**Лекція 1. Поняття програмної інженерії. Ядро знань SWEBOK**

**https://github.com/omc-college/ipz4-Software-Engineering/tree/master/2024**

Термін програмна інженерія вперше був введений у 1968 р. З того часу протягом 40 років зміст цього поняття, поступово змінювався і вдосконалювався, що обумовлювалося постійним розвитком програмування, спочатку як мистецтва одинаків, потім як теоретичної та прикладної науки, і, з часом, як інженерної діяльності.

**Програмна інженерія** — це застосування системного, вимірюваного підходу до розробки, використання та супроводу програмного забезпечення, та дослідження цих підходів, тобто застосування принципів **інженерії** до програмного забезпечення.

Інженерна галузь характеризується діяльністю, що ґрунтується на таких принципах**:**

- ***ефективність*** - результати отримують за допомогою заданих ресурсів, які відповідають висунутим вимогам і стандартам;

- ***практичність*** - результати мають конкретних замовників;

- ***фундаментальність*** - результати отримують на основі знань фундаментальних наук;

- ***успадкованість*** - результати отримують на основі нагромадженого досвіду, виключаючи діяльність «з нуля»;

- ***відчутність*** - результати є відчутними продуктами, які можна застосовувати, руйнувати, а також досліджувати за допомогою емпіричних методів пізнання;

- ***супроводження*** - результати, знаходячись в експлуатації, обов'язково супроводжуються (обслуговуються),

Інженерна діяльність реалізується інженерами в контексті технологій. **Технологія** - це організована сукупність процесів, спрямованих на отримання з початкових матеріалів кінцевих продуктів за допомогою методів і засобів технологій. Інженери - це професіонали, чия освіта дозволяє їм, використовуючи знання фундаментальних наук і конкретних технологій, реалізовувати процеси, застосовуючи методи і засоби технологій для створення надійних, широко використовуваних продуктів.

**Програмна інженерія** - це наука побудови комп'ютерних програмних систем (ПС), що містить у собі теоретичні концепції, методи і засоби програмування, технологію програмування, системи та інструменти їхньої підтримки, сучасні стандарти, зокрема, процеси життєвого циклу (ЖЦ), вимірювання і оцінювання якості розробки ПС. Головне призначення програмної інженерії – побудова ПС, починаючи з аналізу предметної області (ПрО) і закінчуючи виготовленням вихідного коду для виконання на комп'ютері.

***Особливістю виробництва нових систем*** є технологія їх проектування від аналізу предметної області до утворення коду для виконання на комп'ютерах. ***Основа інженерії проектування*** – теорія алгоритмів і програмування, теорія обчислень і розподіленої обробки, теорія обчислювальних мереж, математична логіка, теорія керування та ін.

Програмна інженерія (ПІ) містить у собі методи і засоби керування програмними проектами (планування робіт і регулювання ресурсів), експертне оцінювання проміжних результатів розроблення під час процесів ЖЦ, оцінювання ризику побудови програмної системи і досягнутої для неї якості. Ця дисципліна використовує стандарти (наприклад, ISO/IEC 12207, ДСТУ 9126), що регламентують процеси ЖЦ, інженерію вимог, тестування і забезпечення якості шляхом перевірки показників на процесах ЖЦ і кінцевого продукту для їхнього оцінювання. Інакше кажучи, *в програмної інженерії подані питання теоретичної і практичної побудови різних програмних систем для виконання задач з оброблення інформації на комп’ютерах з метою отримання корисних даних.*

Проектування у ПІ – це конструювання комп'ютерних систем методами та засобами програмування за такими *загальними кроками*:

– опис вимог;

– опис специфікацій системи;

– розроблення системи;

– тестування, оцінка надійності і якості системи.

Проектування за цими кроками містить у собі експертні методи оцінки прийняття рішень, змінювання програм і систем, валідацію нових гіпотез, теорію абстрагування, методи трансляції та керування ресурсами виготовлення, розподілення обчислень тощо. Інженерія проектування програмних систем починає діяти, починаючи з формулювання вимог до них, розроблення і супроводу до зняття з експлуатації. До інженерії належать методи оцінки продукту, а саме, розрахунки трудовитрат, обсягу, вартості та якості.

Інженерна діяльність у програмуванні на теперішній час за своєю сутністю дуже *близька до інженерної конвеєрної діяльності в промисловості*, тільки тут готовими «деталями» виступають поки ще не достатньо при промисловому використанні багаторазових програм і систем. На сучасному етапі розвитку базисом інженерії проектування програмних систем стали компоненти повторного використання (reuse), яких достатньо створено у різних областях, і вони подібні до готових деталей в промисловості. Це є фундаментальним для становлення конвеєрного виробництва програмних продуктів, як продуктів промислово-технічного призначення.

Характерною ознакою виробництва програмних продуктів стала поява *нових категорій фахівців*, крім програмістів, а саме, менеджерів, керівників команди розробників, інженерів служби ведення бібліотек, технологів, тестувальників і різного роду контролерів проміжних результатів проектування на процесах ЖЦ.

Програмна інженерія відрізняється від традиційної промислової інженерії природою свого продукту, який не відчувається і не матеріалізується в наочний фізичний предмет, а постійно змінюється під час супроводження та при стрімких темпах розвитку комп’ютерних платформ і середовищ.

**Зміст програмної інженерії за дисциплінами**

*– наукова дисципліна* визначена як сукупність формальних методів специфікації, доведення та верифікації програмних об’єктів, методів їх об’єднання, теоретичних і прикладних методів програмування та теоретичних моделей надійності програм та методів їхнього застосування;

*– інженерна дисципліна* сформульована як сукупність технологічних засобів і методів проектування ПС за фундаментальними моделями ЖЦ, положеннями сучасного стандарту із процесів ЖЦ, техніки аналізу предметної області, формулювання вимог з розробленням за ними відповідного вихідного коду, його супроводу та внесення до нього різного роду змін, включаючи ті, що забезпечують перенесення програмного продукту на інші комп’ютерні платформи;

*– дисципліна керування* базується на теорії управління, як підґрунті для визначення сутності базових методів керування програмним проектом за графіками робіт, спостереження за виконанням планів, керуванням ризиками та формуванням версій (конфігурацій) виготовленого програмного продукту та передачі його користувачам;

– *економічна дисципліна* сформульована як сукупність методів експертного, якісного і кількісного оцінювання проміжних об’єктів ЖЦ, а також економічних методів розрахунків часу, обсягу і вартості виготовлення програмних продуктів, що поставляються на ринок;

– *виробнича дисципліна* – це сучасні промислові технологічні прийоми виробництва прикладних систем, сімейств систем з застосуванням готових програмних ресурсів, включаючи компоненти повторного використання, накопичених у сучасних інформаційних сховищах, одиночні готові програми розв’язку деяких задач, сервісні, агентні артефакти тощо. Для забезпечення їх правильності виконуються методи верифікації, тестування і оцінки за отриманими на них показниками якості програмного продукту.

Між програмною інженерією і комп’ютерними науками в цілому існує різниця. Комп’ютерні науки займаються теорією й методами комп’ютерних і програмних систем, в той час, як програмна інженерія займається практичними проблемами створення програмного забезпечення. Природно, інженер-програміст зобов’язаний в деякому обсязі знати інформатику, проте обсяг теоретичних знань у різних спеціалістів сильно відрізняється і може бути дуже невисоким, що тим не менше, не заважає їм вирішувати деякі задачі. В ідеалі, вся програмна інженерія повинна підтримуватися якимось теоріями інформатики, але на практиці все значно складніше. Інженери часто застосовують перші методи, які попадуться, а елегантні теорії інформатики не завжди можна застосувати до реальних великих систем.

Програмна інженерія відрізняється від системної інженерії. Системна інженерія, а точніше – комп’ютерна системна інженерія, займається всіма аспектами створення й еволюції комплексних систем, в яких програмне забезпечення відіграє значну роль. Таким чином, програмна інженерія є частиною системної інженерії, поряд зі створенням апаратних платформ, створенням архітектури, системною інтеграцією і т.п. У *системній інженерії основний ухил робиться саме на системні питання* (специфікація системи, проектування архітектури, інтеграція й впровадження), а не на складові частини системи. Системна інженерія значно старша програмування як дисципліни.

Програмна інженерія займається питаннями ефективної розробки програмного забезпечення. Існують всього на всього три основні труднощі програмної інженерії:

* Застарілі системи, які необхідно супроводжувати і розвивати;
* Робота в гетерогенному середовищі з розподіленими системами, які включають різне програмне і апаратне забезпечення;
* Нестача часу, яке відводиться на розробку програмних продуктів.

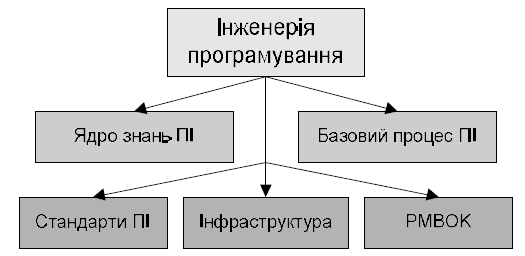
Зрозуміло, що ці проблеми не є абсолютно незалежними і можуть накладатися одна на одну.

**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 SWEBOK** - 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** **(РМВОК)** – ядро знань для керування проектами – набір стандартних процесів, принципів і методів планування і управління проектом.

**Програмна інженерія** – інженерна дисципліна, зв’язана з теорією, методами і засобами професійної розробки ПЗ.

Відомо, що:

1. **ПЗ = програми + вся супутня документація**
2. Висока вартість розробки ПЗ (вища, ніж для апаратури)
3. Вартість розробки ПЗ постійно зростає
4. Програмна інженерія допомагає вирішити **проблему** **зростання вартості розроблення ПЗ**
5. Програмна інженерія має справу з усіма аспектами створення ПЗ

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, що охоплює такі аспекти:
  + 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:***

Багато програм просто не роблять того, чого хочуть кінцеві користувачі.

Типові відсотки для широкомасштабних систем введення в експлуатацію:

* + 45% поставляється, але не використовується
  + 27% оплачено, але не поставлено
  + 17% відкинуто
  + 6% використовуються після змін
  + 5% використовуються як поставлено.

Користувачам важко сформулювати те, що вони хочуть.

Розробникам важко зрозуміти, що кажуть користувачі!

***2. Висока надійність:***

Помилки в програмах загалом відомі як помилки (bugs).

Важливий урок: Ви можете довести, що помилки є; ви не можете довести, що їх нема.

Помилки можуть бути дорогими в термінах:

**– життя людей**: у критичних для безпеки системах, наприклад, управління ядерним реактором, літаючі літальні апарати;

**– гроші:** помилка програмного забезпечення в невдалому запуску Ariane 5 коштувала 500 мільйонів доларів;

**– погані відносини з клієнтами**: проблеми Microsoft з оригінальним випуском Windows викликали у компанії величезні проблеми.

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

Де *функціональність* і *надійність* є обов'язковими критеріями якості ПЗ.

* Забезпечення надійності ПЗ:
* Боротьба зі складністю
* Точність інтерпретації документів
* Подолання бар’єру між розробником і користувачем ПП
* Контроль схвалюваних рішень
* Взаємодія програмних інженерів з науковими розробками

**3. 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** частопризводить до сповільнення проекту.

***Методи боротьби зі складністю програмної системи***

* забезпечення незалежності компонентів системи
* використання в системах ієрархічних структур

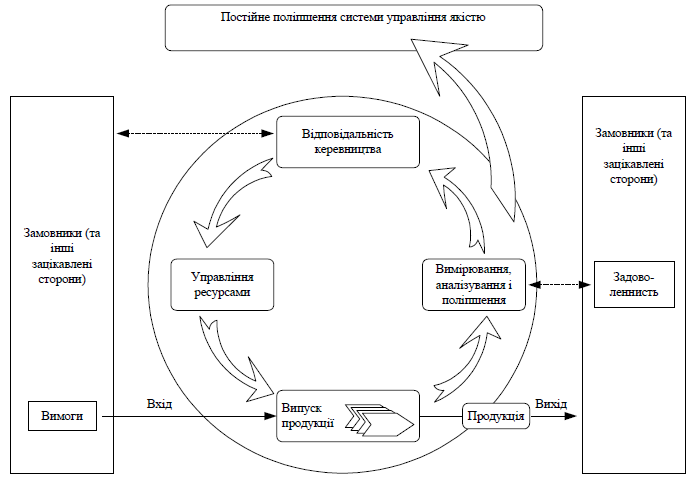
**Моделі якості розроблення ПП**

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

Інформаційні потоки

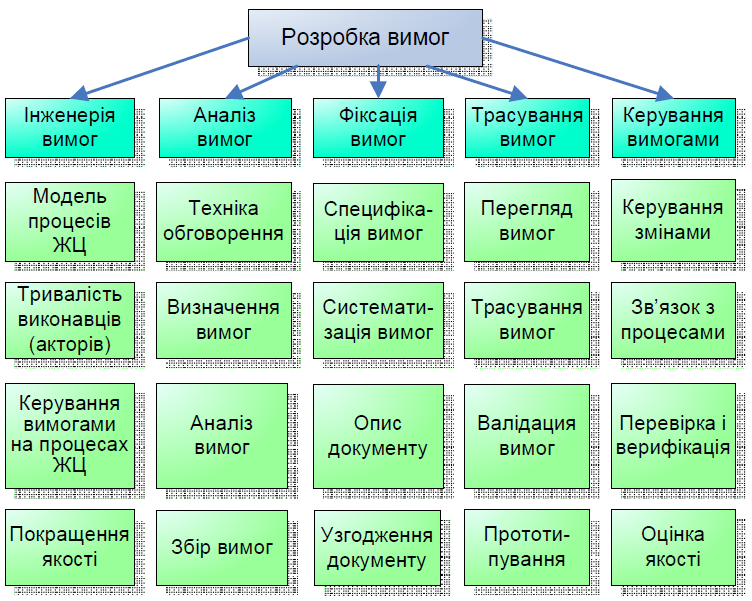
Матеріальні потоки



**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**

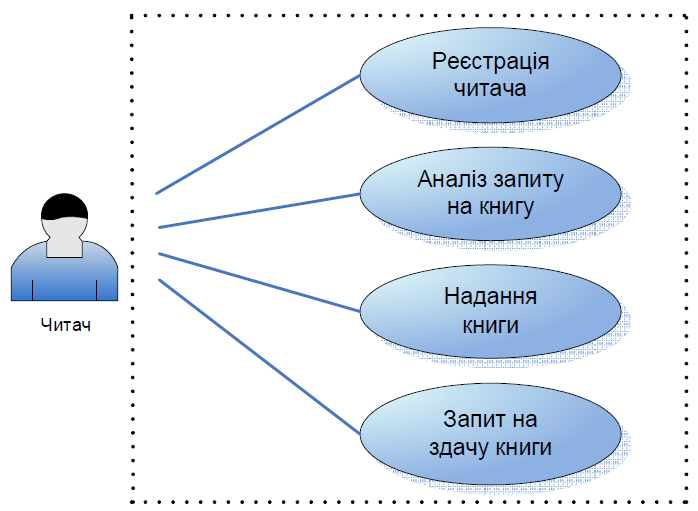
***Вимоги до Програмного продукту:***

* ***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.

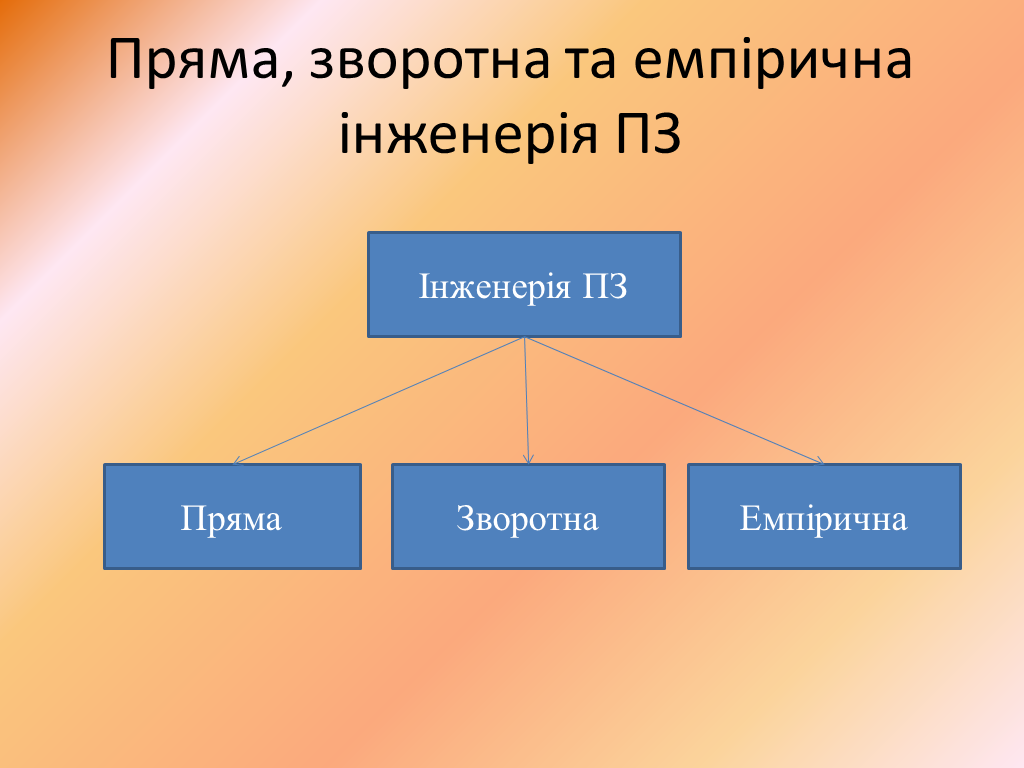
**Візуальний підхід в інженерії вимог**



***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.

**Інженерія програмного забезпечення: пряма, зворотна та емпірична**



* Пряма інженерія ПЗ – це інженерія ПЗ, яка забезпечує процеси розробки ПЗ, починаючи з високорівневих абстракцій в вигляді специфікацій вимог, і закінчуючи реалізацією програмного продукту у вигляді виконуваного коду.
* Зворотна інженерія ПЗ – це інженерія ПЗ, яка забезпечує зворотні процеси. Задача зворотної інженерії полягає у забезпеченні процесів отримання із низькорівневого представлення програмного забезпечення (як правило, вихідного коду) високорівневе його представлення (часто – це проектна інформація).
* Емпірична інженерія програмного забезпечення – сукупність дій для отримання знань з метою кращого розуміння аспектів розробки програмного забезпечення. Результатом дій є ряд тверджень щодо визначеного переліку проблем. Ці твердження являються відповідями на поставлені запитання та підтвердженням чи спростуванням гіпотез.

**Моделювання у програмній інженерії**

Існує багато аспектів, пов'язаних з успішною розробкою програмних проектів, і одним з головних є *моделювання*. Це загальноприйнята в інженерії методика.

*Модель* – це *спрощене відображення дійсності*, при цьому важливо, щоб модель надавала певну уяву про об’єкт моделювання

Чим більша й складніша система, тим важче її "охопити" у цілому, а, отже, тим важливіше моделювання.

Засоби *CASE* (*Computer Aided Software Engineering*) підтримують *моделювання* ПС. Моделі

* + - програмних систем;
    - архітектури ПС;
    - життєвого циклу.

Основні принципи моделювання ПС.

* *Принцип абстрагування*. Система представляється моделлю, що відповідає *певному рівню абстракції*. Вибір моделі (вибір певного рівня абстракції) визначає те, як будуть осмислюватися проблеми реалізації і які рішення будуть прийматися;
* *Принцип візуальності.* Моделі повинні забезпечувати можливість одержувати *візуалізоване* відображення системи (“одна діаграма може замінити *100 сторінок* тексту”);
* *Принцип багатомодельності.* Не слід обмежуватися єдиною моделлю, якщо система досить нетривіальна. Найкраще використовувати сукупність моделей, що *незалежні* одна від одної або, іншими словами, що задають різні *подання системи*.
* *Принцип ієрархічності (ієрархічної побудови моделей)*. Цей принцип обумовлює можливість розробляти моделі у відповідності до різних рівнів конкретизації (абстрагування).
* *Принцип еволюційності.* Як правило, якісна модель системи не є результатом одного акту створення, а отримується шляхом послідовних (еволюційних) уточнень.

**Ядро знань SWEBOK (*Software Engineering Body of Knowledge*)**.

Провідні зарубіжні професіональні об’єднання і виробники програмної продукції розробили визначення ядра професійних знань (*Body of Knowledge* – BoK). Ці знання складають предмет програмної інженерії, а також управління проектами створення промислової продукції. Ними підготовані два керівництва (Guide), відповідно

* *Guide for SWEBOK*
* Guide *for PMBOK (Project Management Body of Knowledge) -* Довідни́к з управлі́ння прое́ктами.

В області якості ПС подібне ядро знань не має чітких обрисів і документально не оформлено.

Виникнення і високі темпи розвитку програмної інженерії визначаються такими факторами:

* Накопичення значного об’єму знань в області практичного створення ПЗ, які потребують систематизації.
* Поява різноманітних методів аналізу, моделювання і проектування ПЗ, а також високо технологічних засобів та інструментів розробки ПЗ, не забезпечених рекомендаціями по ефективному використанню.
* Високий рівень виявлення дефектів в ПЗ, не дивлячись на використання прогресивних методів проектування і програмування.
* Неефективна організація праці колективів розробників ПЗ (менеджерів, проектувальників, програмістів, тестерів, технологів та інших).
* Використання готових програмних компонентів, які підлягають ідентифікації та систематизованому веденню.
* Застосування реінженерії існуючих компонентів як засобів їх адаптації до нових умов і середовищам, що швидко змінюються

Один із метрів програмної інженерії Джексон визначив золоте правило програмування так: *”Будь-яка тільки що завершена програмна система одразу потребує змін”*

Значні зусилля направлені на перетворення програмної інженерії в інженерну спеціальність. Підтвердженням цього є створення ядра swebok, різноманітних програм навчання, інститутів і комітетів, міжнародних професійних об’єднань в області інформатики.

Практика спеціалізації професійної діяльності дозволяє рахувати професію «зрілою» тоді, коли для неї існують:

* Система початкового навчання спеціальності
* Механізми розвитку умінь і навиків персоналу, які необхідні для його практичної діяльності
* Ліцензування спеціалістів, організоване під керівництвом відповідних державних органів
* Системи професійного підвищення кваліфікації персоналу та відстежування сучасного рівня знань і технологій по спеціальності для того, щоб спеціалісти могли вижити в умовах інтенсивного розвитку спеціальності.
* Етичний кодекс спеціалістів.
* Професійне об’єднання.

Програмна інженерія як дисципліно тісно пов’язана з суміжними дисциплінами: комп’ютерні науки, математика, менеджмент, когнітивні науки, керування проектом, телекомунікації та мережі, електротехнічна інженерія та інші інженерні дисципліни

Для створення ядра знань по програмній інженерії SWEBOK в 1993 році сумісними зусиллями ACM (Association for Computing Machinery) та IEEE був створений спеціальний комітет SWECC (Software engineering coordinationg committee). В рамках цього комітету були організовані групи по наступним напрямам досліджень:

* Визначення необхідного ядра знань і рекомендованих практичних засобів діяльності в програмній інженерії
* Визначення норм професійної етики і стандартів з програмної інженерії
* Визначення програм навчання студентів ВНЗ із спеціальності.

*Ядро SWEBOK складають знання з десяти різних областей знань*:

* Програмні вимоги
* Проектування (дизайн) ПЗ
* Конструювання ПЗ
* Тестування ПЗ
* Супровід ПЗ
* Керування конфігурацією ПЗ
* Керування інженерією ПЗ
* Процес інженерії ПЗ
* Інструменти та методи інженерії ПЗ
* Якість ПЗ

Кожній області знань присвячена окрема глава, яка структурована по розділам та рубрикам. Глави завершуються об’ємними списками літератури по предмету, яка по суті і являє матеріал, що утворює ядро знань.

**Програмні вимоги**:

* Основи вимог
* Процес інженерії
* Витяг вимог
* Аналіз вимог
* Специфікація вимог
* Перевірка вимог
* Практичні міркування

**Проектування**:

* Основи проектування
* Ключові питання
* Структура і архітектура
* Аналіз і оцінка якості проекту
* Нотації дизайну
* Стратегії і методи проектування

**Конструювання**:

* Основи конструювання
* Управління конструюванням
* Практичні міркування

**Тестування**:

* Основи тестування
* Рівні тестування
* Засоби тестування
* Метрики тестування
* Процес тестування

**Супровід:**

* Основи супроводу
* Ключові питання
* Процес супроводу
* Практичні засоби

**Управління конфігурацією**:

* Управління процесом
* Ідентифікація конфігурації
* Контроль конфігурації
* Облік стану конфігурації
* Аудит конфігурації
* Управління випуском та поставкою

**Управління інженерією**:

* Ініціювання та визначення рамок проекту
* Планування проекту
* Огляд і оцінка проекту
* Закриття проекту
* Вимірювання в інженерії ПЗ

**Процес програмної інженерії**:

* Реалізація і зміна процесу
* Визначення процесу
* Оцінювання процесу
* Вимірювання процесу і продукту

**Інструменти і методи програмної інженерії**:

* Інструменти розробки
* Управління вимогами
* Проектування
* Конструювання
* Тестування
* Супровід
* Управління конфігурацією
* Управління інженерією
* Підтримка процесів
* Забезпечення якості
* Інші інструменти
* Методи інженерії ПЗ
* Евристичні методи
* Формальні методи
* Методи прототипування

**Якість:**

* Основи якості
* Процеси керування якістю
* Практичні міркування

**Інструменти роботи з вимогами**:

* Засоби моделювання
* Засоби трасіровки

**Інструменти проектування**

* UML
* Бізнес-проектування
* Проектування БД

**Інструменти конструювання**

* Редактори програм
* Компілятори і генератори коду
* Інтерпретатори
* дебаггери

**Інструменти тестування**

* генератори тестів
* засоби виконання тестів
* інструменти оцінки тестів
* засоби керування тестами
* інструменти аналізу продуктивності

**Інструменти супроводу**

* засоби візуалізації
* інструменти реінженерії

**Інструменти управління конфігурацією**

* інструменти відслідковування дефектів і проблем
* інструменти управління версіями
* інструменти зборки та випуску

**Інструменти управління інженерією**

* інструменти планування та відстежування , прогнозування вартості
* Інструменти керування ризиками
* Засоби кількісної оцінки

**Інструменти підтримки процесів**

* Інструменти моделювання процесів
* Засоби керування процесами
* Інтегровані CASE-середовища і рольові платформи розробки
* Процес-орієнтовані середовища розробки

**Інструменти забезпечення якості**

* Інструменти інспекції, підтримка оглядів та аудитів
* Інструменти статичного аналізу

**Додаткові аспекти**

* Засоби інтеграції інструментів: програмні платформи (Java, microsoft .NET), платформи розподілених обчислень (CORBA, WebServices)
* Мета інструменти: засоби генерації інших інструментів, компілятор компіляторів тощо
* Засоби оцінки інструментів

**Ядро знань по керуванню проектами (PMBOK)** Project Management Body of Knowledge

PMBOK визначає 39 процесів ЖЦ проекту, об’єднаних в 5 базових груп процесів і 9 ключових областей знань, типових практично для будь-яких проектів.

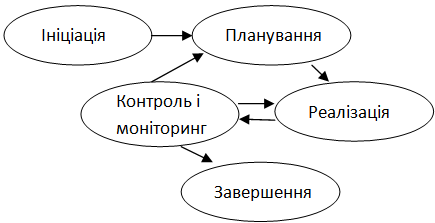
**Групи процесів**:

* Ініціація
* Планування
* Виконання
* Моніторинг і керування
* Завершення

**Ключові області знань**:

* Управління інтеграцією проекту
* Управління змістом проекту
* Управління тривалістю (строками) проекту
* Управління вартістю проекту
* Управління якістю проекту
* Управління людськими ресурсами
* Управління комунікаціями проекту
* Управління ризиками проекту
* Управління закупівлями (поставками) проекту

***Схема взаємодії груп процесі***в



***Для самостійного вивчення***: загальне визначення дисциплін програмної інженерії, характеристика областей знань з інженерії програмного забезпечення за SWEBOK **(**Лавріщева К.М. Програмна інженерія. Електронний підручник розділ 1.1., 1.2.)

***Контрольні запитання****.*

1. Дайте визначення програмної інженерії.
2. Визначте особливості методів програмної інженерії.
3. В чому полягає проектування у програмної інженерії?
4. Загальна характеристика програмної інженерії, як галузі, її порівняння із іншими інженеріями та теоретичною основою – комп‘ютерними науками.
5. Визначтеобласть дії програмної інженерії.
6. Надайте характеристику областей знань SWEBOK.

***Література***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник: http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf (дата звернення 26.01.2021)
2. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
3. Бабенко Л.П., Лавріщева К.М*.* Основи програмної інженерії.– Навч. посібник.–К.: Знання, 2001.– 269с.
4. Лаврищева Е.М., Грищенко В.Н*.* Области знаний программной инженерии – SWEBOK и подход к обучению этой дисциплине// Управляющие системы и машины.– 2005.– №1.– С.38-54.
5. Лавріщева К.М. Основні напрямки досліджень в програмній інженерії і шляхи їхнього розвитку // Проблеми програмування.– 2003. – № 3–4.– С.44-58.
6. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика. – К.: Наук. думка, 2006.– 450с.
7. Основы инженерии качества программных систем / Ф.И.Андон, Г.И.Коваль, Т.М. Коротун, Е.М.Лаврищева, В.Ю. Суслов – К.: Академпериодика.– 2007. – 678с.
8. Лаврищева Е.М., Коваль Г.И., Коротун Т.М*.* Подход к управлению качеством программных систем обработки данных // Кибернетика и системный анализ.– 2006.– № 5.–С.174-185.
9. И. Соммервиль. Инженерия программного обеспечения,  
    6 изд. – И.д. "Вильямс", 2002.
10. Фредерик Брукс. Мифический человеко-месяц или как создаются программные системы. Пер. с англ. – СПб.: из-во «Символ», 2001
11. Уокер Ройс. Управление проектами по созданию программного обеспечения. Пер. с англ. – М.: из-во «Лори», 2002
12. Кендалл С. Унифицированный процесс. Основные концепции.–М.–СПб.– Киев.–2002.– 157с.
13. Guide to the Software Engineering Body of Knowledge. Version 3.0. swebok.°URL: <https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf> (дата звернення 26.01.2021)