Тема 1. Основні поняття програмних систем

1. Поняття інформаційної системи

Інформаційна система (ІС) – це організований комплекс апаратно-програмних засобів, інформаційних ресурсів, а також персоналу, призначений для забезпечення своєчасного збору, обробки та видачі інформації користувачам системи з метою підтримки їх діяльності.

Інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існувала потреба в управлінні. Місія інформаційної системи полягає у виробництві потрібної для організації інформації, необхідної для ефективного управління всіма її ресурсами, у створенні інформаційного та технічного середовища для управління її діяльністю.

Основні функції інформаційної системи:

- Збір даних
- Зберігання даних
- Обробка даних
- Розподіл інформації
- Представлення інформації у зручному для користувача вигляді

Інформаційна система визначається наступними властивостями:

- Будь-яка інформаційна система може бути підсистемою більшої системи
- Всі елементи інформаційної системи взаємопов'язані
- Інформаційна система володіє інтегративними властивостями
- Інформаційна система має певні інтерфейси, через які вона взаємодіє із зовнішнім середовищем

2. Класифікація інформаційних систем

Інформаційні системи можна класифікувати за різними критеріями:

За масштабом:

- **Одиночні** – працюють на одному комп'ютері і призначені для одного користувача
- **Групові** – орієнтовані на колективне використання інформації членами робочої групи
- **Корпоративні** – призначені для великих організацій, мають ієрархічну структуру

За сферою застосування:

- **Системи обробки транзакцій (TPS)** – автоматизують рутинні операції

- **Системи підтримки прийняття рішень (DSS)** – забезпечують інформаційну підтримку прийняття рішень
- **Інформаційно-довідкові системи** – зберігають і надають доступ до структурованої інформації
- **Офісні інформаційні системи** – автоматизують діловодство
- **Системи управління підприємством (ERP)** – комплексно автоматизують всі бізнес-процеси підприємства
- **Системи управління взаєминами з клієнтами (CRM)** – автоматизують процеси взаємодії з клієнтами

За архітектурою:

- **Локальні** – всі компоненти системи розташовані на одному комп'ютері
- **Клієнт-серверні** – обробка даних розподілена між клієнтами та серверами
- **Багаторівневі** – функціональні частини розподілені між кількома комп'ютерами
- **Розподілені** – обробка розподілена між декількома комп'ютерами, об'єднаними мережею

За ступенем автоматизації:

- **Автоматизовані** – передбачають участь людини в процесі функціонування
- **Автоматичні** – функціонують без участі людини

За характером обробки даних:

- **Інформаційно-пошукові** – призначені для пошуку і видачі інформації
- **Інформаційно-вирішальні** – здійснюють обробку інформації за певними алгоритмами

3. Основні компоненти інформаційної системи

Інформаційна система складається з наступних основних компонентів:

Апаратне забезпечення (Hardware)

Включає комп'ютери та їх периферійні пристрої: дисководи, монітори, принтери, сканери, мережеве обладнання тощо. Технічна платформа є фундаментом інформаційної системи.

Програмне забезпечення (Software)

Системні та прикладні програми, які забезпечують функціонування апаратної складової і реалізацію прикладних функцій системи. Включає:

- Системне ПЗ (операційні системи, СУБД, мережеве ПЗ)
- Прикладне ПЗ (бізнес-додатки, утиліти, інструментальні засоби)

Дані (Data)

Інформація, структурована певним чином і організована для зберігання та обробки. Дані є моделлю предметної області:

- Бази даних
- Сховища даних
- Репозиторії
- Файлові системи

Процедури (Procedures)

Формалізовані інструкції, методики та правила виконання завдань для персоналу та користувачів системи:

- Регламенти обробки інформації
- Інструкції користувачів
- Правила роботи з системою
- Документація

Персонал (People)

Люди, які працюють із системою:

- Розробники
- Адміністратори
- Користувачі
- Обслуговуючий персонал

Мережі та комунікації (Networks)

Засоби взаємодії компонентів системи та зв'язку із зовнішнім середовищем:

- Локальні мережі
- Глобальні мережі
- Телекомунікаційне обладнання
- Протоколи передачі даних

4. Складність програмних систем

Складність є фундаментальною властивістю програмних систем, яка впливає на всі аспекти їх розробки, впровадження та експлуатації. Програмні системи належать до найскладніших штучних систем, які створює людина.

Основні причини складності програмних систем:

1. **Нематеріальність програмного забезпечення** - на відміну від матеріальних об'єктів, програмне забезпечення неможливо "побачити" та "відчути" безпосередньо, що ускладнює його сприйняття та розуміння.
2. **Масштабність** - сучасні програмні системи можуть складатися з мільйонів рядків коду, тисяч взаємопов'язаних компонентів.
3. **Високий рівень абстракції** - програмне забезпечення оперує складними концептуальними конструкціями.
4. **Велика кількість станів системи** - складна програмна система може мати астрономічну кількість можливих станів, які неможливо повністю протестувати.
5. **Динамічна природа** - програмні системи постійно змінюються та еволюціонують протягом свого життєвого циклу.
6. **Різноманіття технологій** - при розробці використовуються різні мови програмування, фреймворки, бібліотеки, що збільшує гетерогенність системи.
7. **Інтеграція з іншими системами** - взаємодія з зовнішніми системами додає складність.
8. **Людський фактор** - різні розробники мають різні стилі програмування, рівень кваліфікації, розуміння предметної області.

5. Види складності програмних систем

Структурна складність

Відображає складність внутрішньої організації системи: кількість компонентів, їх взаємозв'язки, ієрархічність структури. Структурна складність виникає через велику кількість елементів та зв'язків між ними. Чим більше компонентів має система і чим складніше вони взаємодіють, тим вища її структурна складність.

Алгоритмічна складність

Пов'язана зі складністю алгоритмів, що реалізують функціональність системи. Характеризується часовою та просторовою складністю алгоритмів, їх обчислювальною ефективністю. Алгоритмічна складність оцінюється в термінах теорії складності обчислень (O-нотація).

Логічна складність

Відображає складність логіки роботи системи, складність бізнес-правил та умов, що обробляються програмним забезпеченням. Висока логічна складність призводить до великої кількості розгалужень у програмі, складних умовних конструкцій, нетривіальних залежностей.

Інформаційна складність

Пов'язана з обсягом та структурою даних, що обробляються системою. Включає складність схеми бази даних, структур даних, обсяги інформації, різноманітність форматів даних.

Динамічна складність

Характеризує складність поведінки системи в часі, особливості функціонування в різних режимах, реакції на зовнішні впливи. Динамічна складність включає аспекти паралелізму, асинхронної взаємодії, реакції на події.

Інтерфейсна складність

Відображає складність взаємодії системи з користувачами та іншими системами. Включає складність користувацького інтерфейсу, API, протоколів обміну даними.

Технологічна складність

Пов'язана з різноманіттям використовуваних технологій, платформ, мов програмування, інструментів. Підвищується при необхідності інтеграції різнорідних технологічних рішень.

Організаційна складність

Відображає складність процесів розробки, впровадження та супроводу системи, включаючи управління командою, координацію робіт, комунікації між учасниками проекту.

6. Методи оцінки складності програмних систем

Метрики розміру програми

- **Кількість рядків коду (LOC, SLOC)** - найпростіша метрика, яка відображає обсяг програмного коду.
- **Функціональні точки (Function Points)** - вимірюють функціональність, яку надає система, незалежно від технологій реалізації.
- **Точки історій користувачів (Story Points)** - відносна міра складності функціональності в гнучких методологіях.

Метрики структурної складності

- **Цикломатична складність (McCabe)** - вимірює кількість лінійно незалежних шляхів через програму, відображаючи складність потоку керування.
- **Глибина вкладеності** - оцінює рівень вкладеності блоків коду.
- **Зв'язність (cohesion) та зчеплення (coupling)** - оцінюють якість модульної структури системи.
- **Метрики залежностей** - аналізують граф залежностей між компонентами системи.

Об'єктно-орієнтовані метрики

- **Метрики Чидамбера-Кемерера (CK)** - набір метрик для оцінки ОО-дизайну, включаючи:
    - Зважені методи на клас (WMC)

    - Глибина дерева наслідування (DIT)
    - Кількість нащадків (NOC)
    - Зв'язність між класами (CBO)
    - Відсутність зв'язності методів (LCOM)
- **Метрики MOOD** - набір метрик на рівні системи, що оцінюють інкапсуляцію, наслідування, поліморфізм.

Когнітивні метрики

- **Когнітивна складність** - оцінює зусилля, необхідні для розуміння коду.
- **Читабельність коду** - оцінює легкість сприйняття коду людиною.

Динамічні метрики

- **Профілювання продуктивності** - аналіз використання ресурсів під час виконання.
- **Аналіз покриття коду** - оцінка повноти тестування.
- **Аналіз часових характеристик** - оцінка швидкодії та відгуку системи.

Інтегральні методи оцінки

- **Аналіз технічного боргу** - оцінка "вартості" недосконалостей дизайну та реалізації.
- **Моделі оцінки трудовитрат (COCOMO, COCOMO II)** - оцінюють зусилля, необхідні для розробки системи певної складності.
- **Моделі надійності програмного забезпечення** - оцінюють ймовірність відмов залежно від складності системи.

7. Необхідність інженерії програмних систем

Інженерія програмних систем виникла як відповідь на "програмну кризу" 1960-х років, коли стало очевидно, що традиційні підходи до розробки не справляються зі зростаючою складністю програмних систем. З того часу необхідність застосування інженерних підходів до створення програмного забезпечення лише посилювалася.

Основні причини необхідності інженерії програмних систем:

1. **Зростаюча складність програмних систем** Сучасні програмні системи досягли такого рівня складності, що їх неможливо розробляти, підтримувати та розвивати без застосування інженерних методів. Інтуїтивні підходи та "програмування на коліні" не працюють для великих систем.
2. **Високі вимоги до якості** Програмне забезпечення стало критично важливим компонентом багатьох систем, від яких залежить безпека, здоров'я та навіть життя людей. Це ставить високі вимоги до надійності, безпеки та інших атрибутів якості ПЗ.

3. **Економічні фактори** Розробка програмного забезпечення потребує значних інвестицій. Інженерні методи дозволяють оптимізувати використання ресурсів, зменшити ризики та збільшити віддачу від інвестицій.
4. **Необхідність управління змінами** Програмне забезпечення постійно еволюціонує. Інженерні підходи забезпечують контрольоване внесення змін без порушення цілісності системи.
5. **Масштабність команд розробників** Великі програмні системи розробляються командами, що можуть налічувати сотні фахівців. Ефективна координація їх роботи неможлива без чітких інженерних процесів.
6. **Повторне використання компонентів** Інженерний підхід сприяє створенню модульних, повторно використовуваних компонентів, що підвищує ефективність розробки.
7. **Складність інтеграції з іншими системами** Сучасні програмні системи рідко функціонують ізольовано. Вони взаємодіють з багатьма іншими системами, що вимагає стандартизованих інтерфейсів та протоколів.
8. **Необхідність документування** Програмні системи супроводжуються протягом багатьох років, часто різними командами. Без документації, створеної за інженерними стандартами, такий супровід стає неможливим.

8. Мета і завдання інженерії програмних систем

Мета інженерії програмних систем

Основна мета інженерії програмних систем – забезпечення систематичного, дисциплінованого, кількісно вимірюваного підходу до розробки, експлуатації та супроводу програмного забезпечення, а також вивчення підходів до цих видів діяльності.

Інженерія програмних систем спрямована на створення якісного програмного забезпечення, яке:

- Задовольняє потреби користувачів
- Доставляється вчасно і в межах бюджету
- Може ефективно розвиватися та супроводжуватися

Завдання інженерії програмних систем

1. **Формалізація процесів розробки ПЗ**
    - Впровадження стандартизованих методологій і процесів
    - Забезпечення повторюваності та передбачуваності результатів
    - Створення формальних моделей та специфікацій
2. **Забезпечення якості програмного забезпечення**

- Впровадження стандартів якості
- Розробка та застосування методів верифікації та валідації
- Тестування на всіх етапах життєвого циклу
- Забезпечення надійності, безпеки, продуктивності та інших атрибутів якості

3. **Управління складністю**
   - Декомпозиція складних систем на керовані компоненти
   - Застосування принципів модульності, абстракції, інкапсуляції
   - Використання архітектурних патернів та шаблонів проектування
4. **Оптимізація використання ресурсів**
   - Планування та контроль ресурсів проекту
   - Оцінка трудовитрат та термінів
   - Оптимізація процесів розробки
5. **Забезпечення простоти супроводу та розвитку**
   - Розробка з урахуванням майбутніх змін
   - Створення підтримуваної архітектури
   - Розробка технічної документації
6. **Стандартизація та уніфікація**
   - Впровадження галузевих стандартів
   - Використання стандартних інтерфейсів та протоколів
   - Забезпечення сумісності компонентів
7. **Управління ризиками**
   - Ідентифікація та аналіз ризиків
   - Розробка стратегій зниження ризиків
   - Моніторинг та контроль ризиків протягом життєвого циклу
8. **Управління знаннями**
   - Накопичення та передача досвіду
   - Створення бази знань та найкращих практик
   - Навчання та розвиток персоналу

9. Переваги використання інженерії програмних систем

Для бізнесу та організацій

1. **Підвищення передбачуваності результатів**
    - Точніші оцінки термінів та вартості
    - Зниження ризиків проекту
    - Більш реалістичне планування
2. **Оптимізація використання ресурсів**
    - Ефективніше використання людських ресурсів
    - Зниження вартості розробки
    - Оптимізація використання технічних ресурсів
3. **Підвищення якості продукту**
    - Відповідність вимогам користувачів
    - Надійність та стабільність роботи
    - Безпека та захищеність
4. **Скорочення часу виведення продукту на ринок**
    - Прискорення розробки за рахунок використання готових компонентів
    - Паралельна робота над різними частинами системи
    - Автоматизація рутинних процесів
5. **Підвищення конкурентоспроможності**
    - Створення продуктів вищої якості
    - Можливість швидше реагувати на зміни ринку
    - Зниження вартості володіння програмним забезпеченням

Для команди розробників

1. **Спрощення комунікації**
    - Єдина термінологія та нотація
    - Чіткі ролі та відповідальність
    - Стандартизовані артефакти
2. **Підвищення ефективності роботи**
    - Чіткі процеси та методологія
    - Автоматизація рутинних завдань
    - Повторне використання компонентів та шаблонів
3. **Зниження стресу та вигорання**

- Структуровані процеси знижують хаотичність роботи
- Реалістичні оцінки зменшують надмірне навантаження
- Чіткі критерії якості знижують кількість переробок

4. **Професійний розвиток**
    - Застосування сучасних методів та інструментів
    - Формалізація та передача знань
    - Спеціалізація та поглиблення експертизи

Для кінцевих користувачів

1. **Вища якість програмного забезпечення**
    - Відповідність функціональним вимогам
    - Надійність та стабільність роботи
    - Зручність використання
2. **Більш швидке отримання функціональності**
    - Інкрементальна розробка та часті релізи
    - Пріоритизація функцій за цінністю для користувача
    - Швидше виправлення дефектів
3. **Краща підтримка та супровід**
    - Документований продукт
    - Структурований процес підтримки
    - Зрозумілі процедури оновлення

10. Поняття проекту

Проект – це тимчасове підприємство, спрямоване на створення унікального продукту, послуги або результату, з визначеним початком та кінцем у часі, а також з визначеним обсягом робіт та ресурсів.

У контексті програмної інженерії, **програмний проект** – це скоординований набір дій, спрямованих на розробку програмної системи відповідно до заданих вимог, з обмеженнями за часом, бюджетом та ресурсами.

Ключові характеристики проекту:

1. **Унікальність** – кожен проект створює унікальний результат, навіть якщо використовуються типові процеси.

2. **Тимчасовість** – проект має чітко визначені початок і кінець. Проект закінчується, коли досягнуті його цілі, або коли стає ясно, що цілі не можуть бути досягнуті, або коли потреба в проекті більше не існує.
3. **Обмеженість ресурсів** – проект має обмежений бюджет, персонал, обладнання та інші ресурси.
4. **Поступова деталізація** – проекти розвиваються поетапно. На початку проекту інформація про нього часто є неповною і уточнюється в процесі виконання.
5. **Наявність замовника або спонсора** – проект завжди має замовника або спонсора, який визначає цілі та надає ресурси.

Основні атрибути проекту:

1. **Цілі** – чітко визначені результати, яких потрібно досягти.
2. **Вимоги** – умови та обмеження, яким повинен відповідати результат проекту.
3. **Обсяг робіт** – перелік усіх робіт, які необхідно виконати для досягнення цілей проекту.
4. **Ресурси** – люди, обладнання, матеріали, необхідні для виконання проекту.
5. **Бюджет** – фінансові ресурси, виділені на проект.
6. **Графік** – розклад виконання робіт з термінами та контрольними точками.
7. **Ризики** – потенційні події, які можуть вплинути на хід проекту.

Учасники проекту:

1. **Спонсор проекту** – особа або група, яка забезпечує фінансові ресурси для проекту.
2. **Замовник** – особа або організація, яка ініціює проект і буде використовувати його результати.
3. **Керівник проекту** – особа, відповідальна за планування, виконання та завершення проекту.
4. **Команда проекту** – група фахівців, які виконують роботи проекту.
5. **Зацікавлені сторони** – особи або організації, які активно залучені до проекту або інтереси яких можуть бути порушені в результаті виконання або завершення проекту.

11. Основні етапи життєвого циклу проекту

Життєвий цикл проекту – це послідовність фаз, через які проходить проект від його початку до завершення. Хоча конкретні фази можуть відрізнятися залежно від методології та специфіки проекту, можна виділити типові етапи, характерні для більшості проектів розробки програмного забезпечення.

1. Ініціація проекту

На цьому етапі визначається необхідність та доцільність проекту, формується його бізнес-обґрунтування.

Основні активності:

- Визначення цілей та результатів проекту
- Аналіз економічної доцільності
- Ідентифікація зацікавлених сторін
- Призначення керівника проекту
- Розробка статуту проекту

Артефакти:

- Статут проекту
- Бізнес-кейс
- Реєстр зацікавлених сторін
- Документ про ініціацію проекту

2. Планування проекту

На цьому етапі розробляється детальний план реалізації проекту, включаючи обсяг робіт, терміни, ресурси та ризики.

Основні активності:

- Збір вимог
- Створення ієрархічної структури робіт (WBS)
- Розробка графіка проекту
- Планування ресурсів
- Оцінка вартості
- Ідентифікація та аналіз ризиків
- Планування якості
- Планування комунікацій

Артефакти:

- План управління проектом
- Документація вимог
- Структура декомпозиції робіт
- Графік проекту
- Бюджет проекту

- Реєстр ризиків
- План управління якістю
- План комунікацій

3. Виконання проекту

На цьому етапі виконуються заплановані роботи для досягнення цілей проекту.

Основні активності:

- Координація людей та ресурсів
- Виконання запланованих робіт
- Управління зацікавленими сторонами
- Забезпечення якості
- Управління змінами

Артефакти:

- Результати робіт
- Протоколи зустрічей
- Запити на зміни
- Звіти про статус проекту

У контексті розробки програмного забезпечення, на цьому етапі виконуються такі специфічні активності:

- Аналіз та проектування
- Написання коду
- Тестування компонентів
- Інтеграція компонентів
- Системне тестування
- Створення документації

4. Моніторинг та контроль

Цей етап є паралельним до етапу виконання і забезпечує відстеження, перегляд та регулювання ходу та ефективності проекту.

Основні активності:

- Моніторинг виконання робіт
- Контроль дотримання графіка
- Контроль витрат

- Контроль якості
- Управління ризиками
- Управління змінами

Артефакти:

- Звіти про виконання
- Запити на зміни
- Оновлені плани проекту
- Протоколи перевірок якості

5. Завершення проекту

На цьому етапі проект формально закривається, проводиться оцінка результатів та документування досвіду.

Основні активності:

- Передача результатів замовнику
- Отримання формального приймання результатів
- Підготовка заключної документації
- Вивільнення ресурсів
- Аналіз досвіду проекту

Артефакти:

- Акт приймання-передачі
- Заключний звіт
- Архів проекту
- Документація щодо отриманого досвіду

Особливості життєвого циклу в різних методологіях

Класичний (каскадний) підхід:

- Послідовне виконання етапів
- Перехід до наступного етапу тільки після завершення попереднього
- Фіксований обсяг робіт, визначений на початку проекту

Ітеративний підхід:

- Цикли повторення (ітерації) етапів
- Поступове уточнення вимог та рішень
- Можливість внесення змін між ітераціями

Гнучкий (Agile) підхід:

- Короткі ітерації (спринти)
- Постійна взаємодія з замовником
- Адаптивне планування
- Поступова розробка функціональності з найвищою цінністю

DevOps підхід:

- Інтеграція розробки та експлуатації
- Автоматизація процесів інтеграції та доставки
- Безперервне тестування та розгортання

Незалежно від обраної методології, розуміння етапів життєвого циклу проекту допомагає ефективно управляти проектом та забезпечувати досягнення його цілей.