Курс «Конструювання ПЗ»

Лекція 1. Технології, моделі і процеси створення ПЗ

Програмна інженерія

Чим *програмування* відрізняється від програмної інженерії? Тим, що перше є деякою абстрактною діяльністю і може відбуватися в багатьох різних контекстах. Можна програмувати для задоволення, для того, щоб навчитися (наприклад, на практичних заняттях, на семінарах в університеті), можна програмувати в межах наукових розробок. А можна займатися промисловим програмуванням. Як правило, це відбувається в команді, і абсолютно точно – для замовника, який платить за роботу гроші. При цьому необхідно точно розуміти, що потрібне замовникові, виконати роботу в певні терміни і результат повинен бути потрібної якості – такий, який задовольнить замовника і за який він заплатить. Щоб задовольнити цим додатковим вимогам, програмування набуває різних додаткових видів діяльності: розробкою вимог, плануванням, тестуванням, конфігураційним управлінням, проектним менеджментом, створенням різної документації (проектною, призначеною для користувача і ін.).

Розробці програмного коду передує аналіз і проектування (перше означає створення функціональної моделі майбутньої системи без урахування реалізації, для усвідомлення програмістами вимог і очікувань замовника; друге означає попередній *макет*, ескіз, план системи на папері). Трудовитрати на *аналіз* і проектування, а також форма представлення їх результатів сильно варіюються від видів проектів і вподобань розробників і замовників.

Потрібні також спеціальні зусилля *для* організації процесу розробки. У загальному вигляді це ітеративно-інкрементальна модель, за якою необхідна функціональність створюється порціями, яку менеджери і замовник можуть оцінити, і тим самим, є можливість управління перебігом розробки. Проте, ця загальна модель має безліч модифікацій і варіантів.

Розробку системи також необхідно виконувати з урахуванням зручностей її подальшого супроводу, повторного використання і інтеграції з іншими системами. Це означає, що система розбивається на компоненти, які є зручними в розробці, придатними для повторного використання і інтеграції та мають необхідні характеристики, наприклад, *за* швидкодією. Для цих *компонент ретельно* опрацьовуються інтерфейси. Сама ж система документується на багатьох рівнях, створюються правила оформлення програмного коду – тобто залишаються численні семантичні сліди, що допомагають

створити, зберегти і підтримувати єдину, стрункy архітектуру, одноманітний стиль, порядок.

Всі ці та інші додаткові види діяльності, що виконуються в процесі промислового програмування і є необхідними для успішного виконання замовлень, і називатимемо **програмною інженерією** (software engineering). Так ми позначаємо, по-перше, деяку практичну *діяльність*, а по-друге, спеціальну **галузь знання**, або іншими словами, наукову дисципліну. Адже для полегшення виконання кожного окремого проекту, для можливості використовувати різноманітний позитивний *досвід*, що був досягнутий іншими командами і розробниками, цей самий *досвід* піддається осмисленню, узагальненню і належному оформленню. Так з'являються різні методи і практики (*best practices*), саме: тестування, проектування, робота над вимогами, архітектурні шаблони та ін. Також з'являються стандарти і методології, що стосуються всього процесу в цілому (наприклад, *MSF*, *RUP*, *CMMI*, Scrum). Ось ці-то узагальнення і входять в програмну інженерію, як у галузь знання.

Ще можна визначити, що ***Програмна інженерія (Software Engineering)*** є галуззю інформатики, яка вивчає питання побудови комп'ютерних програм, відображає закономірності розвитку програмування, узагальнює досвід програмування у вигляді комплексу знань і правил регламентації інженерної діяльності розробників ПЗ.

Як інженерна дисципліна, вона охоплює всі аспекти створення ПЗ, починаючи від формування вимог до створення, супроводу і зняття з експлуатації ПЗ, а також включає інженерні методи оцінки трудовитрат, вартості, продуктивності і якості. Тобто мова йде саме про інженерну діяльність в програмуванні, оскільки її сутність близька до визначення інженерної діяльності в тлумачному словнику:

- ***інженерія*** - це спосіб застосування наукових результатів, що дозволяє отримувати користь від властивостей матеріалів і джерел енергії;
- ***інженерія*** - діяльність по створенню машин для надання корисних для споживача послуг і виробів.

Історична довідка з програмної інженерії

Необхідність в програмній інженерії, як у спеціальній галузі знань, була усвідомлена світовою спільнотою в кінці 60-х років минулого століття, більш ніж на 20 років пізніше за народження самого програмування, якщо вважати таким фактом появу відомого звіту фон Неймана "*First Draft*of a *Report*onthe EDVAC" («Перший проект звіту на EDVAC»), що був надрукований ним в 1945 році. Народженням програмної інженерії є 1968 рік – конференція *NATO Software Engineering*, м. Гарміш (ФРН), яка цілком була присвячена розгляду

цих питань. У сферу програмної інженерії потрапляють всі питання і теми, які пов'язані з організацією і поліпшенням процесу розробки ***ПЗ***, управлінням колективом розробників, розробкою і впровадженням програмних засобів підтримки життєвого *циклу розробки ПЗ.*

Новий програмний проект розробляється 1-2 роки, а еволюціонує 6-7 років. На супровід проекту витрачається 61% проти 39% коштів на його розробку. Ефективність розробників в залежності від кваліфікації коливається в співвідношенні 1:10, а значить, потрібно підвищувати рівень знань розробників ПЗ. На сьогодні ядро стабільних знань з програмної інженерії становить 75% від тих знань, якими користуються в практичній діяльності. У зв'язку з цим проведена систематизація накопичених знань в програмуванні та ряді інших областей інформатики. Міжнародним комітетом при американському об'єднанні комп'ютерних фахівців ACM (Association for Computing Machinery) і інституті інженерів з електроніки та електротехніки IEEE Computer Society було створено ядро знань SWEBOK. У цьому ядрі були систематизовані різнорідні знання в області програмування, планування і управління, сформульовано поняття програмної інженерії та десяти областей, які відповідають процесам проектування ПЗ і методам їх підтримки.

Складові частини програмної інженерії

В редакції SWEBOK 2004 року визначаються десять областей знань в контексті програмної інженерії, які поділено на 2 групи.

Основні області (Рис. 1.1):

- Вимоги до ПЗ (Software requirements).
- Проектування (Software design).
- Конструювання (Software construction).
- Тестування (Software testing).
- Супроводження (Software maintenance).

Організаційні області (Рис. 1.2):

- Управління конфігурацією (Software configuration management).
- Управління проектами (Software engineering management).
- Процеси програмної інженерії (Software engineering process).
- Засоби та інструменти (Software engineering tools and methods).
- Якість ПЗ (Software quality).

Також SWEBOK визначає дисципліни, що відіграють велику роль в програмній інженерії:

- Комп'ютерна інженерія
- Комп'ютерні науки
- Менеджмент

- Математика
- Контроль якості
- Ергономіка ПЗ
- Системне адміністрування

Рис. 1.1 Основні області програмної інженерії

Рис. 1.2 Організаційні області програмної інженерії

Термінологія

Програмне забезпечення (ПЗ) – комп'ютерні програми і відповідна документація. Розробляється за приватним замовленням або для продажу на ринку ПЗ.

Інженерія ПЗ інженерна дисципліна, що охоплює всі аспекти розробки ПЗ.

Системотехніка (технологія створення обчислювальних систем) дисципліна, що охоплює всі аспекти створення і модернізації складних обчислювальних систем, де програмне забезпечення грає провідну роль. Сюди можна віднести технології створення апаратних засобів, створення обчислювальних процесів, розгортання всієї системи, а також технологію створення безпосередньо.

Процес створення ПЗ – сукупність процесів, що приводять до створення програмного продукту.

Визначення програмного забезпечення за Харальдом Мілсом.

Програмне забезпечення (*ПЗ*) безліч логічних розпоряджень, що розвиваються в часі, за допомогою яких деякий колектив людей управляє і використовує багатопроцесорну і розподілену систему обчислювальних пристроїв.

Це ***визначення***, яке дане Харальдом Мілсом, відомим фахівцем в галузі програмної інженерії з компанії *IBM*, містить в собі таке.

1. Логічні розпорядження – це не лише самі програми, але і різна документація (наприклад, з експлуатації програм) і ширше – певна система відносин між людьми, що використовують ці програми в межах деякого процесу діяльності.
2. Сучасне ПЗ призначене, як правило, для одночасної роботи з багатьма користувачами, які можуть бути значно віддалені один від одного у фізичному просторі. Таким чином, обчислювальне середовище (персональні комп'ютери, сервера і так далі), в якому ПЗ функціонує, виявляється розподіленим.
3. Завдання, які вирішуються сучасним ПЗ, часто вимагають різних обчислювальних ресурсів через різну спеціалізацію цих завдань, із-за великого об'єму виконуваної роботи, а також з міркувань безпеки. Наприклад, з'являється сервер бази даних, сервер додатків і ін. Таким чином, обчислювальне середовище, в якому ПЗ функціонує, виявляється багатопроцесорним.

4. ПЗ розвивається в часі – виправляються помилки, додаються нові функції, випускаються нові версії, міняється його апаратна база.

Властивості програмного забезпечення

Таким чином, ПЗ є складною динамічною системою, що включає технічні, психологічні і соціальні аспекти. ПЗ помітно відрізняється від інших видів систем, що створюються (створених) людиною, – механічних, соціальних, наукових і ін., і має такі особливості, які виділені Фредеріком Бруксом (*Фредерік Філіпс Брукс — інженер програмного забезпечення та вчений-інформатик, найбільш відомий за управління розробкою ОС System/360 для IBM, та її наступників, а пізніше за опис цього процесу в книжці «Міфічний людино-місяць».*) в його статті "Срібної кулі немає".

1. ***Складність програмних об'єктів***, яка істотно залежить від їх розмірів. Як правило, більше ПЗ (більша кількість користувачів, більший об'єм оброблюваних даних, жорсткіші вимоги по швидкодії і ін.) з аналогічною функціональністю – це інше ПЗ. Класична наука будувала прості моделі складних явищ, і це вдавалося, оскільки складність не була характеристичною межею даних явищ. (Порівняння програмування саме з наукою, а не з театром, кінематографом, спортом і іншими галузями людської діяльності, виправдано, оскільки воно походить, головним чином, з математики, а перші його плоди – програми – призначалися для використання при наукових розрахунках. Крім того, більшість програмістів мають природничо-наукову, математичну або технічну освіту. Таким чином, парадигми наукового мислення широко використовуються при програмуванні – явно або неявно.)

2. ***Узгодженість*** – ПЗ ґрунтується не на об'єктивних посилках (подібно до того, як різні системи в класичній науці ґрунтуються на постулатах і аксіомах), а повинно бути узгоджене з великою кількістю інтерфейсів, з якими згодом воно повинне взаємодіяти. Ці інтерфейси погано піддаються стандартизації, оскільки ґрунтуються на численних людських угодах, що погано формалізуються.
3. ***Змінність*** – ПЗ легко змінити і, як наслідок, вимоги до нього постійно змінюються в процесі розробки. Це створює багато додаткових труднощів підчас його розробки і еволюції.

4. ***Нематеріальність*** – ПЗ неможливо побачити, воно віртуальне. Тому, наприклад, важко скористатися технологіями, якізасновані на попередньому створенні креслень, що успішно використовуються в інших промислових галузях (наприклад, в будівництві, машинобудуванні). Там на кресленнях в схемному вигляді відтворюються геометричні форми створюваних об'єктів. Коли об'єкт створений, ці форми можна побачити.

Фундаментальні процеси, властиві будь-якому проекту створення ПЗ:

- Розробка специфікації вимог на ПЗ (Визначають функціональні характеристики системи і обов'язкові для виконання).
- Створення програмного забезпечення (створення ПЗ згідно специфікації).
- Атестація ПЗ (Створене ПЗ повинно пройти атестацію для підтвердження відповідності вимогам замовника).
- Модернізація ПЗ (вдосконалення ПЗ згідно зміненим вимогам споживача).

Методи створення ПЗ є структурний підхід до створення ПЗ, який сприяє виробництву ПЗ ефективних, з економічної точки зору, способом.

Всі методи засновані на використанні моделей системи як специфікації її структури:

1. **Функціонально-орієнтовані** (структурний аналіз, JSD, 70-і роки) засновані на визначенні основних функціональних компонент системи.
2. **Об'єктно-орієнтовані** (Booch, Rumbaugh) використовують підходи, засновані на використанні уніфікованої мови моделювання UML.

Computer-Aided Software Engineering – це засоби автоматизованої розробки ПЗ.

Процеси створення ПЗ

Базові процеси створення ПЗ:

- Розробка специфікації.
- Проектування і реалізація.
- Атестація.

- Еволюція.

Життєвий цикл ПЗ сукупність процесів, що протікають від моменту ухвалення рішення про створення ПЗ до його повного виводу з експлуатації.

Модель процесу створення ПЗ – послідовність етапів, необхідних для розробки створюваного ПЗ.

Моделі процесу розробки ПЗ:

1. Каскадна модель
2. Еволюційна модель
3. Формальне перетворення
4. Збірка програмних продуктів з раніше створених компонентів (модель збірки)
5. Ітераційна (спіральна) модель

Каскадна модель

Каскадна модель життєвого циклу ПЗ наведена на Рис. 1.3.

Переваги:

- Документування кожного етапу.

Недоліки:

- «Негнучке» розбиття процесу створення на окремі етапи.

Застосування:

- Вимоги сформульовані достатньо чітко.
- Повсюдно для розробки невеликих систем, що входять до складу крупного проекту.

Еволюційна модель

Еволюційна модель життєвого циклу ПЗ наведена на Рис. 1.4.

Прототип – програмний модуль, що діє, реалізовує окремі функції створюваного ПЗ.

Переваги:

- Специфікація розробляється поступово, у міру вимоги замовника.

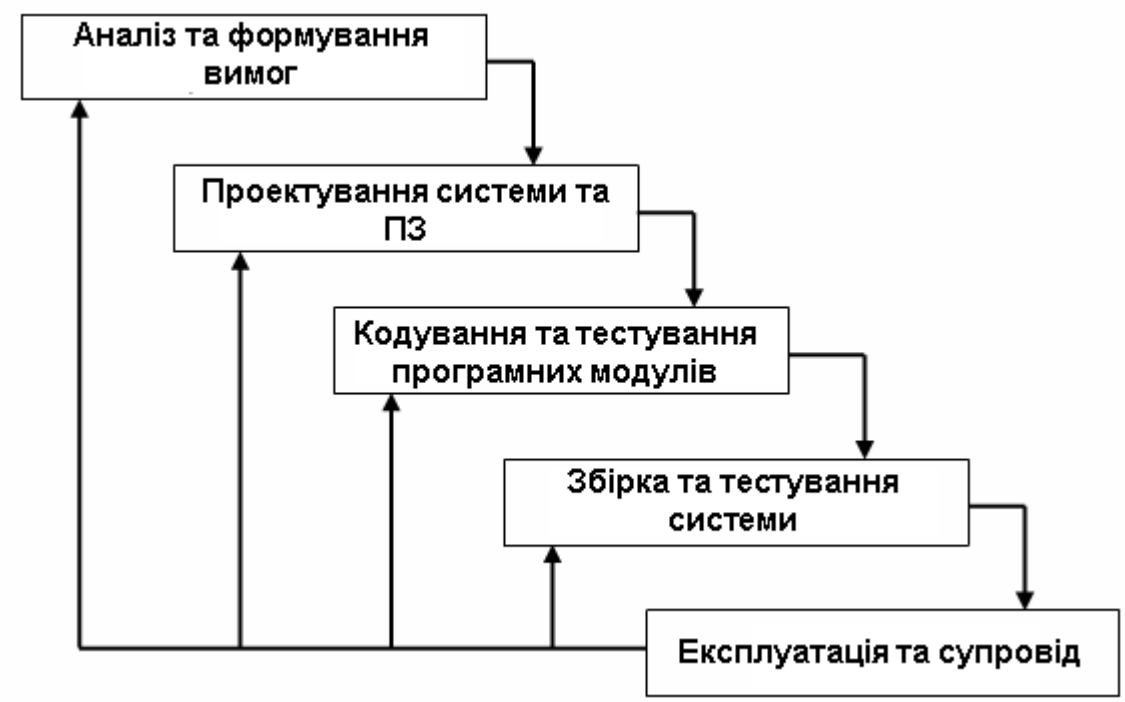

Рис. 1.3 Каскадна модель створення ПЗ

Недоліки:

- Багато етапів створення ПЗ не документовані.
- Система часто виходить погано структурованою.
- Потрібні спеціальні засоби і технології розробки ПЗ.

Застосування:

- Розробка невеликих систем (<100 000 рядків) або середніх (<500 000 рядків) з відносно коротким терміном життя.

Формальна розробка

Модель життєвого циклу ПЗ «Формальна розробка» наведена на Рис. 1.5.

Переваги:

- Точна відповідність програми специфікації.
- Відмова від тестування окремих модулів.
- Тестування всієї системи тільки після її збірки.

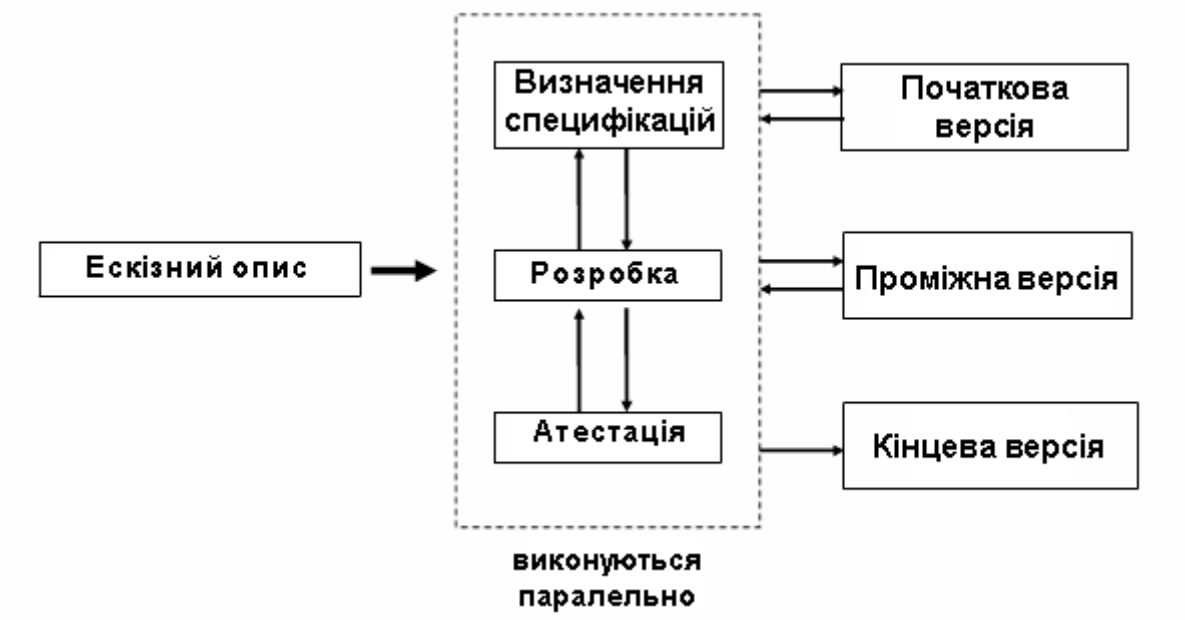

Рис. 1.4 Еволюційна модель створення ПЗ

Недоліки:

- Вимагають спеціальних знань і досвіду використання.
- Не дають істотного виграшу у вартості розробки.
- Більшість складних систем дуже важко піддаються формальному опису.

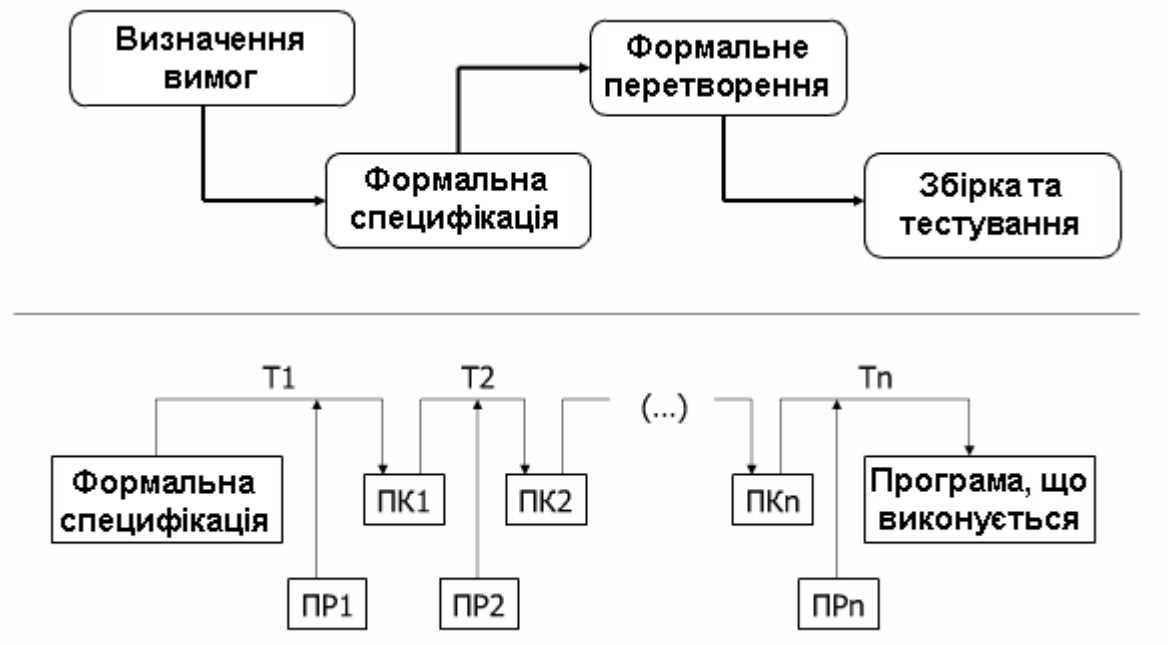

Рис. 1.5 Процес формальних перетворень

Модель покрокової розробки

Модель життєвого циклу ПЗ покрокової розробки наведена на Рис. 1.6.

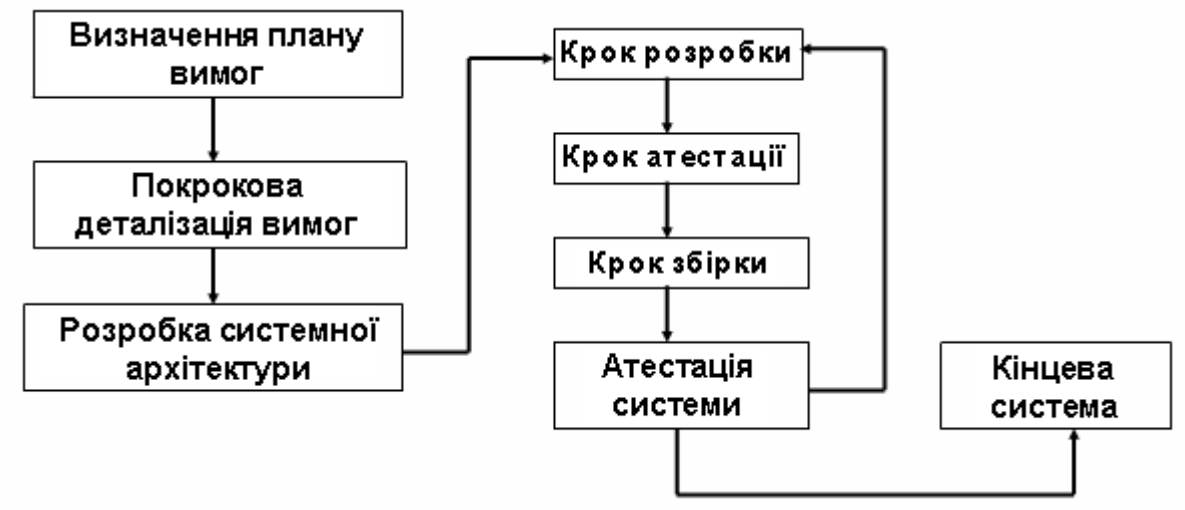

Рис. 1.6 Модель покрокової розробки

На кожному кроці відсутня вимога використання одного і того ж підходу до процесу розробки!

Переваги:

- Немає необхідності чекати повного завершення розробки системи.
- Можна використовувати компоненти, отримані на перших кроках, як прототипи.
- Зменшується ризик загальносистемних помилок.
- Системні сервіси з високим пріоритетом розробляються першими, а все подальші інтегруються з ними. Це дозволяє понизити вірогідність програмних помилок в особливо важливих частинах системи.

Недоліки:

- Компоненти, що отримуються на кожному кроці, мають невеликий розмір.
- Складно визначити на перших етапах загальносистемні функції.
- Неможливо відразу визначити набір базових властивостей, які часто розробляються спільно з іншими частинами системи.

Спіральна модель

Спіральна модель життєвого циклу ПЗ наведена на Рис. 1.7.

Рис. 1.7 Спіральна модель

Переваги:

- Немає фіксованих етапів.
- Ця модель може включати будь-які інші моделі на кожному витку спіралі:
  - прототипіювання може використовуватися при нечіткому визначенні вимог;
  - Каскадна модель у разі послідовного виконання деяких етапів;
  - Модель формальних перетворень – якщо чітко сформульовані вимоги.

Недоліки:

- Складна процедура автоматизації процесів розробки.
- Величезна роль під час розробки системи відводиться управлінню проектом.

Методи створення ПЗ

Методи являють собою структурний підхід до створення ПЗ, який сприяє виробництву ПЗ ефективним, з економічної точки зору, способом.

Всі засновані на використанні моделей системи в якості специфікації її структури:

- **Функціонально-орієнтовані** (структурний аналіз, JSD, 70-ті роки) засновані на визначенні основних функціональних компонент системи.
- **Об'єктно-орієнтовані** (Booch, Rumbaugh) використовують підходи, засновані на використанні уніфікованої мови моделювання UML.

CASE-технології

Computer-Aided Software Engineering – це засоби автоматизованої розробки ПЗ.

Широкий спектр програм, що застосовуються для підтримки і супроводу різних етапів створення ПЗ:

Верхній рівень:

- Аналіз системних вимог
- Моделювання системи

Нижній рівень:

- Налагодження і тестування
- Створення документації
- Генерація вихідного коду програм
- Інші...

Характеристики якості ПЗ

Зручність супроводу

- удосконалення у відповідь на змінені вимоги замовника

Надійність

- безвідмовність
- захищеність
- безпеку

Ефективність

- швидкість виконання
- процесорний час
- обсяг необхідної пам'яті

Зручність використання

- не вимагає надмірних зусиль користувача
- відповідний призначений для користувача інтерфейс
- документація

Базові процеси створення ПЗ

Розробка специфікації ПЗ – це визначення сервісів, якими володітиме створюване ПЗ, а також обмежень, що накладаються на функціональні можливості і розробку ПЗ.

Результатом процесу визначення вимог є документація, яка формалізує вимоги, що пред'являються до системи.

Два рівні деталізації:

- Вимоги, що пред'являються кінцевими користувачами (Рис. 1.8);
- Системна специфікація для розробників (Рис. 1.9).

Реалізація ПЗ – це процес перекладу системної специфікації в працездатну систему. Включає процеси проектування і програмування.

Процес проектування включає визначення структури ПЗ, даних, інтерфейсів взаємодії системних компонентів, використовувані алгоритми. Проектування припускає послідовну формалізацію і деталізацію створюваного ПЗ.

Результат кожного етапу проектування – специфікація, яка є необхідною для виконання наступного етапу.

Методи проектування – суть безліч формалізованих нотацій і нормативних документів для проектування ПЗ.

Структурні методи підтримують моделі системи:

- Модель потоків даних;
- Модель «сутність-зв'язок»;
- Структурна модель;
- Об'єктно-орієнтовані ієрархічна модель системи, модель відносин між об'єктами, модель взаємодії об'єктів;
- Діаграми переходів або сценарії життя суті.

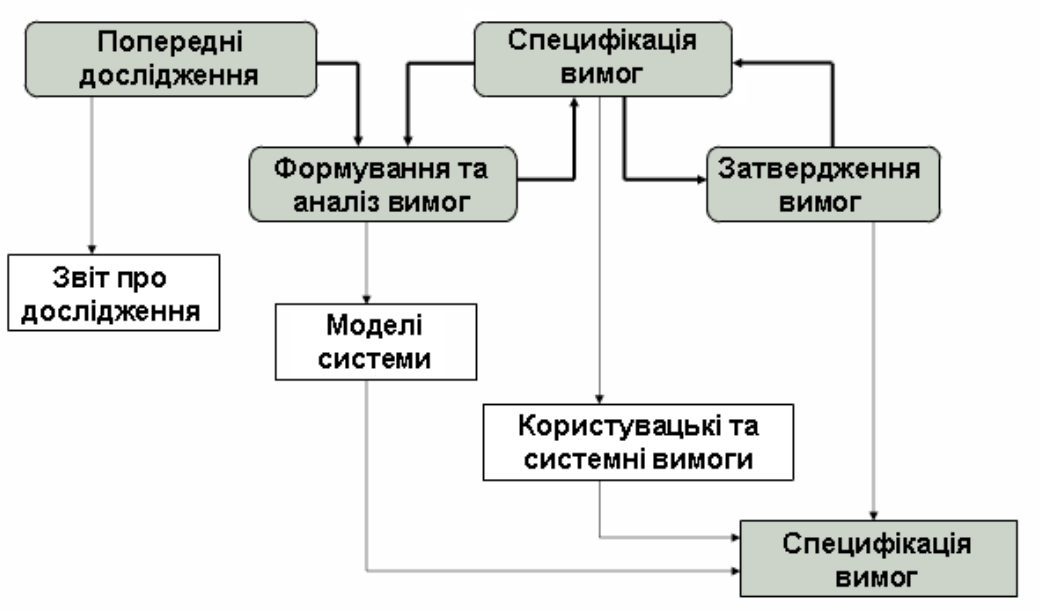

Рис. 1.8 Процес визначення специфікації вимог

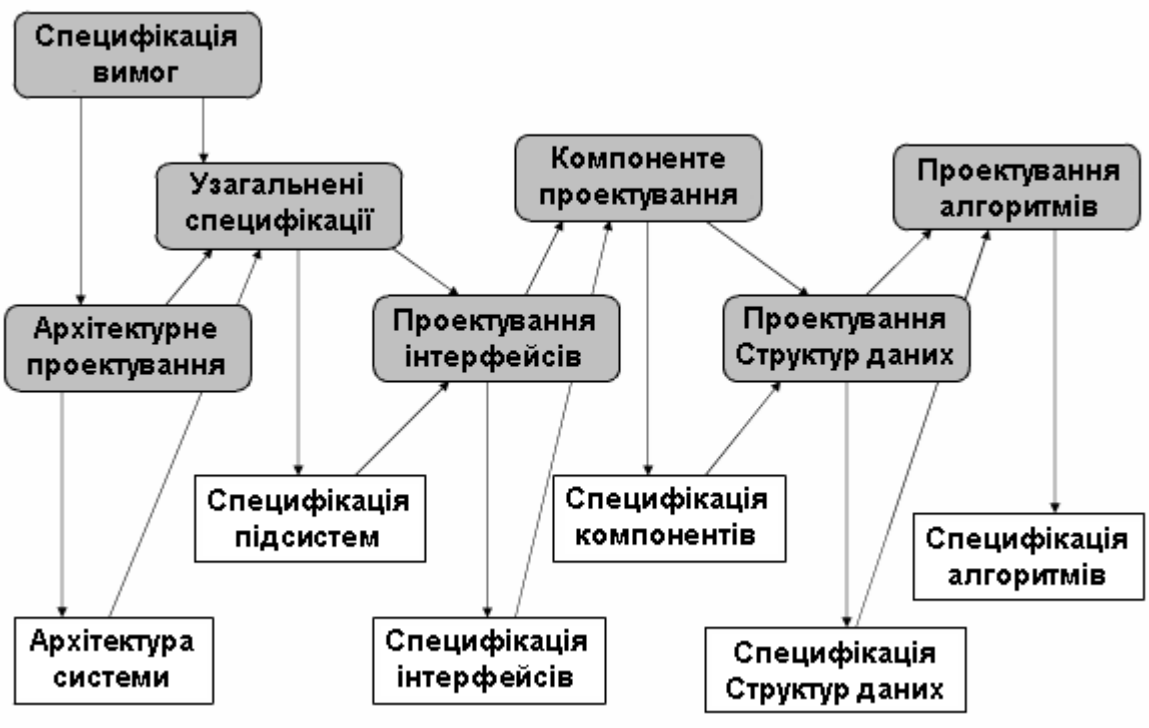

Рис. 1.9 Процес формування специфікації вимог

Програмування, тестування і відладка

Програмування розглядається як кодування — реалізація у вигляді програми одного чи кількох взаємопов'язаних алгоритмів (у сучасних умовах це здійснюється з застосуванням мов програмування).

Тестування – це процес встановлення програмних помилок.

Відладка – це встановлення місцеположення помилок і їх усунення.

Структура цих процесів наведена на Рис. 1.10.

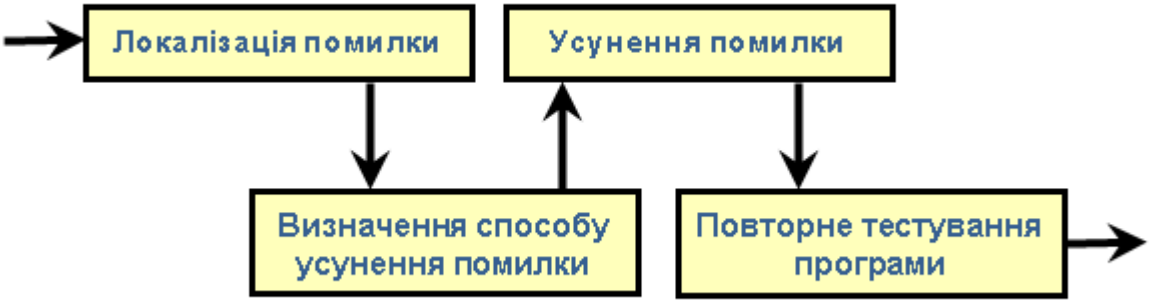

Рис. 1.10 Процес тестування та відладки ПЗ

Етапи тестування програмної системи наведені на Рис. 1.11.

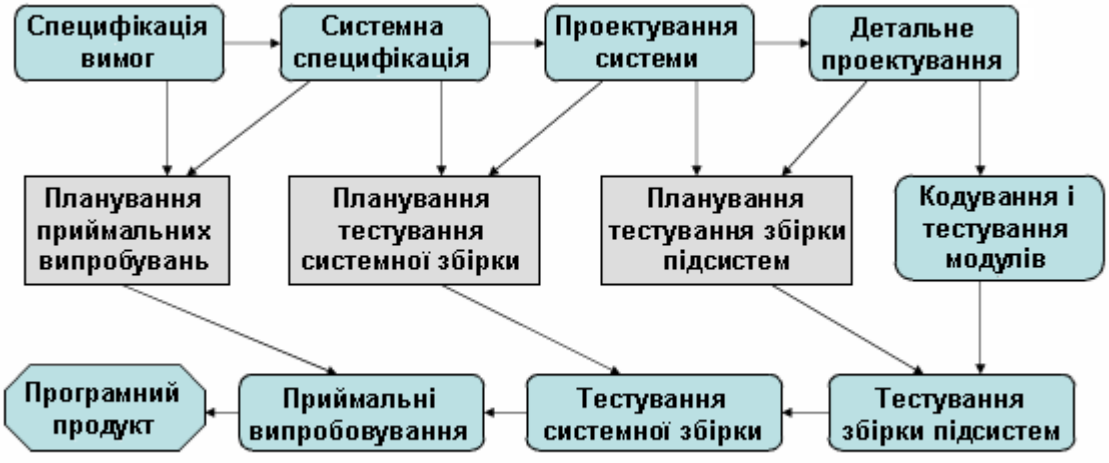

Рис. 1.11 Етапи тестування

Атестація і верифікація – це процес встановлення відповідності ПЗ її специфікації, а також очікуванням і вимогам користувачів і замовника (Рис. 1.12).

Рис. 1.12 Процес атестації і верифікації

Супровід системи – це внесення змін до системи, яка знаходиться в експлуатації (Рис. 1.13).

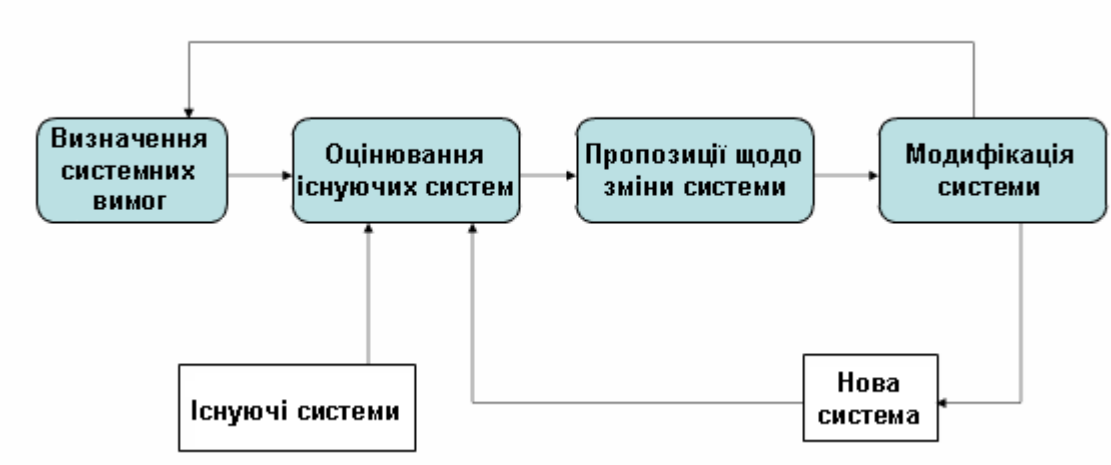

Рис. 1.13 Еволюція систем

Питання для обговорення

1. Чому в процесі визначення вимог необхідно розрізняти розробку вимог, які призначені для користувача, і розробку системних вимог?

2. Які п'ять основних компонентів будь-яких методів проектування?

3. Розробіть модель процесу тестування виконуваної програми.