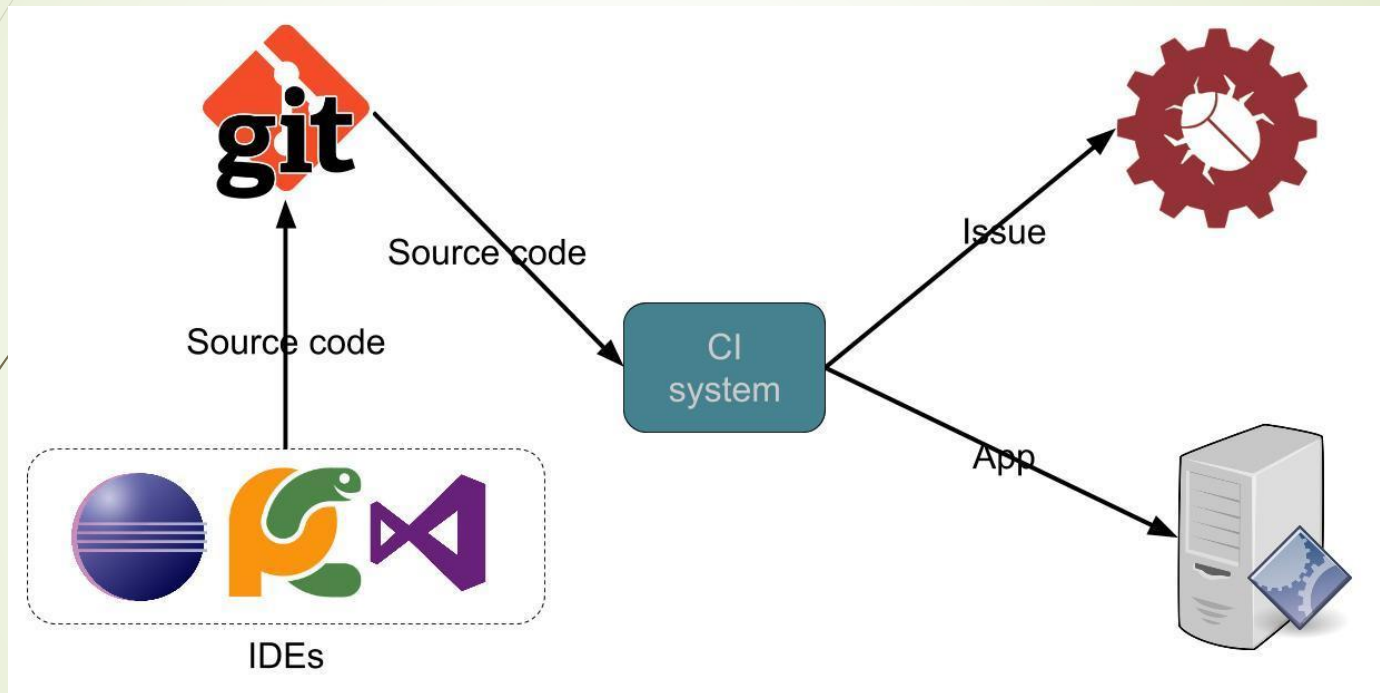


# Технологія створення програмних продуктів



Автоматизація розробки

# Сбірка проектів

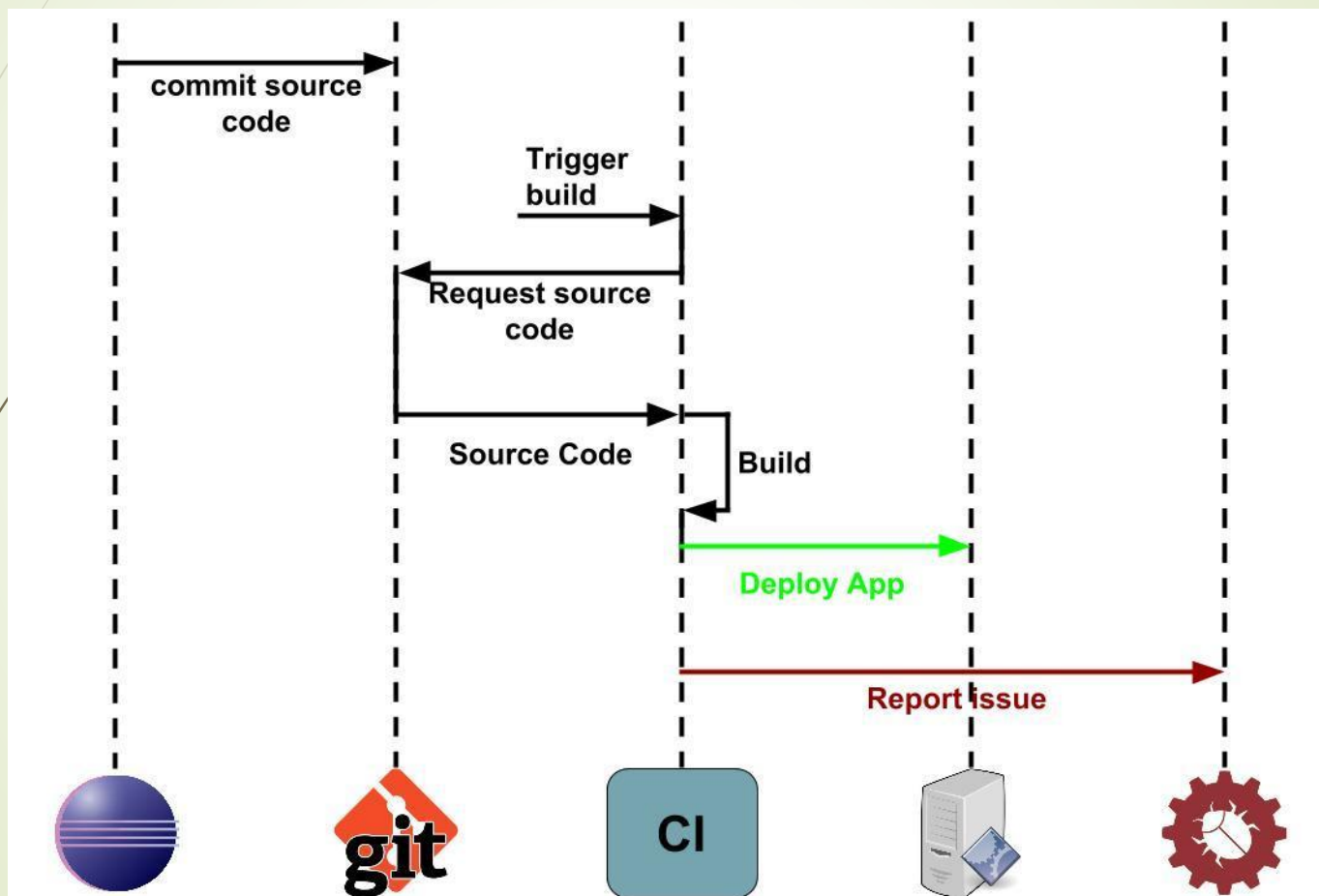


# Сбірка проектів

На малюнку ми бачимо наступні сутності:

- **IDE**  
Розробник. Пише код і в якийсь момент публікує його в репозиторій вихідних кодів.
- **Репозиторій**  
Сховище вихідного коду і різних метаданих про нього. Звідси система безперервної інтеграції завантажує код в процесі складання.
- **Система безперервної інтеграції**  
Дозволяє автоматизувати процеси збирання вихідних кодів, публікації зібраних додатків, або інформації про виявлені проблеми.
- **Стенд**  
В даному випадку сервер, на який завантажується додаток після успішної збірки.
- **Трекер помилок**  
Система контролю і відстеження помилок в роботі або при збірці додатку. Саме сюди система безперервної інтеграції може записати звіт в разі неможливості успішно завершити збірку додатку.

# Сбірка проєктів





# Інструменти та інструментальні середовища розробки ПЗ

За функціональним призначенням інструментальні середовища можна розбити на чотири групи:

- редактори;
- аналізатори;
- транслятори;
- інструменти підтримки процесів виконання програм.

# Інструменти та інструментальні середовища розробки ПЗ

**Редактори** призначені для підтримки конструювання тих чи інших програмних документів на різних етапах ЖЦ.

**Аналізатори** призначені для статичної обробки документів чи динамічного аналізу програм.

**Транслятори** призначені для автоматичного перетворення документів до іншої форми представлення (форматери, компілятори, конвертери) чи синтезу документа з окремих частин тощо.

**Інструменти підтримки** процесів виконання програм призначені для виконання на комп'ютері опису процесів або їхніх частин, які представлені у вигляді відмінному від машинної мови.



# Інструменти та інструментальні середовища розробки ПЗ

Програмні середовища можна класифікувати за:

- Орієнтованістю на певну мову програмування;
- Спеціалізованістю;
- Комплексністю;
- Орієнтованістю на конкретну технологію програмування;
- Орієнтованістю на групову розробку;
- Інтегрованістю.



# Комп'ютерні технології розробки ПЗ (CASE-технології)

*Computer-Aided Software Engineering (Інженерія Програмування, що Підтримується Комп'ютером)* — інженерія всього життєвого циклу програмного продукту, характеризується використанням:

- програмної підтримки графічної розробки вимог та специфікації програмного продукту;
- автоматичної генерації програм на певній мові програмування;
- програмної підтримки прототипування.





# Комп'ютерні технології розробки ПЗ (CASE-технології)

**Прототипування ПП** – це етап розроблення програмного продукту, процес створення прототипу програми – макету (чорнової, пробної версії) програми, як правило, з метою перевірки придатності пропонованих для застосування концепцій, архітектурних та/або інших технологічних рішень, а також для представлення програми замовнику на ранніх етапах процесу розроблення. Прототип дозволяє також одержати зворотній зв'язок від майбутніх користувачів, причому, саме тоді, коли це найбільш необхідно: на початку проекту, коли ще є можливість виправити помилки проектування практично без втрат. Прототипування спрямовано на перевірку концепції та мінімізацію ризиків на етапі розроблення програмного забезпечення.

# Комп'ютерні технології розробки ПЗ (CASE-технології)

## Цілі прототипування:

- 1) перевірка концепції та моделювання процесів – прототип дозволяє максимально наблизити бачення майбутньої системи до реального функціонування, включаючи емуляцію робочих процесів з використанням тестових даних, а також оцінити зручність використання;
- 2) керування інвестиціями та мінімізація ризиків – верифікація концепції та деталізація вимог, яка досягається в процесі створення та оцінювання прототип

## Типи прототипування:

- *швидке прототипування* (rapid або throwaway prototyping) – створюється макет, який на певному етапі буде залишений і не стане частиною готової системи.
- *еволюційне прототипування* (evolutionary prototyping) – має за мету послідовно створювати макети системи, які будуть все ближче й ближче до реального продукту

# Комп'ютерні технології розробки ПЗ (CASE-технології)

## Переваги застосування прототипування:

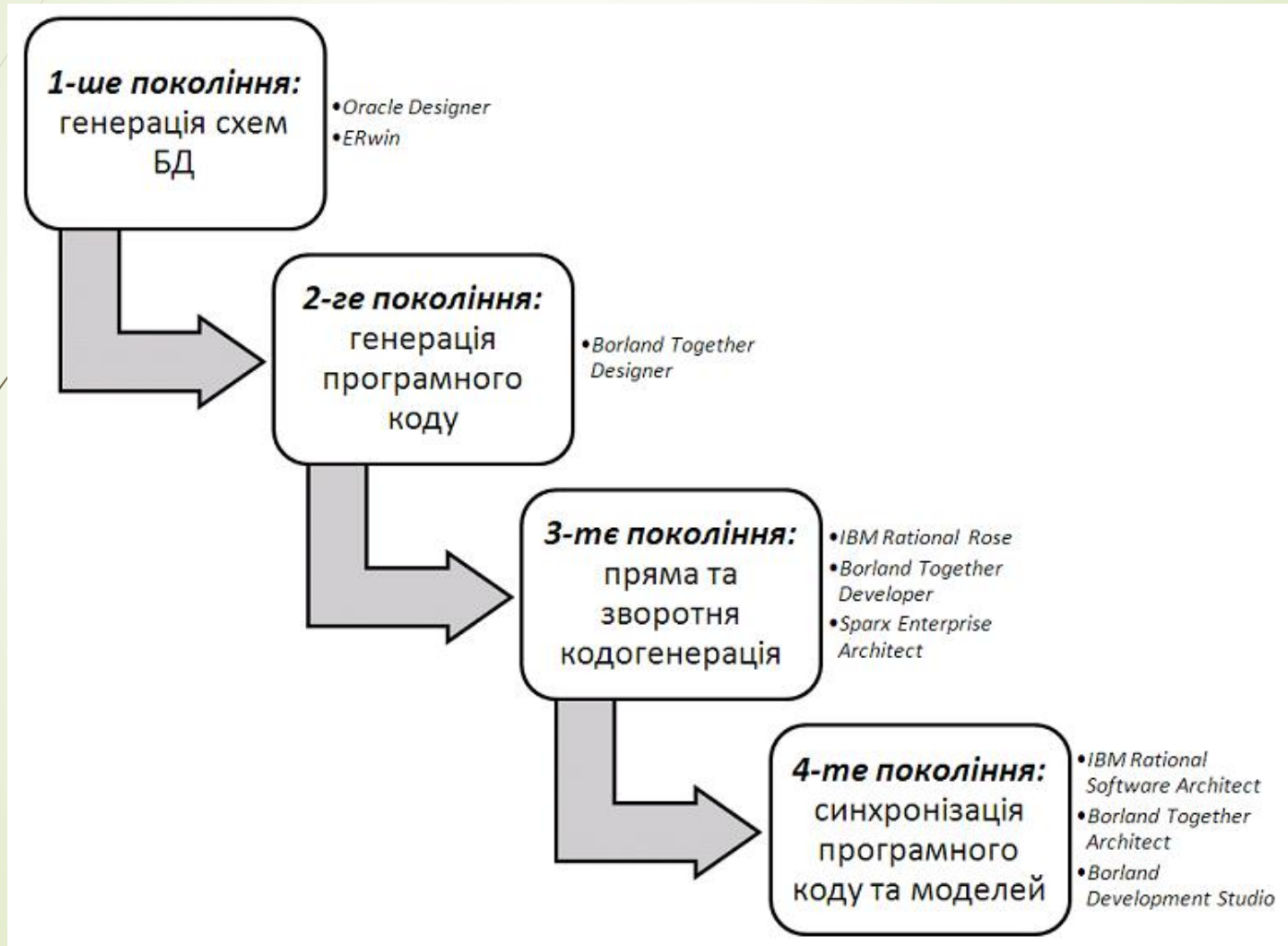
- *зменшення часу, вартості, ризиків*: прототипування покращує якість специфікацій; чим пізніше проводяться зміни у специфікації, тим вони дорожчі, тому уточнення «чого ж замовники хочуть насправді» на ранніх стадіях розроблення знижує загальну вартість;
- *залучення користувача у процес розроблення*: прототипування залучає майбутніх користувачів до процесу розроблення і дозволяє їм бачити те, як саме виглядатиме майбутня програма, що дозволяє позбавитись від можливих розбіжностей в уявленні про програму між розробниками та користувачами.

# Комп'ютерні технології розробки ПЗ (CASE-технології)

## Недоліки застосування прототипування:

- *недостатній аналіз*: концентрація зусиль на обмеженому прототипі може відволікати розробників від необхідного аналізу вимог на повну систему;
- *змішування прототипу та готової системи* в уявленні користувачів: користувачі можуть подумати, що прототип, який пропонується відкинути і є основою майбутньої системи, тому можуть вимагати від прототипу більш точної поведінки або розчаруватись у можливостях розробників;
- *надмірний час на створення прототипу*: ключова властивість прототипу – короткий час його створення; якщо ж розробники не приймають це до уваги, то вони витрачають час на створення надто складного прототипу та втрачають переваги від застосування прототипування взагалі.

# Розвиток CASE-засобів





# Класифікація CASE-засобів

Усі сучасні CASE-засоби можуть бути класифіковані в основному за типами і категоріями.

Класифікація *по типах* відображає функціональну орієнтацію CASE-засобів на ті чи інші процеси життєвого циклу.

Класифікація *по категоріях* визначає ступінь інтегрованості по виконуваних функціях і включає окремі локальні засоби, що вирішують невеликі автономні задачі (*tools*), набір частково інтегрованих засобів, що охоплюють більшість етапів життєвого циклу програмного продукту (*toolkit*) і цілком інтегровані засоби, що підтримують весь життєвий цикл програмного продукту і пов'язані спільним репозиторієм.



# Класифікація CASE-засобів

Крім цього, CASE-засоби можна класифікувати за наступними ознаками:

- методології та моделі систем і баз даних;
- ступінь інтегрованості із СУБД;
- доступні платформи.



# Класифікація по типах

- *засоби аналізу (Upper CASE), призначені для побудови й аналізу моделей предметної галузі (Design/IDEF, BPwin);*
- *засоби аналізу і проектування (Middle CASE), що підтримують найбільш розповсюджені методології проектування і, що використовуються для створення проектних специфікацій (Vantage Team Builder, Silverrun, PRO-IV, CASE.Аналітик);*
- *засоби проектування баз даних, що забезпечують моделювання даних і генерацію схем баз даних (як правило, мовою SQL) для найбільш розповсюджених СУБД. До них відносяться ERwin, S-Designer і DataBase Designer (ORACLE);*
- *засоби розробки додатків. До них відносяться засоби 4GL (Uniface, JAM, PowerBuilder, New Era, SQLWindows, Delphi і ін.) і генератори кодів, що входять до складу Vantage Team Builder, PRO-IV і частково - у Silverrun;*
- *засоби реінжинірінга, що забезпечують аналіз програмних кодів і схем баз даних і формування на їхній основі різних моделей і проектних специфікацій (Засоби аналізу схем БД і формування ERD: Vantage Team Builder, PRO-IV, Silverrun, ERwin і S-Designor; реінжинірінг програм мовою C++: Rational Rose, Object Team).*



# Переваги CASE-технологій

- Єдина графічна мова
- Єдина БД проекту
- Інтеграція засобів
- Підтримка колективної розробки й управління проектом
- Макетування
- Генерація документації
- Верифікація проекту
- Автоматична генерація об'єктного коду
- Супровід і реінжинірінг

# Переваги CASE-технологій

Інтегрований CASE-засіб (чи комплекс засобів, що підтримують повний ЖЦ ПЗ) містить наступні компоненти:

- репозиторій, що є основою CASE-засобу. Він повинен забезпечувати збереження версій проекту і його окремих компонентів, синхронізацію надходження інформації від різних розробників при груповій розробці, контроль метаданих на повноту і несуперечність;
- графічні засоби аналізу і проектування, що забезпечують створення і редагування ієрархічно зв'язаних діаграм (DFD, ERD тощо), що утворюють моделі ІС;
- засоби розробки додатків, включаючи мови 4GL і генератори кодів;
- засоби конфігураційного управління;
- засоби документування;
- засоби тестування;
- засоби управління проектом;
- засоби реінжинірингу.

# ОСНОВНІ ЗМІНИ ЖИТТЄВОГО ЦИКЛУ

Традиційна технологія розробки	Розробка за допомогою CASE-технології
Основні зусилля на <i>кодування і тестування</i>	Основні зусилля на <i>аналіз і проектування</i>
"Паперові" специфікації	Швидке ітеративне макетування
Ручне кодування	Автоматична генерація машинного коду
Тестування ПЗ	Автоматичний контроль проекту
Супровід програмного коду	Супровід проекту

# Основні зміни життєвого циклу

Аналіз	Проектування	Програмування	Тестування
20%	15%	20%	45%
30%	30%	15%	25%
40%	40%	5%	15%

# Структурний аналіз

У структурному аналізі використовуються в основному три групи засобів, що ілюструють функції, виконувані системою і відносини між даними. Кожній групі засобів відповідають певні види моделей (діаграм), найбільш розповсюдженими серед яких є наступні:

- SADT (Structured Analysis and Design Technique) моделі і відповідні функціональні діаграми;
- DFD (Data Flow Diagrams) діаграми потоків даних;
- STD (State Transition Diagrams) діаграми зміни станів;
- ERD (Entity-Relationship Diagrams) діаграми "сутність-зв'язок".





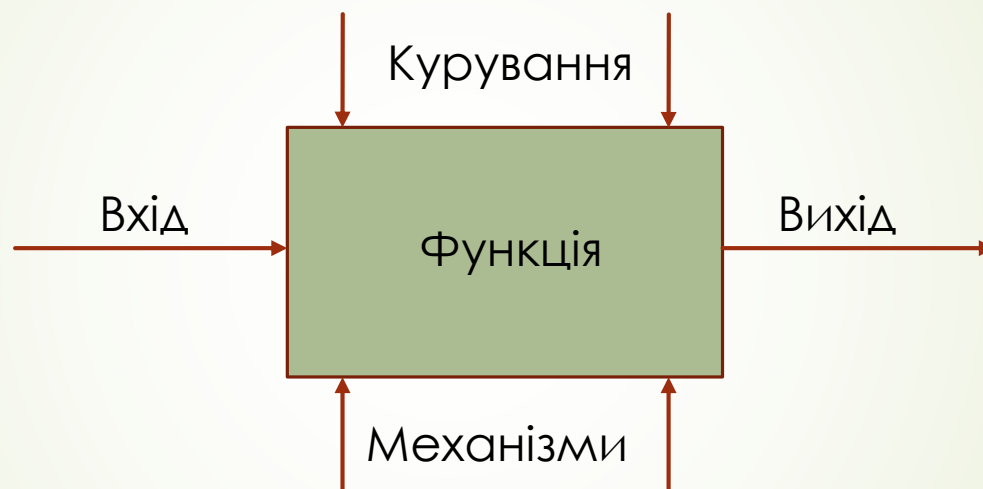
# Структурний аналіз

Метод SADT – це сукупність правил та процедур, призначених для побудови функційної моделі об'єкту будь-якої предметної галузі. Функційна модель SADT відображає функційну структуру об'єкту, тобто дії, які він виконує, та зв'язки між цими діями. Моделі SADT традиційно використовуються для моделювання організаційних систем (бізнес-процесів). Слід зазначити, що метод SADT успішно працює лише при описі добре специфікованих та стандартизованих бізнес-процесів у корпораціях, тому прийнятий у США в якості типового. Перевагами застосування моделей SADT є: повнота опису бізнес-процесу, жорсткі вимоги методу (як результат – моделі стандартного вигляду), відповідність підходу до опису процесів стандартам ISO 9000.



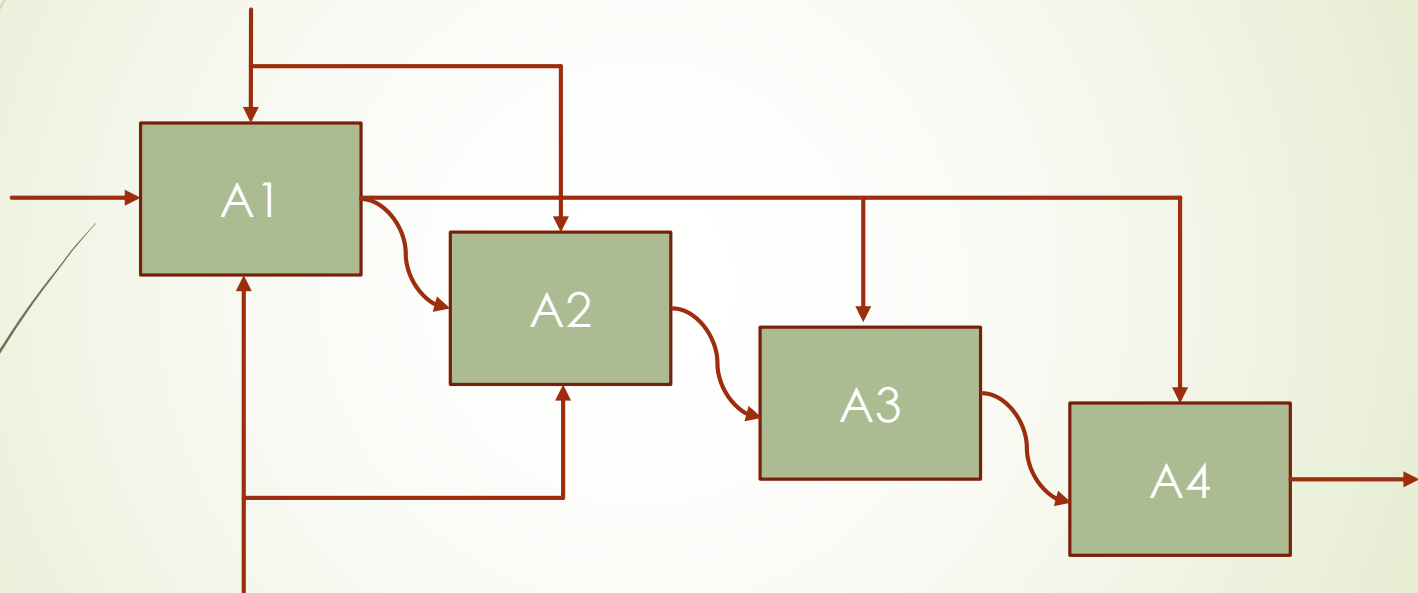
# Структурний аналіз

## *Основа SADT*



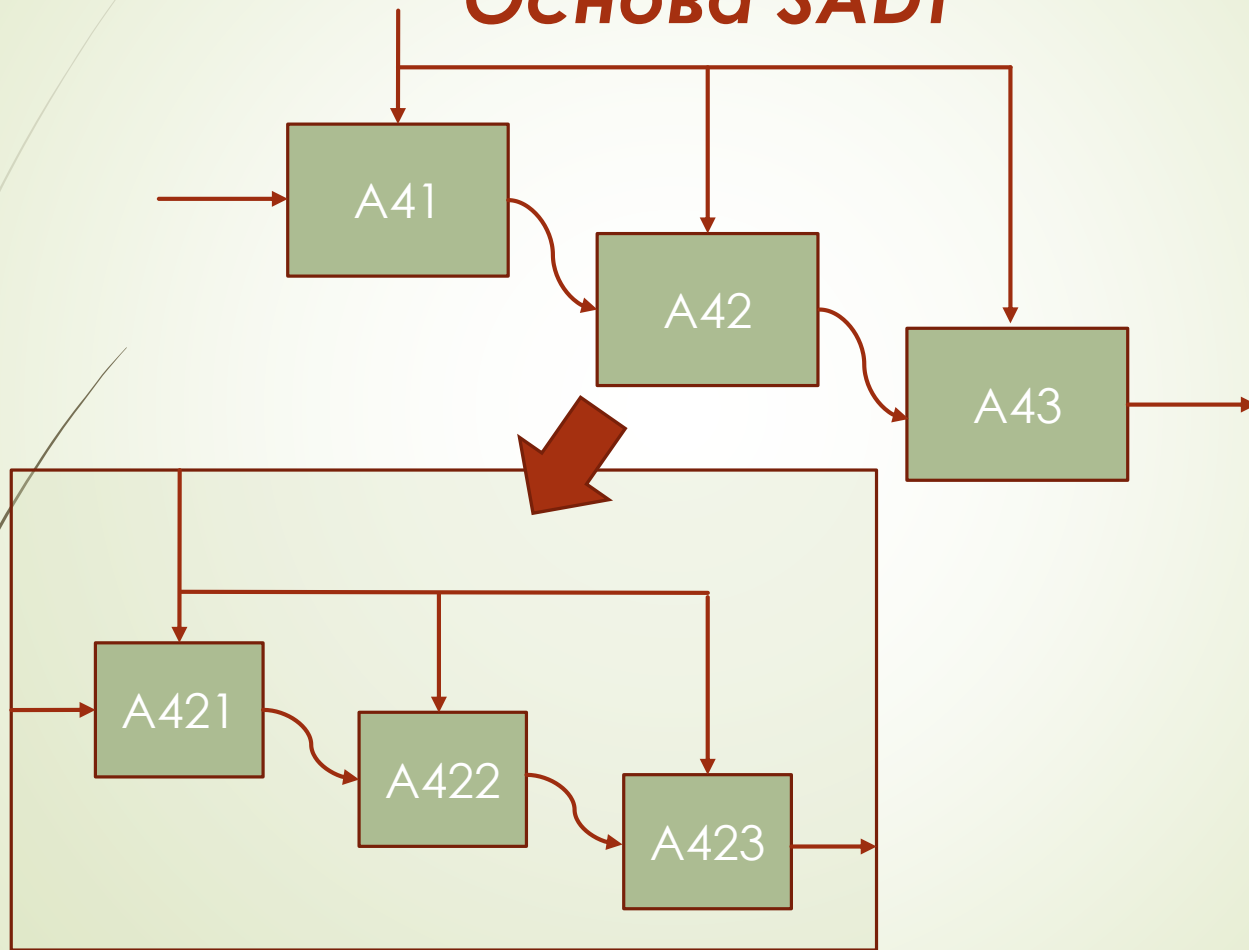
# Структурний аналіз

## *Основа SADT*



# Структурний аналіз

## Основа SADT



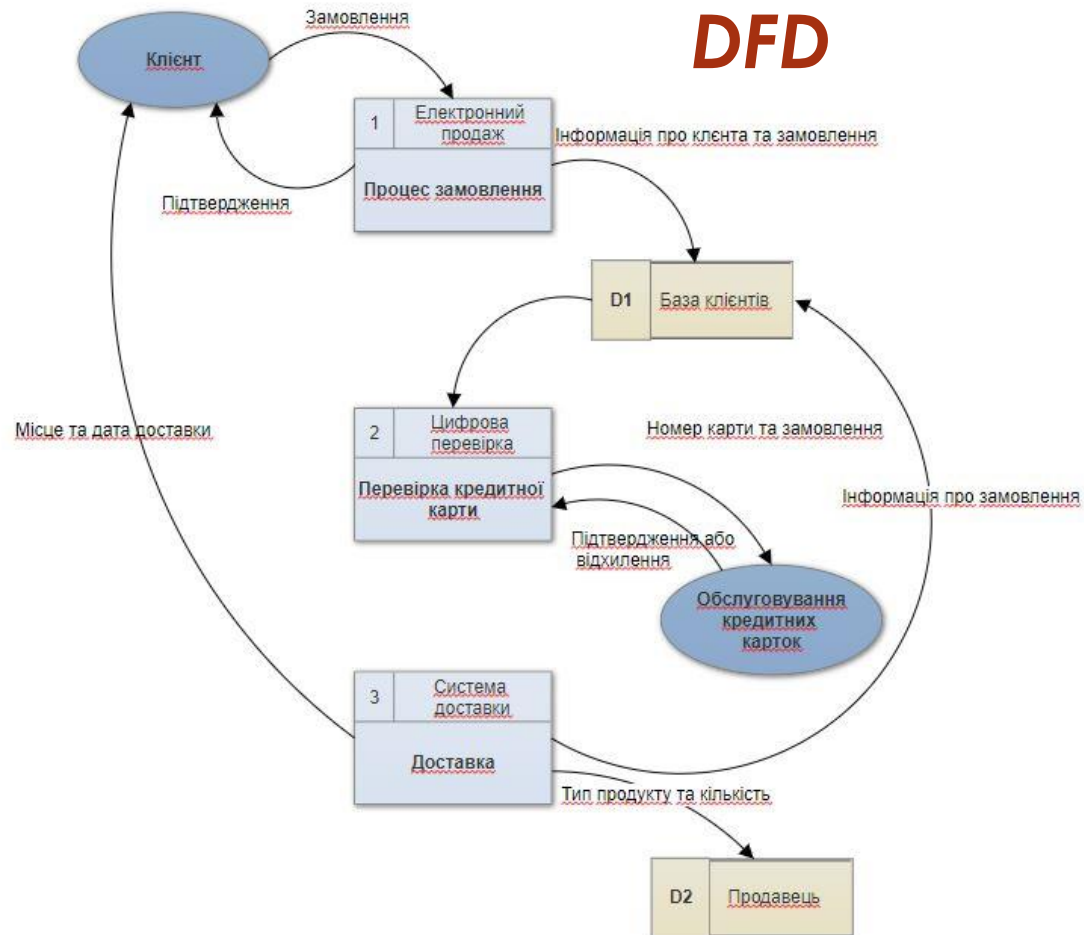
# Засоби функціонального моделювання


Для рішення задачі функціонального моделювання на базі структурного аналізу традиційно застосовуються два типи моделей: SADT-діаграми і діаграми потоків даних.

DFD - показують зовнішні джерела і потоки даних, визначають процеси обробки і потоки даних, ідентифікують сховища даних (накопичувачі).

- DFD із самого початку створювалися як засіб проектування програмних систем і мають більш багатий набір елементів, що адекватно відображають їхню специфіку.
- Наявність міні-специфікацій DFD-процесів нижнього рівня дозволяє перебороти логічну незавершеність SADT і побудувати повну функціональну специфікацію системи.
- Існують алгоритми автоматичного перетворення ієрархії DFD у структурні карти, що демонструють міжмодульні зв'язки, а також ієрархію модулів, що в сукупності з мініспецифікаціями є завершеним завданням для програміста.

# Засоби функціонального моделювання

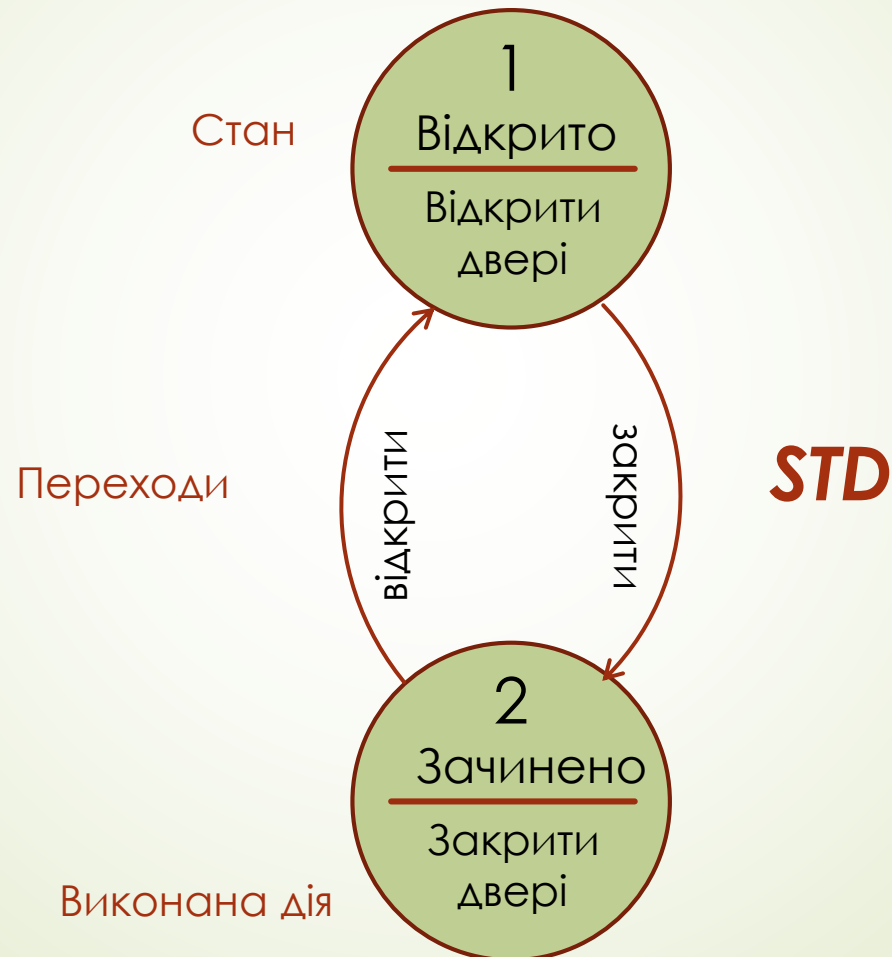





# Засоби подійного моделювання

Найчастіше специфікації управління формалізуються за допомогою діаграм переходів та станів (STD – state transition diagrams), що дозволяють задавати стани різних об'єктів системи (наприклад, особовий рахунок може мати стани ВІДКРИТИЙ, ЗАКРИТИЙ, ЗАБЛОКОВАНИЙ тощо), умови переходів з одного стану в інший (як зовнішній стосовно системи, так і внутрішній, що виникає в самій системі), а також чинені при переходах дії.

# Засоби подійного моделювання







# Засоби інформаційного моделювання

Для цілей інформаційного моделювання на сьогоднішній день не існує альтернативи діаграмам "сутність-зв'язок" (ERD – entity-relationship diagrams).

Вміст накопичувача даних зберігається в Словнику даних і розкривається за допомогою ERD (даної діаграми в основному використовуються при проектуванні БД, зокрема продуктом Logic Works ERWin – засобом для розробки моделей даних). У випадку наявності реального часу DFD доповнюються STD.

# Засоби інформаційного моделювання

