

# Технологія Програмування та Створення Програмних Продуктів.

## Лекція 2.

### Інженерія програмного забезпечення

# Зміст

- Основні поняття програмної інженерії
- Цілі діяльності програмних інженерів
- Шляхи забезпечення надійності розроблення ПЗ
- Інженерія вимог до ПЗ
- Модель зрілості процесу конструювання ПЗ (СММ)
- Література

# Базові дисципліни

- ✧ “Інформаційні технології”
  - Знання основних засад в області ІТ
- ✧ “Основи програмування”
  - Знання основних принципів розроблення програм
- ✧ “Дискретна математика”
  - Знання математичних операцій
- ✧ “Алгоритми і структури даних”
  - Вміння оперувати даними з допомогою алгоритмів
- ✧ “Об’єктно-орієнтоване програмування”
  - Розуміння об’єктного підходу до програмування

# Основні поняття програмної інженерії

**Програмна інженерія – розділ комп’ютерної науки, який :**

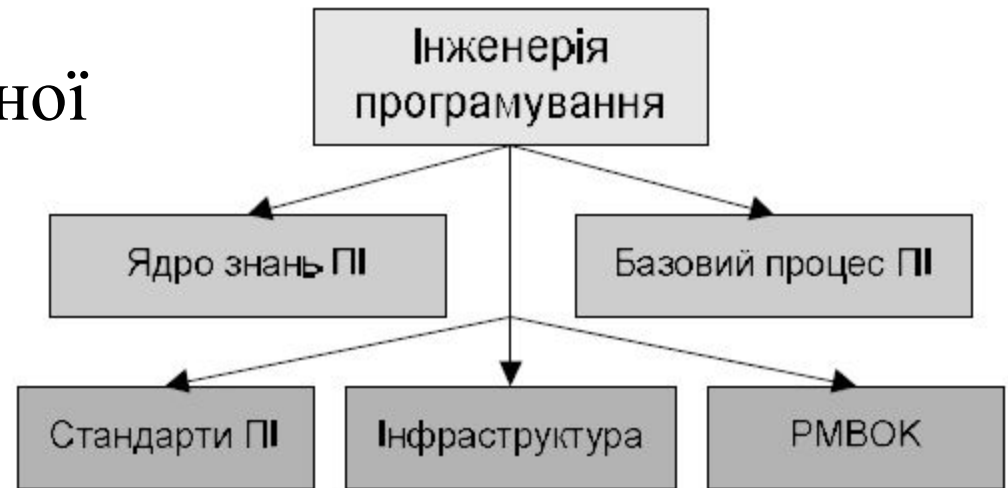
1. вивчає методи і засоби побудови комп’ютерних програм; відображає закономірності розвитку і узагальнює накопичений досвід програмування;
2. оперує об’єктами (модулями, компонентами, програмними аспектами тощо) та визначає автоматизовані операції щодо їх виробництва;
3. виробляє правила і порядок інженерної діяльності і керування технологічним процесом побудови з простих об’єктів нових, більш складних, об’єктів (ПЗ, ПС, ПП, програмних проектів тощо), а також методи виміру й оцінки готового продукту

**Область дії програмної інженерії :**

- *Computer Science* – описує теорію і основи розроблення ПЗ
- *System Engineering* – розглядають питання розробки систем з залученням комп’ютерних засобів
- *Software Engineering* – частина системної інженерії, що включає розроблення ПЗ

# ПІ - інженерна дисципліна

Базові складові інженерної дисципліни:



- 1) **ядро знань SWEBOOK** - набір теоретичних концепцій і формальних визначень методів і засобів розробки та керування програмними проектами;
- 2) **базовий процес ПІ** - стрижень процесної діяльності в організації-розробнику ПП;
- 3) **стандарти** - набір регламентованих правил конструювання проміжних артефактів у процесах ЖЦ;
- 4) **інфраструктура** – умови середовища та методичне забезпечення базового процесу ПІ і підтримка дій його виконавців, що займаються виробництвом ПП;
- 5) **менеджмент проекту (PMBOK)** – ядро знань для керування проектами – набір стандартних процесів, принципів і методів планування і управління проектом.

# Програмна інженерія як дисципліна

- **Програмна інженерія** – інженерна дисципліна, зв'язана з теорією, методами і засобами професійної розробки ПЗ
- Відомо, що:
  - ПЗ = програми + вся супутня документація
  - Висока вартість розробки ПЗ (вища, ніж для апаратури)
  - Вартість розробки ПЗ постійно зростає
- Програмна інженерія допомагає вирішити **проблему зростання вартості розроблення ПЗ**
- Програмна інженерія має справу з усіма аспектами створення ПЗ

# Область програмної інженерії

- Отже, **computer science** представляє теоретичний базис. На практиці його недостатньо. Залишаються проблеми:
  - Пошук фінансування.
  - Робота з замовником.
  - Підбір кадрів і персоналу.
  - Етичні питання. Мікроклімат в колективі. Команда.
  - Забезпечення якості програмного продукту.
  - ...
- Всім цим займається **програмна інженерія.**

# Цілі діяльності програмних інженерів

- Створити якісний програмний продукт
  - Функціональність
  - Надійність
  - Легкість застосування
  - Ефективність
  - Легкість супроводу
  - Мобільність
- Вкластися в бюджет проекту
  - 60% розроблення ПЗ
  - 40% тестування ПП
- Вкластися у заплановані терміни
  - Грамотне планування
  - Аналіз ризиків
  - Межі проекту
  - Мотивування співробітників



# Якісний Програмний Продукт

## 1. Вимоги замовника:

Many programs simply don't do what end users want.

Typical percentages for large-scale commissioned systems:

- 45% delivered but not used
- 27% paid for but not delivered
- 17% abandoned
- 6% used after changes
- 5% used as delivered

Users find it hard to articulate what they want.

Developers find it hard to understand what users say!

## 2. Висока надійність:

Mistakes in programs are generically known as bugs.

A crucial lesson: You can prove that bugs are there; you can't prove that they aren't.

Bugs can be expensive, in terms of. . .

- **human lives:** in safety critical systems, e.g., nuclear reactor control, fly-by-wire aircraft
- **money:** software bug in failed Ariane 5 launch cost US\$500 million
- **poor customer relations:** Microsoft problems with original Windows release caused the company huge problems.

# Поняття "якості" ПП

- ***це сукупність його рис і характеристик, які впливають на здатність задовольняти задані потреби користувачів***
- Критерії якості ПЗ:
  - функціональність \*
  - надійність \*
  - легкість застосування
  - ефективність
  - супроводжуваність
  - мобільність

\* Функціональність і надійність є обов'язковими критеріями якості ПЗ

# Забезпечення надійності ПЗ

- Боротьба зі складністю
- Точність інтерпритації документів
- Подолання бар'єру між розробником і користувачем ПП
- Контроль ухвалюваних рішень
- Взаємодія програмних інженерів з науковими розробками

# Вчасне завершення розробки ПП

- Відомо, що програмні проекти часто перевищують часові рамки
- Надзвичайно важко достовірно прогнозувати, скільки ресурсів потрібно на реалізацію ПП, і коли проект буде завершено
- Співвідношення між наявними людино-місяцями та тривалістю ІТ проекту майже ніколи не буває лінійним :
  - **добавляння людино-місяців до діючого проекту часто взагалі не дає ніякого ефекту;**
  - **добавляння людино-місяців до проблемного проекту часто призводить до сповільнення проекту.**

# Складність програмної системи

- М. Холстед (1977) запропонував міру довжини модуля:  $N \approx n_1 \log_2(n_1) + n_2 \log_2(n_2)$
- другу метрику М. Холстед розглядає об'єм  $V$  модуля:  $V = N \times \log_2(n_1 + n_2)$
- Том Маккейб (1976) розробив метрику цикломатичної складності:  $V(G) = E - N + 2$

## **Методи боротьби зі складністю**

- забезпечення незалежності компонентів системи
- використання в системах ієрархічних структур

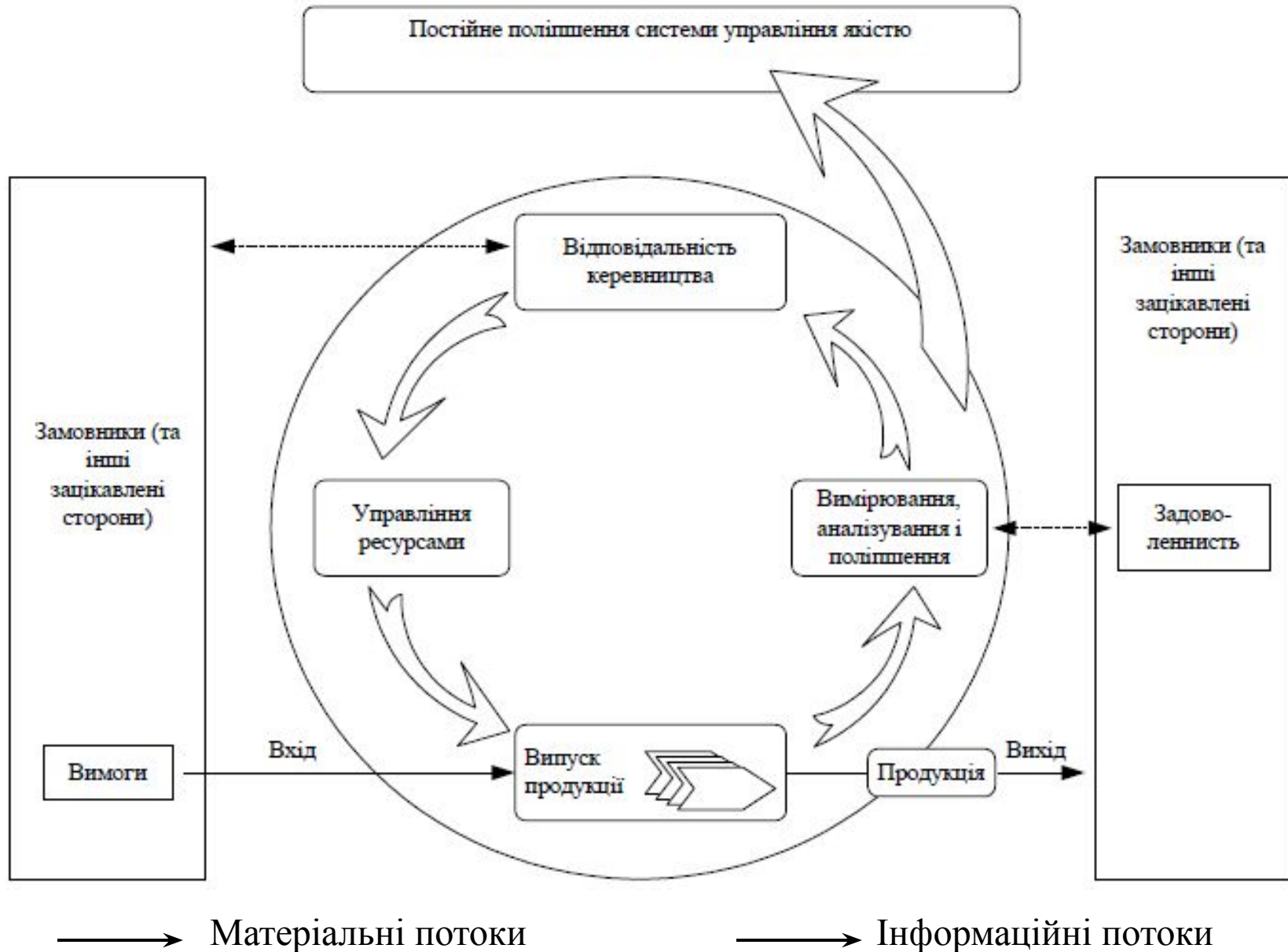
# Моделі якості розроблення ПП

- Стандарти ISO 9001:2000 (ДСТУ ISO 9001-2001)
- Capability Maturity Model (CMM)

***Стандарти ISO 9001:2000*** (міжнародної організації з стандартизації) – це стандарти, що містять вимоги до систем управління якістю, спрямовані на забезпечення якості і підвищення задоволеності споживача

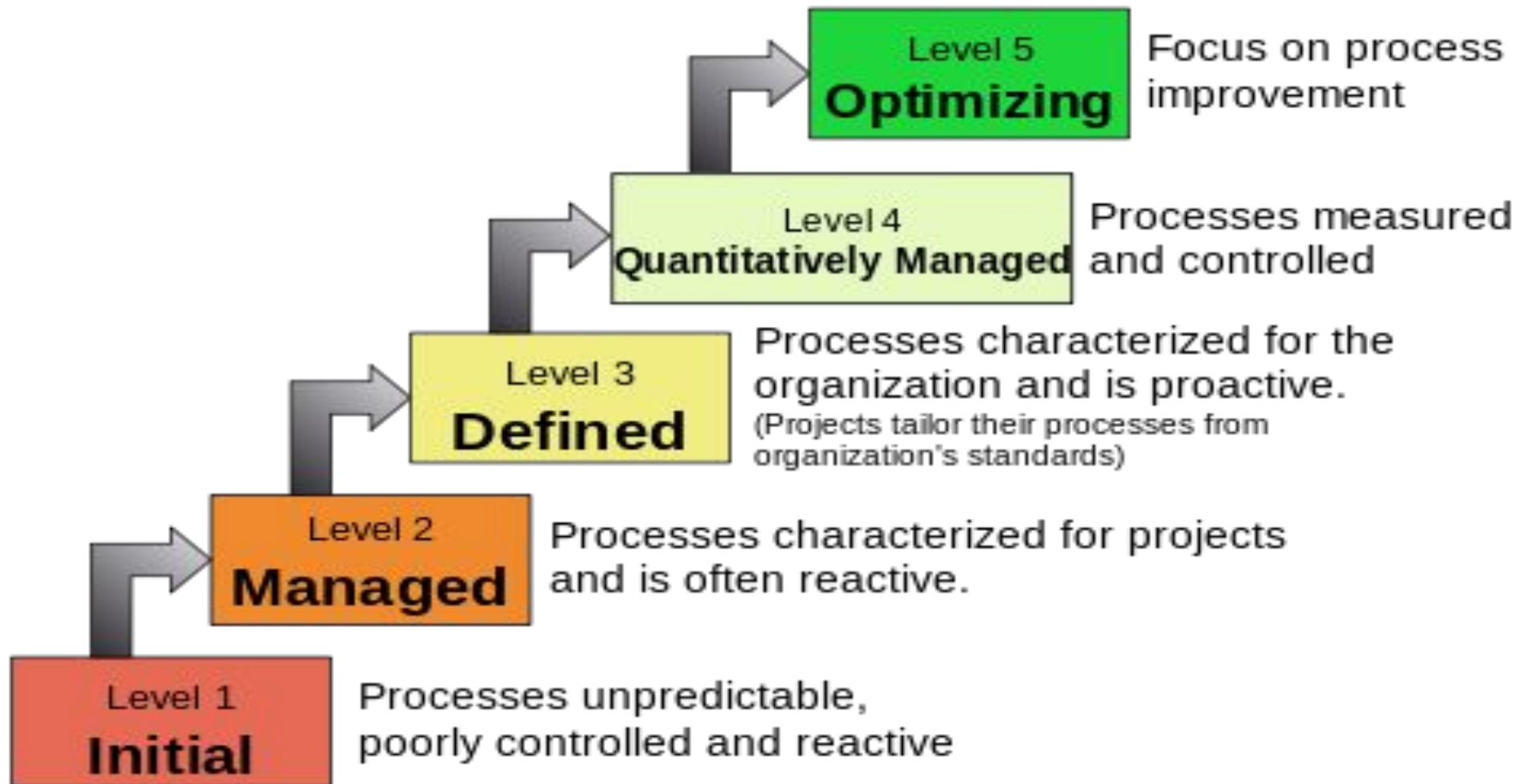
***Capability Maturity Model*** (модель технологічної зрілості) – це еволюційна модель, що описує розвиток здатності компанії розробляти якісне програмне забезпечення

# Процесний підхід до забезпечення якості



# Capability Maturity Model

## Characteristics of the Maturity levels





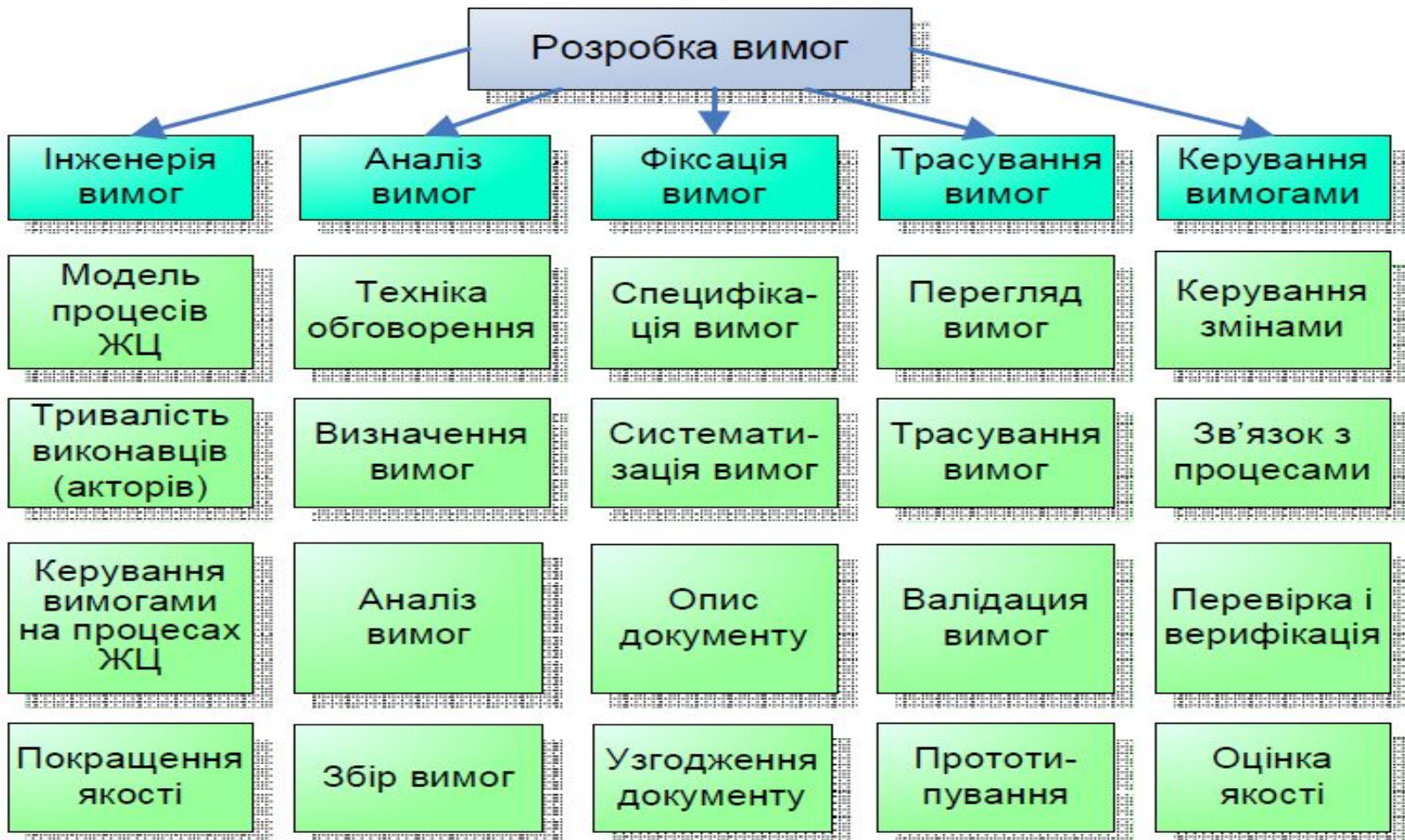
# 5-рівнева Модель Зрілості



# Розробка вимог до ПЗ

*Вимоги до ПЗ – сукупність властивостей, які повинно мати ПЗ.*

*Вимоги призначені для адекватного визначення функцій, умов і обмежень на використання ПП, а також обсягів даних, технічного забезпечення і середовища для його функціонування.*



# Класифікація вимог до ПЗ

## Вимоги до Програмного продукту:

• **Вимоги користувачів** (*user requirements*) щодо зовнішнього поводження системи - це умовами досягнення цілей і задач, віддзеркалюють вимоги споживачів до спектра задач, що буде розв'язувати майбутній програмний продукт

## Вимоги до Програмного забезпечення:

• **Системні вимоги** визначають зовнішні умови виконання системних функцій і обмежень на створення продукту, а також вимоги до опису програмно-апаратних підсистем.

• **Вимоги до атрибутів якості** (*quality attributes*) – це деякі обмеження на властивості функцій або системи, важливі для користувачів або розробників.

• **Функціональні вимоги** – це перелік функцій або сервісів, які повинна надавати система, а також обмежень на дані і поводження системи при їхньому виконанні.

• **Нефункціональні вимоги** визначають умови виконання функцій (напр., захист інформації у БД, аутентифікація доступу до ПС тощо) у середовищі, що безпосередньо не пов'язані з функціями, а відбивають потреби користувачів щодо їх виконання.

# Візуальний підхід в інженерії вимог

*Вимоги задаються за допомогою*

- варіантів використання (*use case*), сценаріїв або прецедентів.

*Для моделі сценаріїв використовується*

графічна нотація UML з такими правилами:

- актор позначається зображенням – іконка людини і можливо з назвою;
- сценарій подається овалом, у середині якого назва зображення іконки;
- актор зв'язується лінійкою з кожним овалом сценарію, що запускається ним в дію.



# Література до лекції

**1. И. Соммервиль.** Инженерия программного обеспечения, 6 изд. – И.д. "Вильямс", 2002.

- <http://www.cs.st-andrews.ac.uk/~ifs/Books/SE9/>

Ian Sommerville. Software Engineering. 9th Edition.

- <http://www.cs.st-andrews.ac.uk/~ifs/Books/SE8/>

Ian Sommerville. Software Engineering. 8th Edition.

**1. Г. Буч.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд. – Бином, 1998.

**2. O. Dahl, E. Dijkstra, C.A.R. Hoare.** Structured Programming.–London, England: Academic Press, 1972.

**3. Р. Лингер, Х. Миллс, Б. Уитт.** Теория и практика структурного программирования. – М.: Мир, 1982.