 16.5 Загальний підхід до використання каркаса

Рисунок 16.6 Послідовність етапів проведення реінжинірингу

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Морзе Н. В. Інформаційні системи: навч. посіб. / Н. В. Морзе, О. З. Піх. Івано-Франківськ: ЛілеяНВ, 2015. - 384 с.
2. Технології проектування комп'ютерних систем: навч. посібник / Б. Г. Масловський, В. І. Дрововозов, О. В. Коба. - Київ : НАУ, 2015. - 499 с.
3. Ременяк Л. В. Проектування інформаційних систем: конспект лекцій / Л.В. Ременяк. - Одеса: ОДЕУ, 2016. - 152 с.
4. Проектування інформаційних систем: навч. посіб. / В. С. Авраменко, А. С. Авраменко. Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. - 434 с.
5. Карпенко М. Ю. Технології створення програмних продуктів та інформаційних систем: навч. посіб. / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова.- Харків : ХНУМГ ім. О. М. Бекетова, 2017. - 93 с.
6. Чиста архітектура: мистецтво розробки програмного забезпечення / Роберт Мартін. Х.:Фабула, 2019.- 416 с.

7. Конспект лекцій з дисципліни «Проектування інформаційних систем» для студентів спеціальності 122 Комп’ютерні науки денної та заочної форми навчання / І. В. Ярош, Т. О. Черняк. Покровськ: ДВНЗ «ДонНТУ», 2019. - 49 с. http://ea.donntu.edu.ua/jspui/handle/123456789/30525
8. Проектування інформаційних систем: загальні питання теорії проектування ІС: навч. посіб. для студ. спеціальності 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського; уклад.: О. С. Коваленко, Л. М. Добровська. - Київ: КПІ ім. Ігоря Сікорського, 2020. - 192 с.
9. Литвин В. В. Проектування інформаційних систем : навч. посіб. (затв. МОН України) / В. В. Литвин, Н. Б. Шаховська; за наук. ред. В. В. Пасічника. - Львів: «Магнолія 2006», 2021. - 380 с.
10. Інформаційна система [Електронний ресурс] : Вікіпедія: Вільна енциклопедія. Режим доступу: https://uk.wikipedia.org/wiki/Інформаційна_система. - Назва з екрана.
11. Коваленко О. С. Проектування інформаційних систем: Загальні питання теорії проектування ІС [Електронний ресурс] / О.С. Коваленко, Л.М. Добровська. - Режим доступу: https://ela.kpi.ua/bitstream/123456789/33651/1/PIS_KL.pdf. - Назва з екрана.
12. Марченко А. В. Проектування інформаційних систем [Електронний ресурс] / Марченко А. В. - Режим доступу : https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf. - Назва з екрана.