Методи проєктування інформаційних систем

Ми розглянемо різні методології та інструменти, від класичних структурних підходів до сучасних об'єктно-орієнтованих парадигм, а також значення CASE-технологій у сучасній розробці. Приготуйтеся зануритися у світ систематичного підходу до розробки програмного забезпечення та інформаційних систем.

Лектор: професор Євгеній БОРОДАВКА

План лекції

1. **Життєвий цикл інформаційних систем**
   Концепція, моделі та етапи життєвого циклу інформаційних систем
2. **Структурні діаграми**
   SADT, DFD, ERD, STD та їх застосування у проєктуванні
3. **Об'єктно-орієнтовані методи проєктування**
   Концепції ООП та схеми алгоритмів
4. **CASE-технології**
   Комп'ютерні засоби проєктування інформаційних систем

Життєвий цикл інформаційних систем

Життєвий цикл інформаційних систем

Життєвий цикл інформаційної системи (ЖЦ ІС) — це структурований підхід до організації та управління всіма процесами, які проходить інформаційна система від ідеї її створення до повного виведення з експлуатації. Він охоплює етапи планування, аналізу вимог, проєктування, реалізації, тестування, впровадження, експлуатації та підтримки, а також завершення її використання. Метою ЖЦ ІС є забезпечення ефективного створення, функціонування та розвитку системи, що відповідає потребам користувачів та бізнесу.

Міжнародний стандарт ISO/IEC/IEEE 12207 Systems and software engineering, а також його український відповідник ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів, є ключовими документами, що детально визначають структуру життєвого циклу ІС. Стандарт описує основні, допоміжні та організаційні процеси, включно з процесами придбання, постачання, розробки, експлуатації та супроводу. Цей стандарт слугує основою для розробки, впровадження та вдосконалення процесів створення та підтримки програмних систем, гарантуючи їхню якість та надійність.

Основні моделі життєвого циклу

Каскадна модель

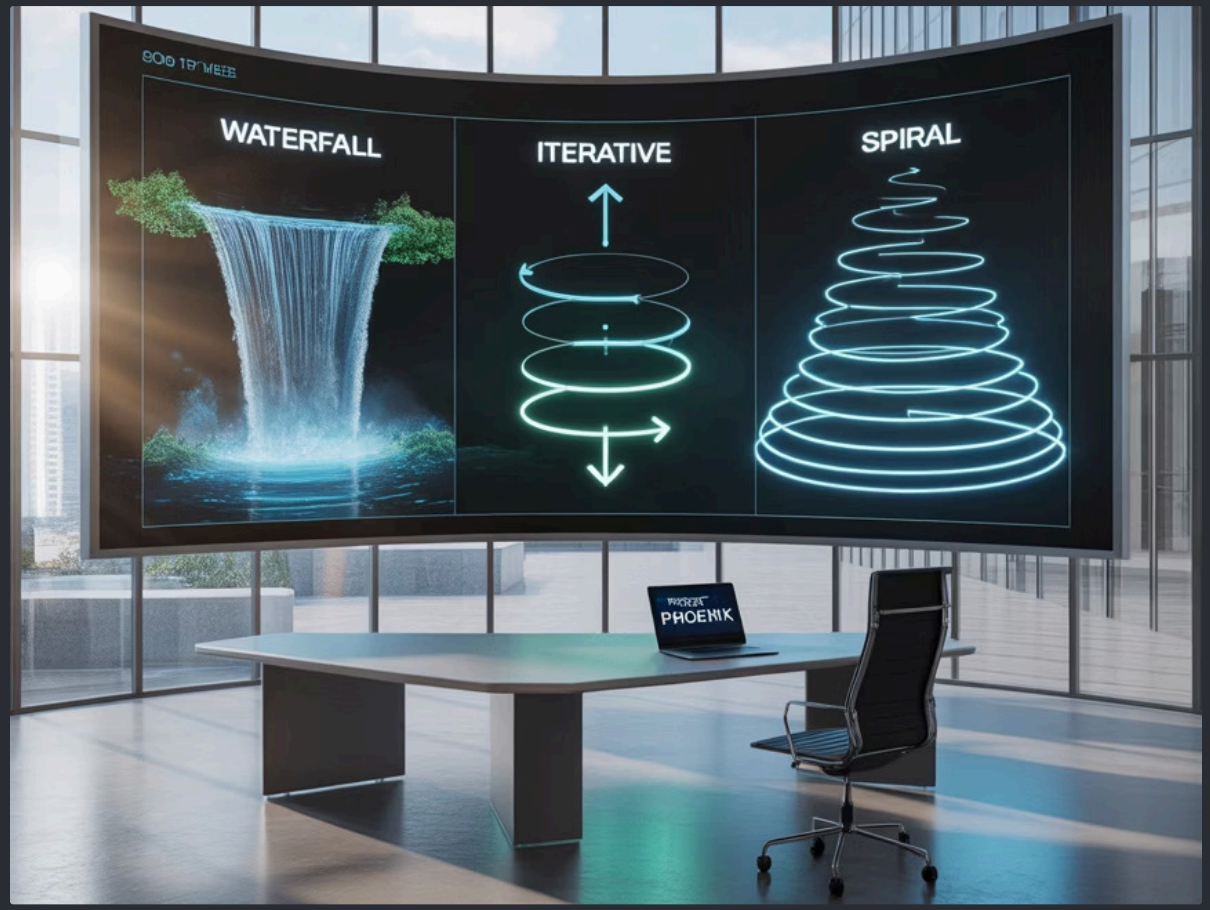
Послідовне виконання всіх етапів проєкту в строго фіксованому порядку. Перехід до наступного етапу тільки після повного завершення попереднього.

Ітераційна модель

Розробка системи з поетапним наближенням до потрібного результату, з можливістю повернення до попередніх етапів.

Спіральна модель

Поєднує в собі елементи каскадної та ітераційної моделей з особливим акцентом на аналіз ризиків.



Етапи життєвого циклу ІС

1

Планування та аналіз вимог

Визначення призначення системи, вимог користувачів, функціональних можливостей, технічних обмежень.

2

Проєктування

Розробка архітектури, інтерфейсів, бази даних, вибір платформи та інструментів.

3

Реалізація (програмування)

Кодування модулів, створення бази даних, інтеграція компонентів.

4

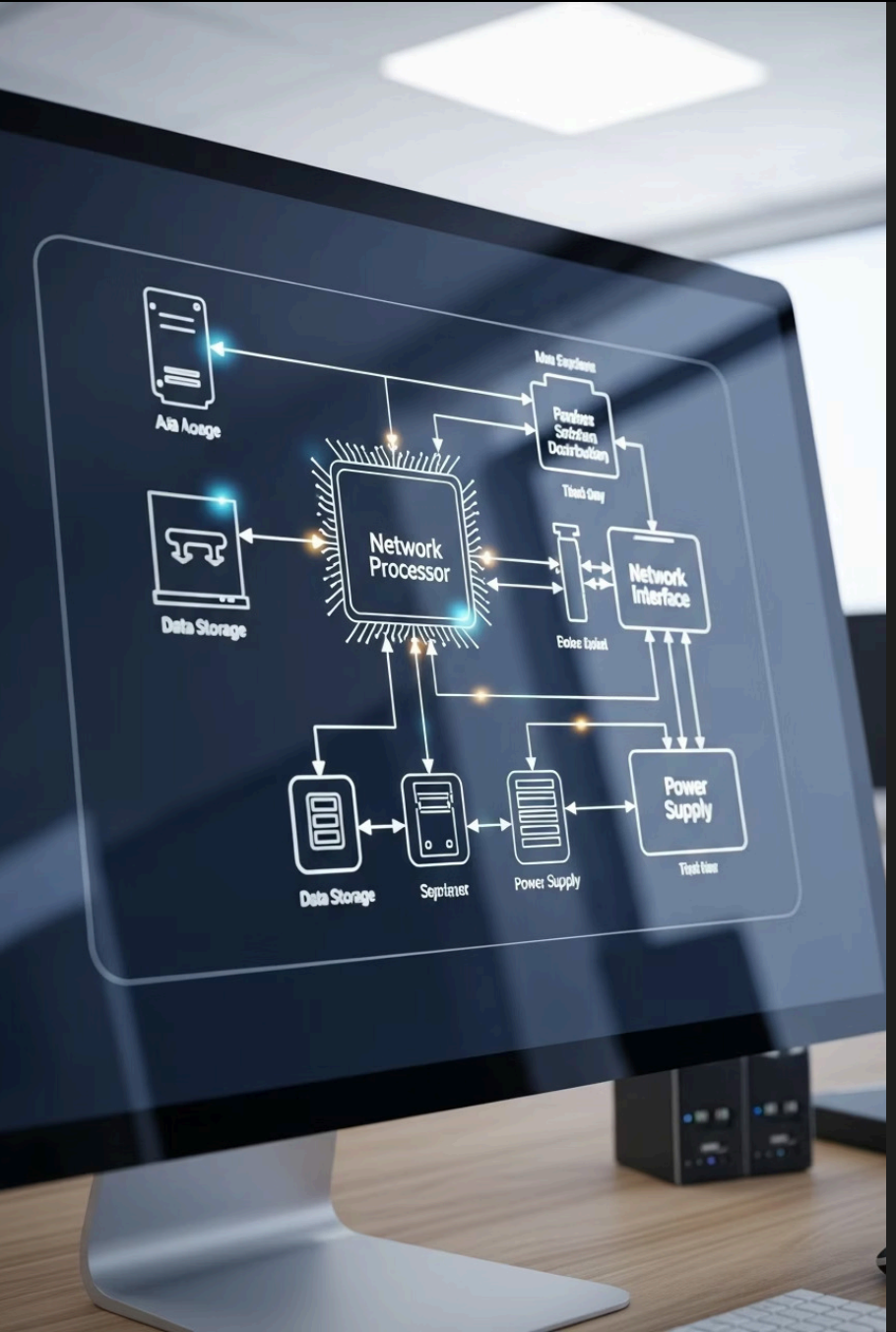
Тестування та впровадження

Верифікація, валідація, виправлення помилок, розгортання системи.

5

Супровід та модернізація

Технічна підтримка, усунення недоліків, вдосконалення функціональності.



Структурні діаграми

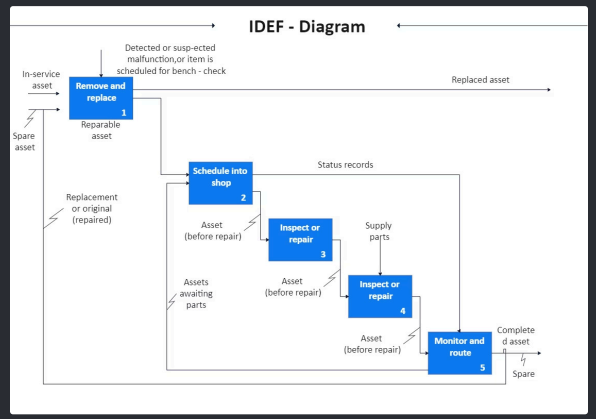
Типи структурних діаграм

Структурні діаграми є невід'ємним інструментом візуального моделювання інформаційних систем. Вони дозволяють детально представити архітектуру та логіку системи, розбиваючи її на зрозумілі, ієрархічні компоненти. Це значно спрощує аналіз вимог, проєктування взаємодії між модулями та комунікацію в команді розробників.

Далі ми детально розглянемо чотири ключові типи структурних діаграм, кожна з яких виконує свою унікальну функцію у життєвому циклі ІС: SADT-діаграми для функціональної декомпозиції, DFD-діаграми для моделювання потоків даних, ERD-діаграми для проєктування баз даних та STD-діаграми для опису поведінки системи. Розуміння цих інструментів є критично важливим для ефективного та систематичного проєктування.

Structured Analysis and Design Technique (SADT)

SADT (Structured Analysis and Design Technique) – це потужна методологія структурного аналізу та проєктування, яка забезпечує графічне представлення складної системи у вигляді ієрархії взаємопов'язаних діаграм. Вона дозволяє візуалізувати як функціональні аспекти системи, так і взаємодію між її компонентами.



Основна концепція SADT полягає у послідовній декомпозиції складної системи на менші, більш керовані функціональні блоки. Кожен блок представляє окремий процес або функцію, а зв'язки між ними чітко відображають потоки інформації та матеріальних об'єктів, забезпечуючи чітке розуміння логіки роботи системи від загального до найменших деталей.

SADT-діаграми базуються на нотації IDEF0 (Integration Definition for Function Modeling), яка використовує наступні графічні елементи для побудови моделей:

- **Прямокутники (блоки активності)** - представляють функції, дії або процеси, що виконуються в системі. Кожен прямокутник має унікальну назву та номер.
- **Стрілки** - відображають взаємодію та залежності між функціями. Стрілки приєднуються до сторін прямокутника і мають чотири типи значень:
  - Вхід (Input) - дані або ресурси, що споживаються функцією (ліва сторона прямокутника).
  - Вихід (Output) - результати або продукти, що генеруються функцією (права сторона прямокутника).
  - Управління (Control) - правила, умови або обмеження, що впливають на виконання функції (верхня сторона прямокутника).
  - Механізм (Mechanism) - ресурси (люди, інструменти, системи), що виконують функцію (нижня сторона прямокутника).

Методології IDEF

IDEF (Integration Definition) — це широка родина методологій моделювання, що активно застосовуються в інженерії систем та програмного забезпечення. Вони дозволяють детально описувати та аналізувати різні аспекти системи: від функціонального моделювання до моделювання даних та об'єктно-орієнтованого проєктування.

IDEF0: Function Modeling

Фокусується на представленні функціональної структури системи, відображаючи, що робить система та які вхідні/вихідні дані, механізми та керуючі елементи використовуються.

IDEF1: Information Modeling

Методологія моделювання інформаційних потоків усередині системи, що дозволяє відображати та аналізувати їх структуру та взаємозв'язки.

IDEF1X: Data Modeling

Використовується для проектування баз даних, надаючи графічне представлення сутностей, їх атрибутів та взаємозв'язків у системі.

IDEF2: Simulation Model Design

Методологія динамічного моделювання розвитку систем.

Методології IDEF: Розширення можливостей моделювання

Окрім IDEF0 та IDEF1X, існує низка інших методологій сімейства IDEF, кожна з яких розроблена для моделювання специфічних аспектів системи, дозволяючи створити всебічний та інтегрований опис.

IDEF3: Process Description Capture

Фокусується на моделюванні послідовностей подій та процесів, що відбуваються в системі, показуючи їх взаємозв'язки, альтернативні шляхи та паралельні дії. Забезпечує детальне відображення динамічної поведінки.

IDEF5: Ontology Description Capture

Призначена для створення та аналізу онтологій, тобто формалізованих представлень знань про певні сфери. Дозволяє чітко визначити сутності, їхні властивості та відношення між ними для забезпечення спільного розуміння.

IDEF4: Object-Oriented Design

Застосовується для об'єктно-орієнтованого моделювання, допомагаючи у проєктуванні та розробці програмних систем з використанням об'єктно-орієнтованих парадигм. Акцентує увагу на класах, об'єктах та їх взаємодії.

Використання різних методологій IDEF дозволяє розробникам та аналітикам отримувати глибоке та багатогранне розуміння складності інформаційних систем, сприяючи ефективному плануванню, проєктуванню та впровадженню.

Data Flow Diagrams (DFD)

Діаграми потоків даних візуалізують рух інформації в системі. Вони показують, звідки надходять дані, як вони обробляються та куди передаються.

Основні елементи DFD

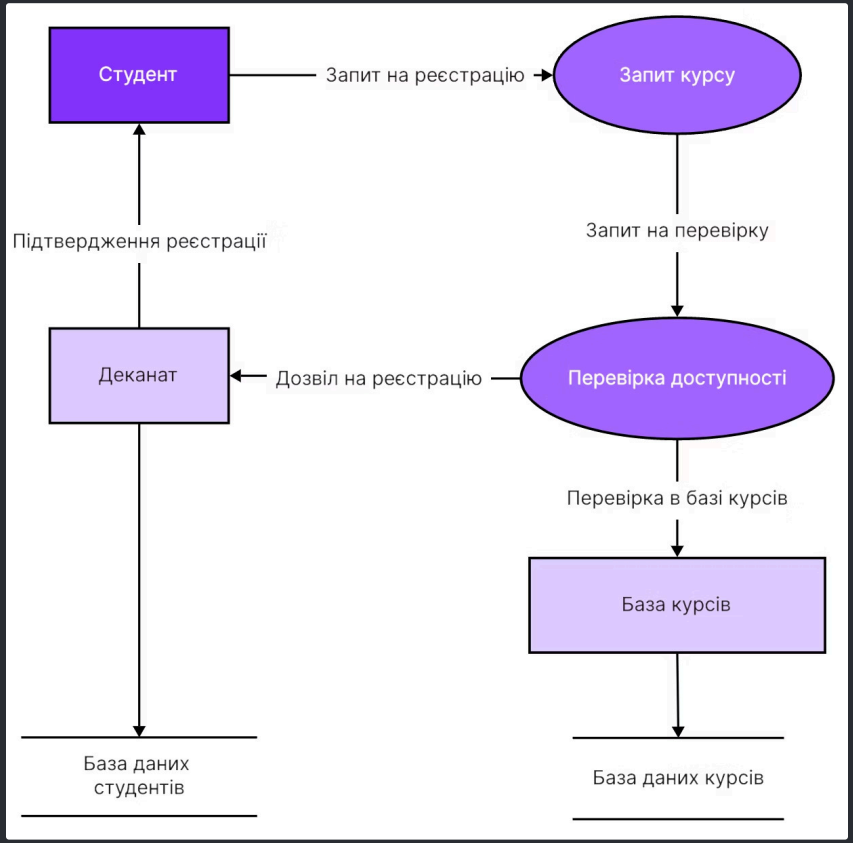
- Процеси (функції системи) - кола або округлені прямокутники
- Зовнішні сутності (джерела або приймачі даних) - прямокутники
- Сховища даних (файли, бази даних) - паралельні лінії або відкриті прямокутники
- Потоки даних - стрілки з назвами даних, що передаються

Рівні деталізації DFD

- Контекстна діаграма (найвищий рівень)
- Діаграма 0-го рівня (деталізація системи)
- Діаграми 1-го і нижчих рівнів (деталізація окремих процесів)

Приклад діаграми потоків даних

На цій діаграмі потоків даних (DFD) подано спрощений приклад системи реєстрації студентів на курси в університеті. Вона наочно демонструє, як дані переміщуються та обробляються між різними компонентами системи.

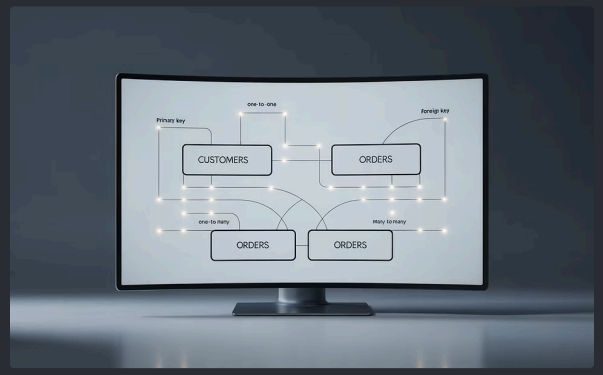


Ключові елементи цього DFD:

- **Зовнішні сутності:**
  - «Студент» – ініціює запит на реєстрацію та отримує підтвердження.
  - «База курсів» – надає інформацію про доступні курси.
  - «Деканат» – отримує оновлені списки студентів та графіки.
- **Процеси:**
  - «Запит курсу» – обробляє запит студента.
  - «Перевірка доступності» – перевіряє наявність місць на курсі.
  - «Оновлення записів» – вносить зміни до студентських та курсових баз даних.
- **Сховища даних:**
  - «Записи студентів» – зберігає інформацію про студентів та їхні реєстрації.
  - «База даних курсів» – містить деталі про всі курси університету.
- **Потоки даних:** Стрілки вказують напрямок руху інформації, наприклад, "Запит на реєстрацію" від студента до процесу "Запит курсу", або "Підтвердження реєстрації" від системи до студента.

Entity-Relationship Diagrams (ERD)

Діаграми "сутність-зв'язок" використовуються для моделювання структури даних системи. Вони відображають сутності (об'єкти), їх атрибути та зв'язки між ними.



Сутності

Прямокутники, що представляють об'єкти реального світу, такі як "Студент" або "Курс".

Атрибути

Властивості, що описують сутності, наприклад, "Ім'я", "Прізвище" або "Дата народження".

Зв'язки

Лінії між сутностями, що показують їх взаємодію: один-до-одного (1:1), один-до-багатьох (1:N) або багато-до-багатьох (M:N).

ERD є основою для розробки бази даних інформаційної системи.

Порівняння нотацій ERD

У світі моделювання даних існує кілька нотацій для побудови діаграм "сутність-зв'язок" (ERD), кожна з яких має свої переваги та застосування. Найбільш відомими є нотація Пітера Чена та "Вороняча лапка".

Нотація Пітера Чена

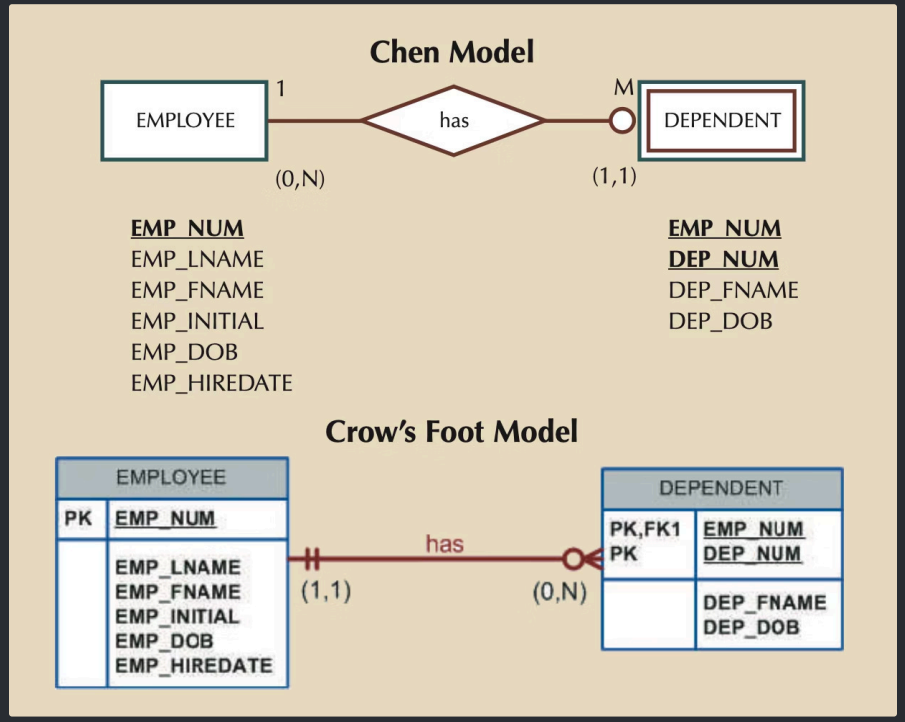
Ця класична нотація акцентує увагу на концептуальному моделюванні, візуально розділяючи сутності, атрибути та зв'язки.

- **Сутності:** Прямокутники.
- **Атрибути:** Овали, що з'єднуються із сутностями.
- **Зв'язки:** Ромби, що з'єднують сутності.
- **Потужність зв'язків:** Позначається числами (1, N, M) поруч з ромбом.

Нотація "Вороняча лапка" (Crow's Foot)

Ця нотація є більш популярною для логічного та фізичного проєктування баз даних завдяки своїй компактності та ясності відображення кардинальності.

- **Сутності:** Прямокутники з назвами та списком атрибутів.
- **Зв'язки:** Лінії між сутностями.
- **Потужність та обов'язковість:** Графічні символи на кінцях ліній, що нагадують "воронячі лапки".



State Transition Diagrams (STD)

Діаграми переходів станів моделюють поведінку системи з точки зору змін її станів у відповідь на зовнішні події.

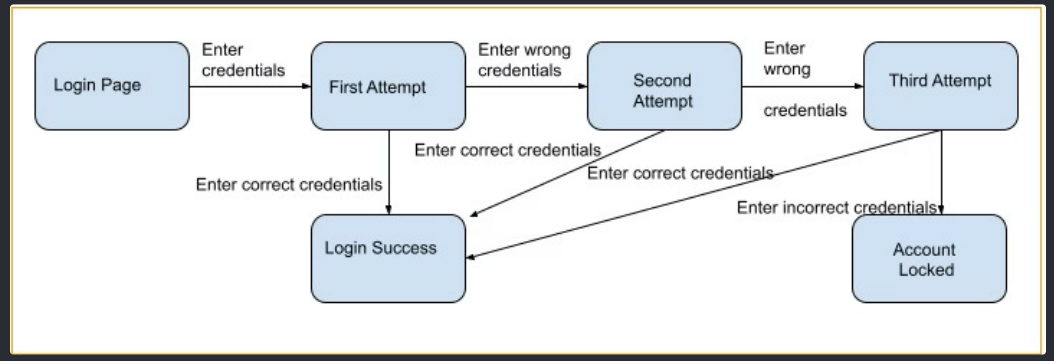
Елементи STD

- Стани (прямокутники із заокругленими кутами)
- Переходи (стрілки між станами)
- Події (підписи на стрілках)
- Дії (вказуються в станах або на переходах)

Застосування STD

- Моделювання життєвого циклу об'єктів
- Опис поведінки інтерфейсу користувача
- Визначення послідовності операцій
- Моделювання реакції системи на події

Подана діаграма переходу станів ілюструє процес автентифікації користувача в інформаційній системі. Користувач має три спроби, щоб ввести правильні облікові дані. У разі хибних даних після перших двох спроб, користувач має можливість спробувати ще раз. Якщо після будь-якої зі спроб буде введено правильні дані – користувач отримає доступ до системи. Після третьої невдалої спроби акаунт користувача буде заблоковано.



Об'єктно-орієнтовані методи проєктування

Об'єктно-орієнтоване проєктування

Об'єктно-орієнтовані методи проєктування інформаційних систем (ООП) відходять від традиційного процедурного підходу, зосереджуючись на створенні моделі реального світу через взаємодію незалежних об'єктів. Кожен об'єкт інкапсулює дані (атрибути) та поведінку (методи).

Цей підхід, що базується на таких принципах, як інкапсуляція, успадкування та поліморфізм, забезпечує значні переваги. Він дозволяє розробляти більш гнучкі, масштабовані та модульні системи, які легше підтримувати, розширювати та повторно використовувати в різних проєктах, що особливо важливо для складних бізнес-процесів.

Основні принципи ООП

Інкапсуляція

Об'єднання даних та методів їх обробки в єдину структуру з обмеженням доступу до внутрішньої реалізації.

Успадкування

Створення нових класів на основі існуючих з успадкуванням їх властивостей та поведінки.

Поліморфізм

Здатність об'єктів з однаковим інтерфейсом мати різну реалізацію методів.

Абстракція

Виділення суттєвих характеристик об'єкта, які відрізняють його від інших видів об'єктів.

UML-діаграми в ООП

Unified Modeling Language (UML) - стандартизована мова моделювання для об'єктно-орієнтованого аналізу та проєктування інформаційних систем.

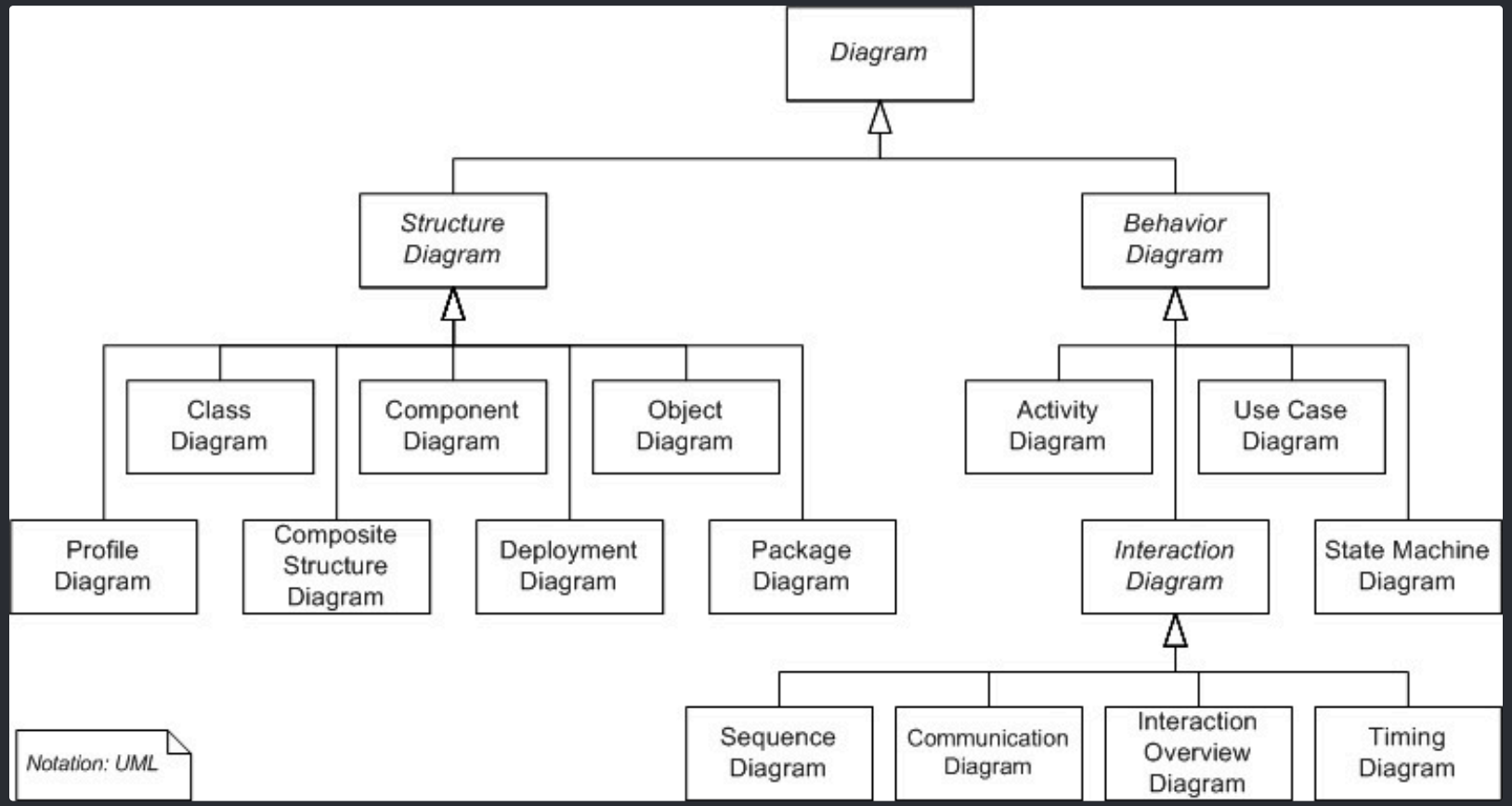
Структурні діаграми UML

- Діаграма класів (Class Diagram)
- Діаграма об'єктів (Object Diagram)
- Діаграма компонентів (Component Diagram)
- Діаграма розгортання (Deployment Diagram)

Поведінкові діаграми UML

- Діаграма прецедентів (Use Case Diagram)
- Діаграма послідовності (Sequence Diagram)
- Діаграма діяльності (Activity Diagram)
- Діаграма станів (State Diagram)

Ієрархія типів UML-діаграм



UML Діаграми Класів Для Побудови Складних Систем

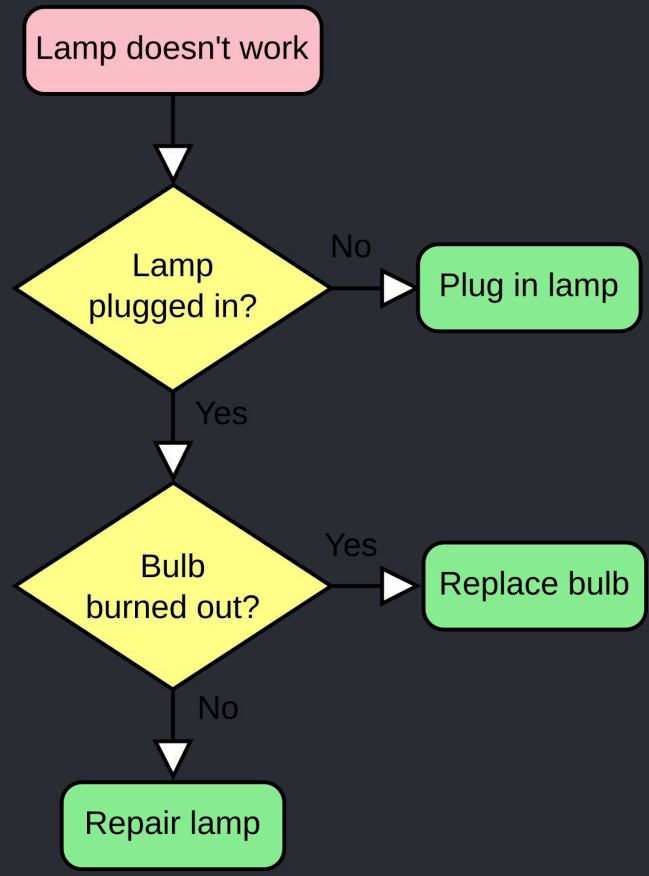
Схеми алгоритмів

Схеми алгоритмів (блок-схеми) - графічне представлення алгоритму у вигляді послідовності блоків різних типів, з'єднаних лініями, що показують порядок виконання операцій.

Основні елементи блок-схем:

- Термінатор (початок/кінець)
- Процес (обчислення)
- Рішення (умова)
- Введення/виведення даних
- Лінії потоку (напрямок виконання)

Схеми алгоритмів регулюються стандартом ДСТУ ISO 5807:2016 Обробляння інформації. Символи та угоди щодо документації стосовно даних, програм та системних блок-схем, схем мережевих програм та схем системних ресурсів (ISO 5807:1985, IDT).



Переваги використання блок-схем

Наочність

Візуальне представлення алгоритму полегшує розуміння логіки роботи програми

Комунікація

Спрощує обговорення логіки алгоритму з колегами та замовниками

Аналіз

Допомагає виявити логічні помилки та неоптимальні рішення на ранньому етапі

Планування

Сприяє структурованому підходу до розробки програмного забезпечення

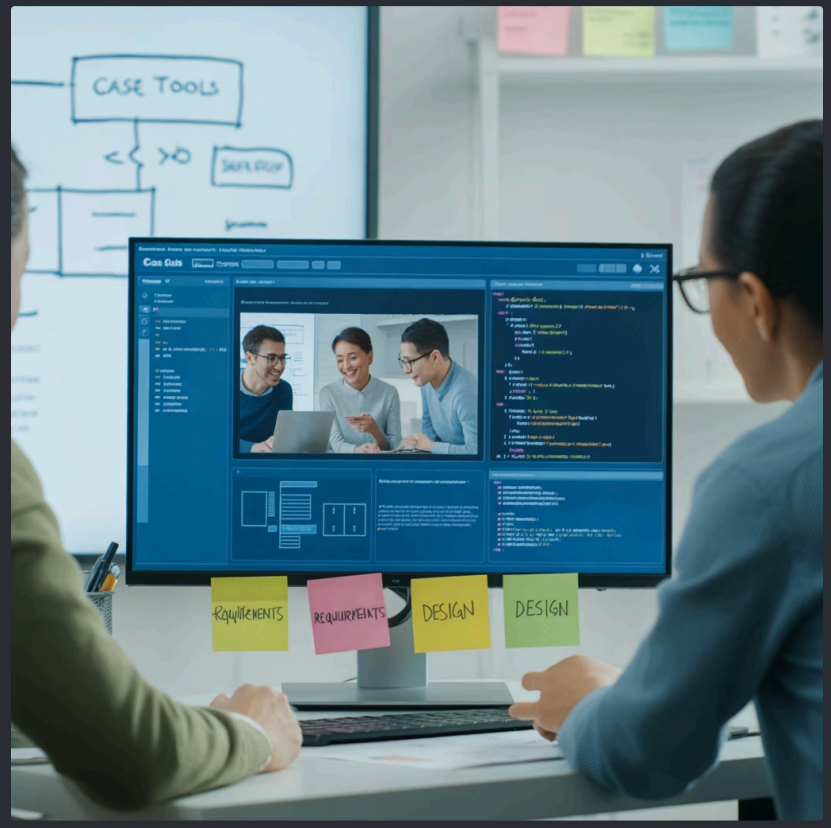
Блок-схеми особливо корисні на етапі проєктування складних алгоритмів, коли необхідно чітко визначити послідовність дій та умови переходів між ними.

CASE-технології

CASE-засоби

CASE (Computer-Aided Software Engineering) – це інтегровані програмні засоби та технології, призначені для автоматизації всього життєвого циклу розробки програмного забезпечення та інформаційних систем. Вони охоплюють процеси від початкового аналізу вимог та проєктування архітектури до кодування, тестування, документування та супроводу системи.

Використання CASE-засобів дозволяє значно підвищити продуктивність праці розробників завдяки автоматизації рутинних операцій, покращити якість створюваних систем шляхом виявлення помилок на ранніх етапах та підтримки стандартизації. Крім того, вони забезпечують детальне та послідовне документування кожного етапу процесу розробки, що критично важливо для великих проєктів, а також значно полегшують подальший супровід та модифікацію системи протягом усього її життєвого циклу.



ЯК і ДЕ створити UML-діаграму? + БОНУС

Класифікація CASE-засобів

За типом

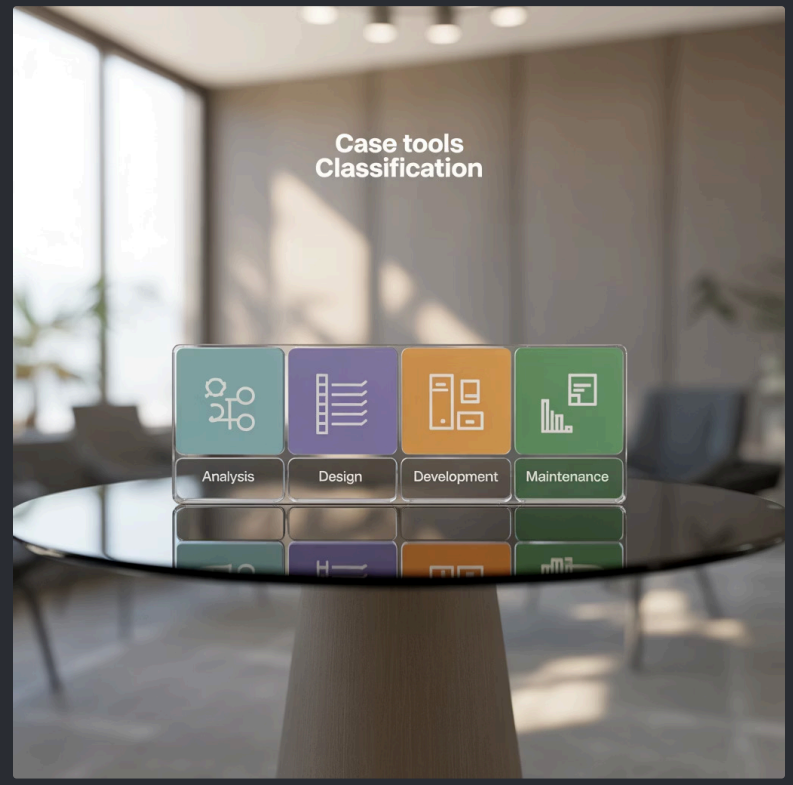
- Аналіз і проєктування (Upper CASE)
- Розробка (Middle CASE)
- Супровід та реінжиніринг (Lower CASE)
- Інтегровані (I-CASE)

За функціональністю

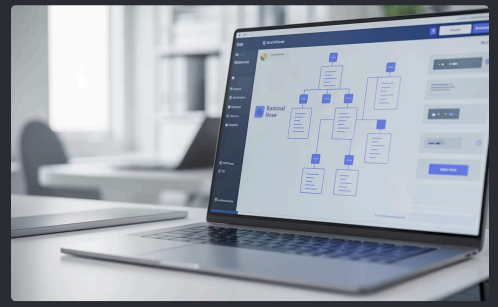
- Засоби аналізу (BPwin, ERwin)
- Засоби проєктування БД (ERwin, Oracle Designer)
- Засоби розробки інтерфейсів
- Засоби програмування
- Засоби тестування

За підтримкою методологій

- Структурного підходу
- Об'єктно-орієнтованого підходу
- Змішаного підходу

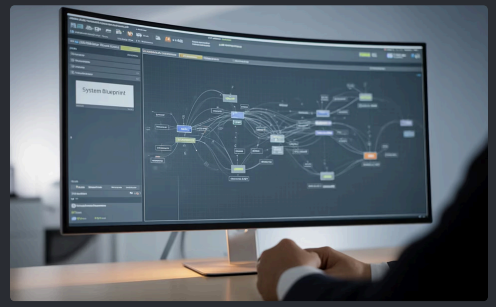


Найпопулярніші CASE-засоби



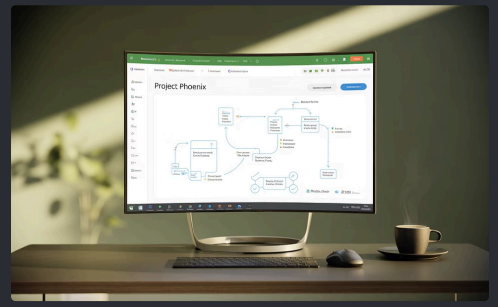
Rational Rose

Засіб візуального моделювання для об'єктно-орієнтованого аналізу та проєктування з підтримкою UML



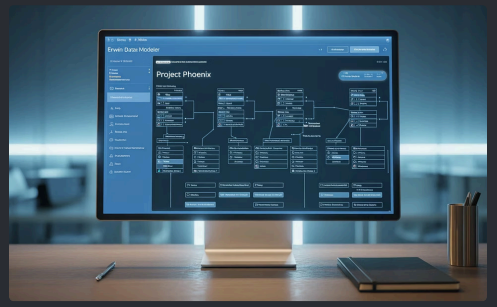
Enterprise Architect

Комплексний інструмент для моделювання, проєктування та управління проєктами



Microsoft Visio

Інструмент для створення діаграм бізнес-процесів, блок-схем та інших видів технічних схем



ERwin Data Modeler

Спеціалізований засіб для моделювання баз даних з можливістю генерації SQL-скриптів

Онлайн засоби: **drawio.com**, **boardmix.com**, **aspose.app**, **astah.net**, **miro.com**

Переваги використання CASE-технологій

Підвищення якості ІС

Формалізація процесів розробки та автоматизація контролю дозволяють виявляти помилки на ранніх стадіях

Скорочення термінів розробки

Автоматизація рутинних операцій, повторне використання компонентів

Покращення документування

Автоматичне генерування документації відповідно до стандартів

Централізоване сховище

Зберігання метаданих проєкту в єдиному репозиторії

Управління проєктом

Координація роботи команди розробників, контроль версій

Підтримка технологій

Впровадження сучасних методологій розробки та стандартів проєктування

Перспективи розвитку методів проєктування ІС

Сучасні методи проєктування інформаційних систем постійно розвиваються, інтегруючи новітні технології та адаптуючись до зростаючих вимог бізнесу.

Інтеграція з DevOps

Поєднання методів проєктування з практиками неперервної інтеграції та розгортання (CI/CD)

Автоматизація проєктування

Використання штучного інтелекту для автоматизації процесів аналізу та проєктування

Гнучкі методології

Розвиток Agile-підходів та їх інтеграція з традиційними методами проєктування

Моделеорієнтована розробка

Розвиток технологій генерації коду та розгортання на основі моделей (Model-Driven Development)

Ефективне проєктування інформаційних систем потребує комплексного застосування різних методологій та інструментів, вибір яких залежить від специфіки проєкту, вимог замовника та компетенцій команди розробників.

Вдячний за увагу!