Лекція 7. Інженерія якості. Ядро професійних знань

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*.

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

Термін *програмна інженерія* чітко затвердився на початку 70-х років з виходом в світ першого професійного журналу в цій області – Transactions on Software Engineering.

Програмна інженерія – область комп'ютерних наук, яка вивчає питання побудови комп'ютерних програм як інженерної регламентованої діяльності колективів розробників. Це інженерна дисципліна, яка охоплює усі аспекти створення ПЗ. Починаючи від формування вимог і закінчуючи його супроводом аж до зняття з експлуатації.

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.
- Не ефективна 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.

Ядро знань по програмній інженерії (SWEBOK)

Для створення ядра знань по програмній інженерії 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

Класифікація інструментів по SWEBOK

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

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.

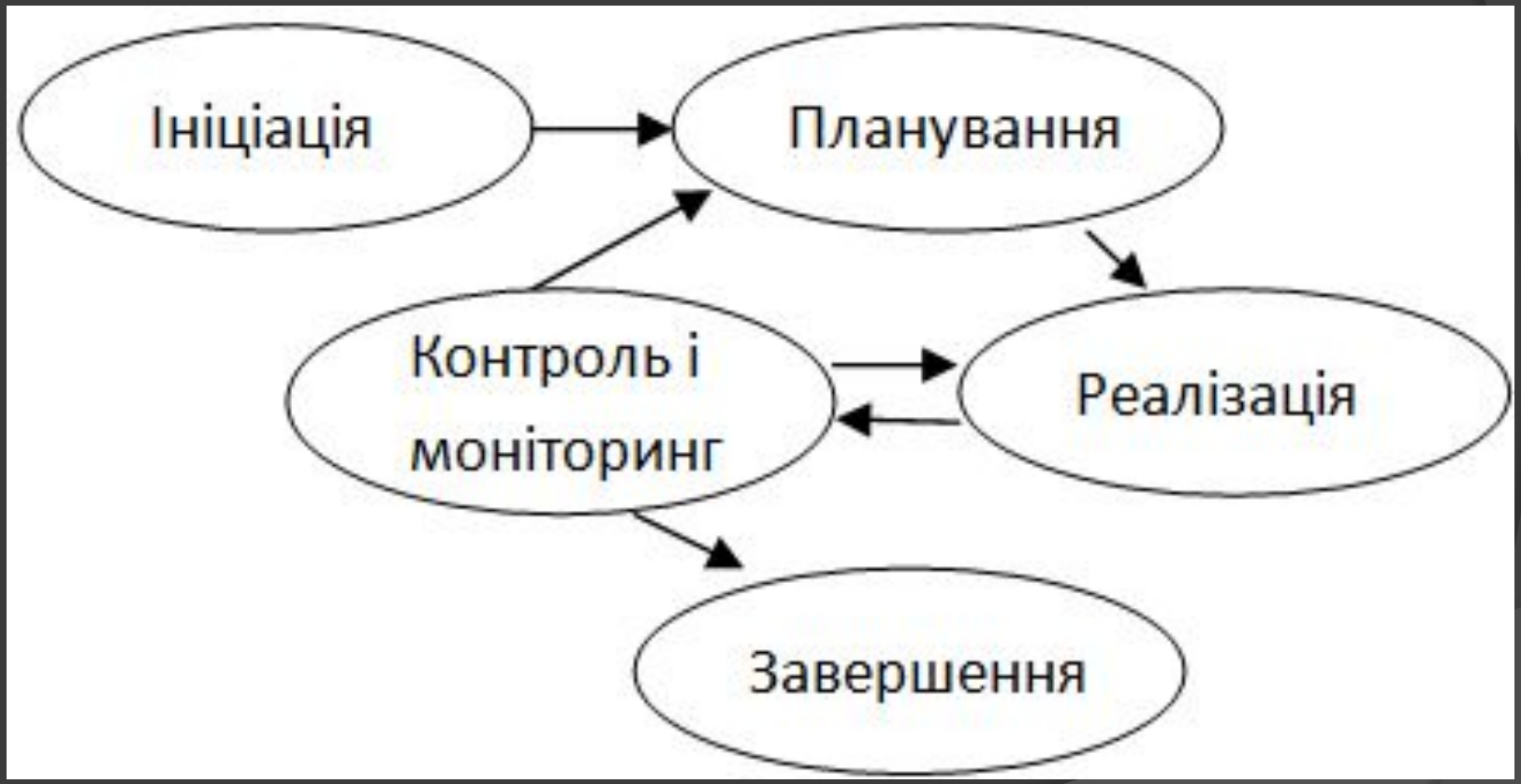
Групи процесів:

- 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 процесів



Парадигми та стилі програмування

Імперативне програмування (основане на машині Тьюрінга-Поста – абстрактному обчислювальному пристрої, яке виконує послідовність інструкцій програми, яка переходить із одного стану в інший):

- 1. Неструктурне програмування - весь код програми представлений одним неперервним блоком (bat-файли, Fortran, Basic, Assembler).
- 2. Процедурне програмування – базужться на концепції виклику процедур (підпрограм, функцій, методів). Algol, Ada, Basic, C, Cobol, Fortran, Pascal, PL/1, Matlab.
- 3. Модульне і збіркове програмування – строго визначені механізми вводу/виводу даних: через аргументи на вході і значення повернення на виході. Мови імперативного та об’єктно-орієнтованого програмування.
- 4. Структурне програмування – направлення процедурного програмування, особливості якого – структурування програми шляхом її розділення на секції одного з видів: послідовність, розгалуження, цикл. Усі сучасні мови імперативного програмування.

Подійно-кероване програмування (event-oriented, event-based, event-driven) – найбільш розповсюджена сучасна парадигма програмування.

Узгоджене програмування (concurrent programming) та ***паралельне програмування***. MPI, OpenMP.

Об'єктно-орієнтоване програмування – основане на представленні предметної області у вигляді системи взаємопов'язаних абстрактних об'єктів та їх реалізацій. Найважливіші принципи ООП – наслідування, інкапсуляція, абстракція, поліморфізм.

- Програмування на класах: SmallTalk, C++, Java, C#, Python, PHP, Object Pascal (Delphi), VB.NET, Xbase++, UML та інші.
- Програмування по прототипах. В цьому стилі поняття класу відсутнє, а повторне використання відбувається шляхом клонування існуючого екземпляру об'єкту – прототипу. Self, JavaScript, Squeak, Cecil, NewtonScript, Io, MOO, REBOL, Kevo та ін.

Декларативне програмування. В такій програмі чітко формулюється ціль і результат її роботи, а не алгоритм отримання результату. HTML, XML, SQL. Domain Specific Language: DSL-мови.

- Функціональне програмування – застосовується для розв'язку задач, які важко сформулювати в термінах послідовних операцій – розпізнавання образів, спілкування на природній мові, реалізація експертних систем, автоматизоване доведення теорем тощо. LISP, ML, Miranda, Haskell, XSLT.
- Логічне програмування. Є ефективним для реалізації задач штучного інтелекту і для описання складних систем, наприклад диспетчерських систем.
- Програмування в обмеженнях (constraint programming) – програмування в термінах постановок задач, де постановка задачі – кінцевий набор змінних, множин значень і набір обмежень. Більшість таких систем – це інтерпретатор мови Пролог із вбудованим механізмом для розв'язку певного класу задач (логічне програмування в обмеженнях). Constraint Logic Programming (CLP). CLP(X), X вказує на клас задач, що розв'язуються. B-Prolog, CHIP V5, Ciao Prolog, ECLiPSe, GNU Prolog. Може бути реалізовано в рамках імперативного програмування як бібліотеки (Java, C++).
- Доказательное програмирование и синтез програм. Побудова програм паралельно з доведенням їх правильності (синтез завідомо правильних програм).

Парадигми прикладного програмування нового покоління.

- ***Сценарна парадигма***. В наш час популярність сценарних мов пов'язана з розвитком Інтернет-технологій. Скриптові мови використовуються для створення динамічних, інтерактивних веб-сторінок, зміст яких модифікується в залежності від дій користувача і стану інших сторінок і даних. Perl, Pyton, PHP, ASP.
- ***Компонентно-орієнтоване програмування***. (Component based development CBD). В основі – індустріальний підхід до розробки програмних систем, не з нуля, а шляхом швидкої зборки з готових програмних компонент. Головна ідея – розповсюдження класів в бінарному вигляді і представлення доступу до методів класу через строго визначені інтерфейси, що дозволяє зняти проблему несумісності компіляторів. COM (DCOM,COM+), CORBA, .Net.

- ***Сервісно-орієнтоване програмування***. Веб-сервіси, інтегровані за допомогою стандартних протоколів SOAP, WSDL. Open Net (Sun), .Net (Microsoft), e-services (HP), Web Services (IBM).
- ***Аспектно-орієнтоване програмування***. Мета – інструментальна підтримка програміста в чіткому поділі компонентів і аспектів за допомогою механізму, який дозволяє абстрагувати і складати компоненти і аспекти для розробки системи в цілому. AspectJ, HyperJ (Java).
- ***Генеруюче програмування***. В її основі суміщення розробки програмних компонент для забезпечення їх повторного використання і наступної розробки ПС із застосуванням компонент, що використовуються повторно. Головним елементом є не унікальний програмний продукт, а родина продуктів. Елементи родини не створюються з нуля. А генеруються на основі загальної генеруючої моделі.

- ***Агентно-орієнтоване програмування***. Інтелектуальний програмний агент – сутність, здатна формулювати цілі, навчатись, планувати свої дії і приймати рішення при обставинах, що динамічно змінюються. Простий приклад – пошук необхідних даних в Інтернет, який вимагає як правило великих затрат часу на вибірку, аналіз і відсіювання зайвої інформації. Найбільш відомі агентні архітектури – PRS, JAM, TOURINMACHINE, COSY, INTERRAP.
- ***Автоматне програмування***. (Switch-технологія). Це стиль програмування, оснований на застосуванні кінцевих автоматів для опису поведінки програм. Автомати задаються графами переходів.

Парадигми теоретичного програмування

- Алгебраїчне програмування базується на теорії переписування термів.
- Інсерційне програмування. Програма в цій парадигмі розглядається як агент, який володіє поведінкою, котрий занурюючись в середовище, міняє його поведінку по відношенню до зовнішнього спостерігача.
- Композиційне програмування, експлікативне програмування.