КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Лекція 1

Основи конструювання програмного забезпечення

Весна 2017

1. ОСНОВИ КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

1.1. Місце КПЗ в життєвому циклі програмної системи.

Розробка програмного забезпечення (ПЗ) – це складний процес, в який входить багато складових.

В загальному випадку це:

- визначення проблеми;
- вироблення вимог;
- створення плану конструювання;
- розробка архітектури ПЗ, або високорівневе проектування;

- детальне проектування;
- кодування і відлагодження;
- блочне тестування;
- інтеграційне тестування;
- інтеграція;
- тестування системи;
- корегувальне супроводження.

Термін **конструювання програмного забезпечення** (software construction) описує детальне створення робочої програмної системи за допомогою комбінації кодування, верифікації (перевірки), модульного тестування (unit testing), інтеграційного тестування та відлагодження.

На рис.1 показано місце конструювання як частину кроків серед процесів, що проходять при побудові ПЗ.

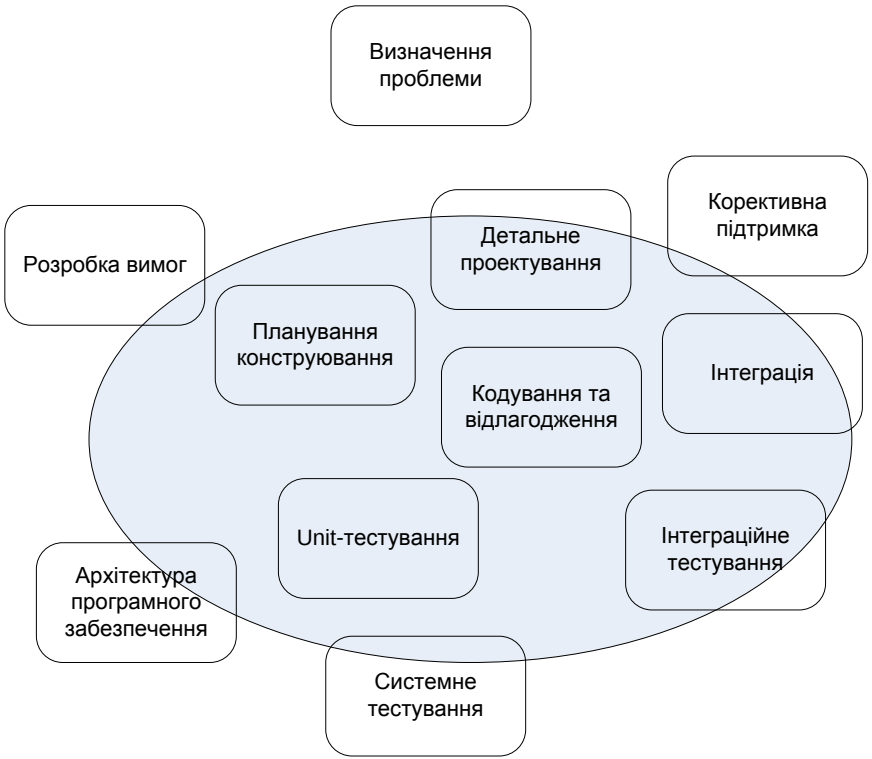

Рис.1. Конструювання серед процесів побудови ПЗ

Процеси конструювання зображені всередині сірого еліпсу. Головними компонентами конструювання є кодування та відлагодження, однак воно включає і детальне проектування, блочне тестування, інтеграційне тестування та інші процеси.

Іноді конструювання називають "кодуванням" або "програмуванням". Кодування в даному випадку видається не найкращим терміном, так як воно має на увазі механічну трансляцію розробленого плану в команди мови програмування, тоді як конструювання є зовсім не механічним процесом і часто пов’язане з творчістю та аналізом. Сенс слів "конструювання" та "програмування" досить близький.

Дана область знань пов'язана з іншими областями. Найбільш сильний зв'язок існує з проектуванням (Software Design) і тестуванням (Software Testing). Причиною цього є те, що сам по собі процес конструювання програмного забезпечення зачіпає важливі аспекти діяльності з проектування й тестування. Крім того, конструювання відштовхується від результатів проектування, а тестування (у будь-якій своїй формі) передбачає роботу з результатами конструювання.

Досить складно визначити межі між проектуванням, конструюванням і тестуванням, тому що всі вони пов'язані в єдиний комплекс процесів життєвого циклу і, в залежності від обраної моделі життєвого циклу і застосовуваних методів (методології), таке розділення може мати різний вигляд.

Хоча ряд операцій з проектування детального дизайну може відбуватися до стадії конструювання, великий обсяг такого роду проектних робіт відбувається паралельно з конструюванням або як його частина. Це є сутність зв'язку з областю знань "Проектування програмного забезпечення".

У свою чергу, протягом всієї діяльності з конструювання, інженери використовують модульне і інтеграційне тестування. Таким чином пов'язана дана галузь знань з "Тестуванням програмного забезпечення".

У процесі конструювання звичайно створюється більша частина активів програмного проекту - конфігураційних елементів (configuration items). Тому в реальних проектах просто неможливо розглядати діяльність по конструюванню у відриві від галузі знань "конфігураційного управління" (Software Configuration Management).

Так як конструювання неможливе без використання відповідного інструментарію і, ймовірно, дана діяльність є найбільш інструментально-насиченою, важливу роль у конструюванні грає область знань "Інструменти і методи програмної інженерії" (Software Engineering Tools and Methods).

Безумовно, питання забезпечення якості значимі для всіх галузей знань і етапів життєвого циклу. У той же час, код є основним результуючим елементом програмного проекту. Таким чином, явно напрошується і присутній зв'язок обговорюваних питань з областю знань "Якість програмного забезпечення" (Software Quality).

З пов'язаних дисциплін програмної інженерії (Related Disciplines of Software Engineering) найбільш тісний і природний зв'язок даної галузі знань існує з комп'ютерними науками (computer science). Саме в них, звичайно, розглядаються питання побудови та використання алгоритмів і практик кодування. Нарешті, конструювання стосується і управління проектами (project management), причому, в тій мірі, наскільки діяльність з управління конструюванням важлива для досягнення результатів конструювання.

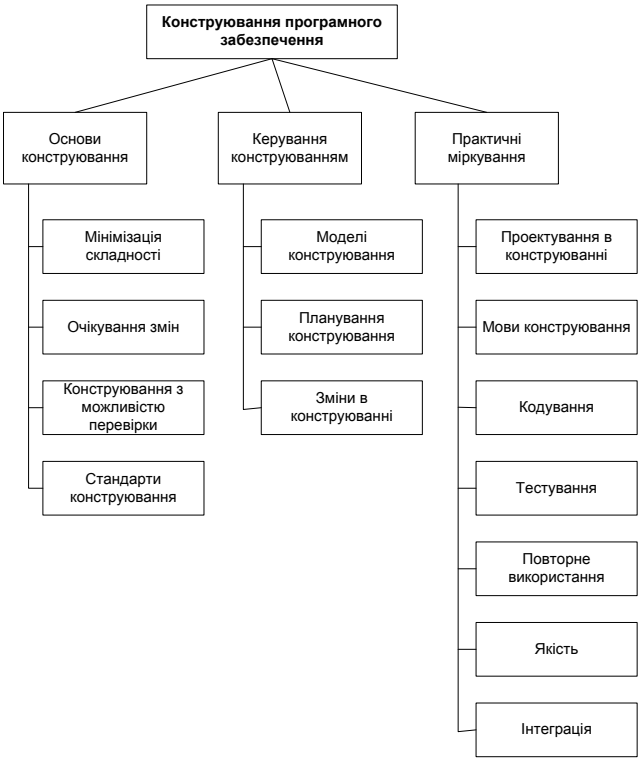

Рис.2. Область знань "Конструювання програмного забезпечення"

Далі наведено деякі конкретні задачі, що виникають в процесі розробки ПЗ, пов’язані з конструюванням:

- перевірка виконання умов, необхідних для успішного конструювання;
- визначення способів подальшого тестування коду;
- проектування та написання класів та методів;
- створення та присвоєння імен змінних та іменованих констант;
- вибір управляючих структур та організації блоків команд;
- блочне тестування, інтеграційне тестування і відлагодження власного коду;
- взаємний огляд коду та низькорівневих програмних структур членами групи;
- "шліфування" коду шляхом його ретельного форматування та коментування;
- інтеграція програмних компонентів, створених окремо;
- оптимізація коду, яка направлена на підвищення його швидкодії і зниження міри використання ресурсів.

З іншого боку, з видів діяльності, що проходять в процесі розробки ПЗ, до конструювання не відносяться: керування процесом розробки, виробки вимог, розробка високорівневої архітектури програми, проектування інтерфейсу користувача, тестування системи і її супроводження – для кожного з цих пунктів є своя наука.

1.2. Фундаментальні складові конструювання програмного забезпечення.

Фундаментальні основи конструювання програмного забезпечення включають:

- Мінімізація складності
- Очікування змін
- Конструювання з можливістю перевірки
- Стандарти у конструюванні

Перші три концепції застосовуються не тільки до конструювання, але й проектування, і лежать в основі сучасних методологій управління життєвим циклом програмних систем.

1.3. Мінімізація складності (Minimizing Complexity)

Основною причиною того, чому люди використовують комп'ютери в бізнес-цілях, є обмежені можливості людей в обробці і зберіганні складних структур і великих обсягів інформації, зокрема, протягом тривалого періоду часу. Це міркування є однією з основних рушійних сил у конструюванні програмного забезпечення: мінімізація складності. Потреба у зменшенні складності впливає на всі аспекти конструювання і особливо критична для процесів верифікації (перевірки) і тестування результатів конструювання, тобто самих програмних систем.

Зменшення складності у конструюванні програмного забезпечення досягається за допомогою звертання особливої уваги на створення простого коду, який легко читається, іноді на шкоду прагненню зробити його ідеальним (наприклад, з точки зору гнучкості або слідування тим чи іншим уявленням про красу, витонченість коду, вправність тих чи інших прийомів тощо). Це не означає, що повинно обмежуватися застосування тих чи інших розвинених мовних можливостей використовуваних засобів програмування. Мається на увазі "лише" надання більшої значимості читаності коду, простоті тестування, прийнятного рівня продуктивності та задоволенню заданих критеріїв, замість постійного вдосконалення коду, не оглядаючись на терміни, функціональність і інші характеристики та обмеження проекту.

Мінімізація складності досягається, зокрема, за рахунок слідування стандартам, використанням низки специфічних технік кодування і підтримкою практик, спрямованих на забезпечення якості в конструюванні.

1.4. Очікування змін (Anticipating Changes)

Більшість програмних систем змінюються з плином часу. Причин цьому - безліч. Очікування змін є однією з рушійних сил конструювання програмного забезпечення. Програмне забезпечення не є ізольованим від зовнішнього оточення (як системного, так і з точки зору галузі діяльності, для автоматизації задач і проблем якого воно застосовується). Більш того, програмні системи є частиною середовища, що змінюється, і повинні змінюватися разом з нею, а, іноді, і бути джерелом змін самого середовища.

Очікування змін підтримується рядом технік кодування.

1.5. Конструювання з можливістю перевірки (Constructing for Verification)

"Конструювання для перевірки" (а саме такий сенс закладений в оригінальній назві даної підтеми) припускає, що побудова програмних систем повинна вестися таким чином, щоб сама програмна система допомагала вести пошук причин збоїв, будучи прозорою для застосування різних методів перевірки (і, до речі, внесення необхідних змін), як на стадії незалежного тестування (наприклад, інженерами-тестувальниками), так і в процесі операційної діяльності - експлуатації, коли особливо важлива можливість швидкого виявлення та виправлення помилок що виникають.

Серед технік, спрямованих на досягнення такого результату конструювання:

- Огляд, оцінка коду (code review)
- Модульне тестування (unit-testing)
- Структурування коду для і спільно з застосуванням автоматизованих засобів тестування (automated testing)
- Обмежене застосування складних або важких для розуміння мовних структур