**Лекція 8. Процеси розробки програмного забезпечення**



Рисунок 1 - Узгодження етапів ЖЦ за каскадною моделлю та стадії проектування ПЗ за ГОСТ 34.601-90

На наданому рис.1 показано, як узгоджуються етапи ЖЦ за каскадною моделлю та стадії проектування ПЗ за ГОСТ 34.601-90.

Створення ПЗ - це сукупність процесів, що призводять до створення програмного продукту. Ці процеси ґрунтуються головним чином на технологіях інженерії програмного забезпечення. Існує 4 фундаментальних процеси, які притаманні будь-якому проекту створення ПЗ.

1. Розробка специфікації вимог на програмне забезпечення. Вимоги визначають функціональні характеристики системи і обов'язкові для виконання.

2. Створення програмного забезпечення. Розробка і створення ПЗ згідно специфікації на нього.

3. Атестація програмного забезпечення . Створене ПЗ повинно пройти атестацію для підтвердження відповідності вимогам замовника.

4. Удосконалення (модернізація) програмного забезпечення. ПЗ повинно бути таким, щоб його можна було модернізувати згідно зміненим вимогам споживача.

При виконанні різноманітних програмних проектів ці процеси можуть бути організовані різними способами і описані на різних рівнях деталізації. Тривалість реалізації цих процесів також далеко не завжди однакова. І взагалі, різні організації, що займаються виробництвом ПЗ, часто використовують різні процеси для створення програмних продуктів навіть одного типу. З іншого боку, певні процеси більш підходять для створення програмних продуктів одного типу і менше - для іншого типу програмних додатків. Якщо використовувати невідповідний процес, це може призвести до зниження якості та функціональності розроблюваного програмного продукту.

Базовим стандартом щодо визначення процесів ЖЦ ПЗ є ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмних засобів. В стандарті всі процеси поділяються на три групи: основні, допоміжні, організаційні.



Рисунок 1 – Групи процесів за стандартом ISO 12207

У загальному випадку життєвий цикл визначається моделлю і описується в формі методології (методу). Модель життєвого циклу визначає концептуальний погляд на його організацію і нерідко - основні фази життєвого циклу і принципи переходів між ними. Модель життєвого циклу ПЗ - це структура, що визначає послідовність виконання і взаємозв'язку процесів, дій і завдань протягом ЖЦ. Модель ЖЦ ПЗ включає в себе стадії, результати виконання робіт на кожній стадії, ключові події - точки завершення робіт і прийняття рішень.

Поняття моделі ЖЦ процесу розробки продукту більш вузьке порівняно з поняттям моделі ЖЦ самого продукту, оскільки конкретний процес розробки відповідає певній частині життєвого циклу продукту (рис. 2).

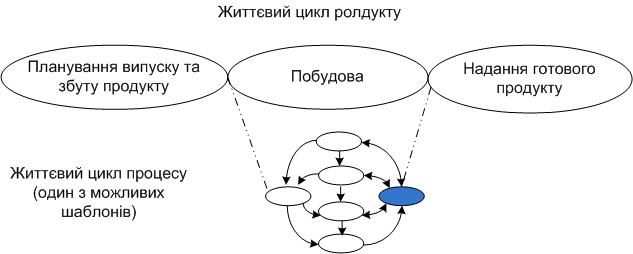


Рисунок 2 – Співвідношення життєвих циклів продукту та процесу розробки

Згідно спрощеної моделі, розробка починається з встановлення вимог предметної області та збору інформації про неї, а закінчується тестуванням і подальшим супроводом. Між ними здійснюються три основні етапи (рис. 3):

• визначення технічних вимог і логічне моделювання (аналіз);

• архітектурне моделювання (проектування);

• реалізація (кодування і тестування).

Ця спрощена модель розробки програмного забезпечення допускає ітерації і створення прототипів. У реальному житті процеси визначення технічних вимог і проектування в значній мірі перекриваються. Це особливо вірно для об'єктно-орієнтованого проектування та аналізу, оскільки абстракції для них моделюються швидше на основі абстракцій додатку, а не процесорів і дисків.



Рисунок 3 – Основні етапи спрощеного процесу розробки програмного забезпечення

Як відомо, проектування може бути розділене на логічне і фізичне. При об'єктно-орієнтованому проектуванні логічна стадія часто не відрізняється від деяких частин об'єктно-орієнтованого аналізу.

Однією з головних проблем, з якими стикаються при структурному аналізі і методах проектування, є відсутність перетину або плавного переходу між двома стадіями. Це часто призводить до труднощів у відстеженні продуктів проектування в зворотному напрямку до первинних вимог користувача або продуктам аналізу.

Для об'єктно-орієнтованого аналізу і об'єктно-орієнтованих методів проектування загальними є такі основні етапи, хоча їх порядок і деталі реалізації в значній мірі варіюються:

* Пошук способів, за допомогою яких система взаємодіє зі своїм оточенням (прецеденти).
* Ідентифікація об'єктів і імен їх атрибутів і методів
* Визначення зв'язків між об'єктами.
* Визначення інтерфейсу(ів) кожного об'єкта, а також обробки виняткових ситуацій.
* Реалізація та тестування об'єктів.
* Компонування і тестування системи.

Аналіз - це розбиття завдання на складові. В обробці даних це розуміється як процес специфікації структури і функцій системи, незалежно від засобів реалізації або фізичного розбиття на модулі або компоненти. Аналіз традиційно виконувався зверху вниз з використанням структурного підходу або еквівалентного методу, що базується на функціональній розбивці і аналізі даних.

При цьому стратегічний аналіз високого рівня, керований виробничими цілями, ізолюється від системного. При об'єктному підході здійснюється аналіз обох типів (функціональний і системний). Це можливо, оскільки об'єктно орієнтований аналіз дозволяє описувати систему в характеристиках реального світу, і системні абстракції більш-менш точно відповідають сутностей бізнес-процесу.

Об'єктно-орієнтований аналіз - це аналіз, який містить і елемент синтезу. Формулювання вимог користувача і ідентифікація ключових об'єктів предметної області супроводжуються компонуванням цих об'єктів в структури, які на більш пізній стадії складуть основу фізичного проектного рішення.

Аспект синтезу присутній обов'язково, оскільки аналізується система; іншими словами, на предметну область накладається певна структура. Це не означає, що проектне рішення не буде уточнюватися. Добре деталізований проект може мати відчутні відмінності від лаконічною моделі специфікації.

«Ідеального» процесу створення програмного забезпечення не існує. Слід зазначити, що певні процеси більш підходять для створення програмних продуктів одного типу і менше - для іншого типу програмних додатків. Якщо використовувати невідповідний процес, це може привести до зниження якості і функціональності розроблюваного програмного продукту.

Важливо підкреслити, що можливо удосконалення обраного в якості основи процесу, перш за все за рахунок використання сучасних технологій і включення кращих методів сучасної інженерії програмного забезпечення.

За ГОСТ 34.601-90 безпосередньо розробка ПЗ відноситься до стадії "6. Робоча документація", яка має два етапи " 6.1. Розробка робочої документації на систему і її частини." та "6.2. Розробка або адаптація програм."

До складу підсистем, що забезпечують функціонування інформаційної системи, звичайно входять такі базові підсистеми:

* інформаційне забезпечення – методи та засоби побудови інформаційної бази системи, що включає системи класифікації та кодування інформації, уніфіковані системи документів, схеми інформаційних потоків, принципи та методи створення баз даних;
* апаратне забезпечення – комплекс технічних засобів, задіяних в технологічному процесі перетворення інформації в системі. У першу чергу це обчислювальні машини, периферійне устаткування, апаратура та канали передачі даних;
* програмне забезпечення – включає в себе сукупність програм регулярного застосування, необхідних для вирішення функціональних завдань, і програм, що дозволяють найбільш ефективно використовувати обчислювальну техніку, забезпечуючи користувачам найбільші зручності в роботі;
* математичне забезпечення – сукупність математичних методів, моделей і алгоритмів обробки інформації, використовуваних у системі;
* лінгвістичне забезпечення – сукупність мовних засобів, що використовуються в системі з метою підвищення якості її розробки і полегшення спілкування людини з машиною.

Таким чином розробка програмного забезпечення (ПЗ) є однією зі складових побудови будь-якої інформаційної системи. Таким чином розробка програмного забезпечення (ПЗ) є однією зі складових побудови будь-якої інформаційної системи.

Процес розробки ПЗ тісно пов’язаний з поняттям «технологія програмування». Технологія програмування – це система методів, способів і прийомів розробки й налагодження комп’ютерних програм. Традиційне розуміння технології програмування (programming technology) – сукупність виробничих процесів, результатом яких є створення необхідного ПЗ, а також опис цієї сукупності процесів. В широкому значенні - це технологія розробки програмних засобів, включаючи в неї всі процеси, починаючи з моменту зародження ідеї розробки, тобто постановки задачі фахівцем з предметної галузі, яка пов'язана зі створенням необхідної програмної документації, до моменту впровадження та супроводження програмного засобу.

Задача виникає в певній предметній області та формулюється фахівцем з предметної області. Для автоматизації розв’язання задачі розробником програм – спеціалістом з інформатики та програмування створюється комп’ютерна програма. Фахівець з предметної області формулює задачу в термінах предметної області. Він може не враховувати (йому можуть бути очевидні) ряд моментів, потрібних розробникові програми. Аналогічно розробник програми може по-своєму тлумачити деякі поняття, потребувати додаткових даних, тобто може бути неадекватне тлумачення задачі.

Тому розробка програми складається з ряду етапів. Зазвичай у ЖЦ ПЗ виділяють такі етапи: формалізація задачі, метод розв’язання задачі, зовнішня специфікація програми, проектування програми, кодування програми, тестування та налагодження програми і, насамкінець, використання програми. Етапи проектування, кодування, тестування і налагодження складають власне розробку програми.

**ЕТАП 1. Формалізація задачі**

Уся множина даних предметної області має бути відображена належним чином у формалізованій задачі. Розробник ПЗ повинен проаналізувати задачу, виявити і визначити всі необхідні  для неї дані, вияснити тлумачення неоднозначних понять тощо.

Треба визначитись з вхідними даними, виявити і явно указати дані, які завуальовані або допускають неоднозначне тлумачення. Наприклад, у задачі сказано, що треба скласти список осіб, молодших заданого віку. Тут  мається на увазі дата, на яку треба видати інформацію. Тобто до даних, що задаються в задачі, треба додати дату.

Треба визначитись з шуканими результатами задачі. Якщо результат має складну структуру, наприклад таблицю, то однієї назви буде замало, бажано дати пояснення: в графічній формі – ескіз таблиці та описати порядок формування та розрахунку показників

Задано: перерахувати всі дані, яких вимагає задача.

Одержати: усі дані, що є результатом розв’язання задачі.

**ЕТАП 2. Метод розв’язання задачі**

Якщо задача може бути розв’язана декількома методами, то вибирається один з методів. Якщо ж відомих методів немає чи вони не влаштовують постановника або розробника, то метод створюється постановником або розробником програми, при цьому може бути запропоновано декілька методів. Методи розглядаються, аналізуються і вибирається найліпший з погляду розробника. Метод кладеться в основу майбутнього алгоритму. Тому його формулювання має бути алгоритмоподібним. Якщо на наступних етапах стане ясно, що вибраний метод не підходить, то треба почати спочатку і вибрати інший метод.

**ЕТАП 3. Специфікація програми**

Майбутня програма розглядається як єдине ціле. Визначається, ЩО вона повинна робити, не розглядаючи, ЯК це робиться. Це значить, що визначаються функціональне призначення програми, вхідні дані, вихідні дані. Для цього формалізована постановка задачі переписується алгоритмічними (програмними) засобами. Визначається мова програмування і специфікуються дані та функції засобами цієї мови. Якщо дані мають складну структуру (наприклад, таблиця), то цю структуру (власне тип як шаблон даних) доцільно зобразити графічно з визначенням усіх деталей. Тобто здійснюється формальний однозначний опис задачі, ліквідуються (уточнюються) недоліки, які могли бути допущені в постановці задачі.

Визначення специфікацій – це визначення виконавця відповідного рівня, що відповідає визначенню модуля відповідного рівня. Таким чином, фактично паралельно розглядаються задача, виконавець, програмний модуль (алгоритм) відповідного рівня.

У специфікації бажано мати підсумковий запис у вигляді:

* Вхідні дані : усі вхідні дані цього рівня.
* Вихідні дані : усі вихідні дані цього рівня.
* Функціональне призначення: перелік функцій або опис дій цього рівня.
* Алгоритм: ім’я алгоритму.

Окрім цього, у зовнішній специфікації вказуються мова програмування, середовище (операційна система) розробки та використання, технічні засоби (комп’ютери), швидкодія, користувач (кваліфікація), частота використання тощо.

**ЕТАП 4.     Проектування**

Реалізація ПЗ – це процес переведення системної специфікації в працездатну систему. Етап реалізації завжди включає процеси проектування та програмування, але якщо для розробки ПЗ застосовується еволюційний підхід, етап реалізації також може включати процес внесення змін до системну специфікацію.

Проектування полягає у створенні:

* архітектури ПЗ;
* модульної структури ПЗ;
* алгоритмічної структури ПЗ – детально розробляються алгоритми, призначені для реалізації системних сервісів;
* структури даних – детально розробляються структури даних, необхідні для реалізації програмної системи;
* вхідного і вихідного інтерфейсу (вхідних та вихідних форм даних) – для кожної підсистеми визначається і документується її інтерфейс.

Вихідні дані для проектування містяться у специфікації, тобто виконується трансляція вимог до ПЗ у безліч проектних уявлень. При вирішенні завдань проектування основна увага приділяється якості майбутнього програмного продукту.

Проектування алгоритму (програми) розв’язання задачі – це створення структури алгоритму і даних, а також реалізацію алгоритму  за допомогою засобів зображення, тобто створення проекту програми.

Існують різні методи проектування: зверху-вниз, знизу-вверх, розширення ядра тощо.

Процес проектування творчий, а не формальний. Виявлення функцій-дій та структур даних не є детермінованим, однозначним. Може бути декілька проектів. Не виключена можливість того, що на певному кроці розробки програми виявиться хибність вибраного підходу, ми змушені будемо повернутись на попередні кроки чи навіть починати все спочатку. Тобто, процес проектування і взагалі розробки програми має ітеративний характер, коли на базі набутих знань необхідно повернутись до початку і прийняти більш точне рішення.

При проектуванні алгоритму виникає запитання – коли і де зупинитись. Зупинятись треба тоді, коли нам ясно, що виконавець (модуль), який розглядається, може без труднощів бути реалізований за скінченне число кроків вибраною мовою програмування.

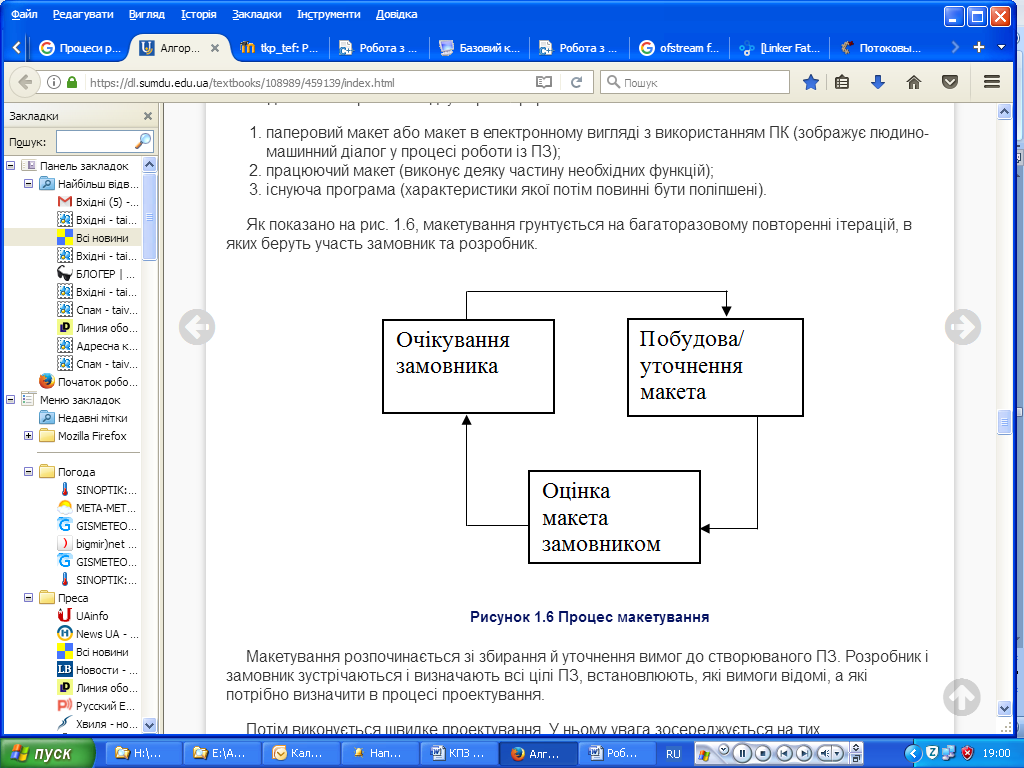
При розробці програми ми повинні забезпечити читабельність алгоритму і програми. Читабельність буде найкращою при умові, що алгоритм чи програма вміщуються на одній сторінці. При проектуванні алгоритму треба старатись виділяти по можливості крупні дії, які, з одного боку, приводили б до простіших задач і , з іншого - давали б можливість реалізувати виконавця вищого рівня.

Досить часто замовник не може сформулювати докладні вимоги щодо маніпулювання даними за допомогою майбутнього програмного продукту, тобто передбачити найоптимальніші механізми введення, обробки та виведення даних.

З іншого боку, розробник може сумніватися, чи пристосується продукт під операційну систему замовника, а також у формі діалогу з користувачем або в ефективності реалізованого алгоритму. У цих випадках доцільно використовувати макетування.

Основна мета макетування (прототипування)– зняти невизначеності у вимогах замовника. Це процес створення моделі необхідного програмного продукту.

Модель може приймати одну з трьох форм:

1. паперовий макет або макет в електронному вигляді з використанням ПК (зображує людино-машинний діалог у процесі роботи із ПЗ);
2. працюючий макет (виконує деяку частину необхідних функцій);
3. існуюча програма (характеристики якої потім повинні бути поліпшені).

Як показано на рис. 4, макетування грунтується на багаторазовому повторенні ітерацій, в яких беруть участь замовник та розробник.

Рисунок 4 Процес макетування

Макетування розпочинається зі збирання й уточнення вимог до створюваного ПЗ. Розробник і замовник зустрічаються і визначають всі цілі ПЗ, встановлюють, які вимоги відомі, а які потрібно визначити в процесі проектування.

Потім виконується швидке проектування. У ньому увага зосереджується на тих характеристиках ПЗ, які повинні бути видимі користувачеві. Швидке проектування призводить до побудови макета. Макет оцінюється замовником і використовується для уточнення вимог до ПЗ. Ітерації повторюються до тих пір, поки макет не виявить всі вимоги замовника і, тим самим, не дасть можливість розробнику зрозуміти, що повинно бути зроблено.

Переваги використання макетування: забезпечує визначення повних вимог до ПЗ.

Недоліки макетування:

* замовник може прийняти макет за продукт;
* розробник може прийняти макет за продукт.

**ЕТАП 5. Програмування (кодування)**

На даному етапі створений проект реалізується вибраною мовою програмування. Якщо вибрана мова програмування підтримує використані в проекті  алгоритму структури керування та структури даних, то цей процес стає майже механічним перекладом (тому його часто називають кодуванням). Якщо ж ці структури не підтримуються, то їх треба моделювати наявними у мові структурами. У цьому випадку робота буде складнішою. Тут важливим фактором є добре знання конкретної мови програмування.

**ЕТАП 6. Тестування і налагодження програми**

На кожному кроці розробки алгоритму і програми можуть бути допущені помилки. Причини помилок різноманітні: неадекватне розуміння задачі, неправильно складені специфікації, невдале проектування, погане знання мови програмування, механічні помилки тощо. Найсерйознішими є помилки, зроблені на перших кроках. Тому створену програму треба довести до робочого стану – відтестувати та налагодити. Це робиться за допомогою спеціальних тестових даних.  Тестові дані – це вхідні і вихідні дані, підібрані так, щоб для заданих вхідних даних були відомі вихідні. Для вхідних даних програма (модуль), що перевіряється, видає свої вихідні дані. Якщо результат роботи програми не збігається з тестовими  даними, то це значить, що в програмі є помилка; якщо ж збігається, то це не гарантує відсутність помилки -  це значить, що на цих вхідних тестових даних помилка не проявилась. При наявності помилки програму треба налагодити. Для цього необхідно виявити причину  і віднайти місце виникнення помилки, після чого причину і помилку ліквідувати.

Тестування та налагодження програми – складна і трудомістка робота. Необхідно відтестувати роботу кожного модуля, усі передачі керування між модулями. Це найпростіше можна зробити за умови, що проектування і програмування  виконувались з використанням структур керування та структур даних. Для кожного модуля треба перевірити, чи реалізує він ті функції, які сформульовані в специфікації модуля. Тобто, потрібно перевірити, ЩО робить модуль, не розглядаючи того,  ЯК  це робиться. Треба перевірити логіку (алгоритм) модуля:  відтестувати усі операції, усі передачі керування, тобто перевірити усі шляхи потоків інформації. Цим самим перевіряється, як реалізується функція модуля, що визначена в специфікації модуля. Ця робота найпростіше виконується для структурної програми.

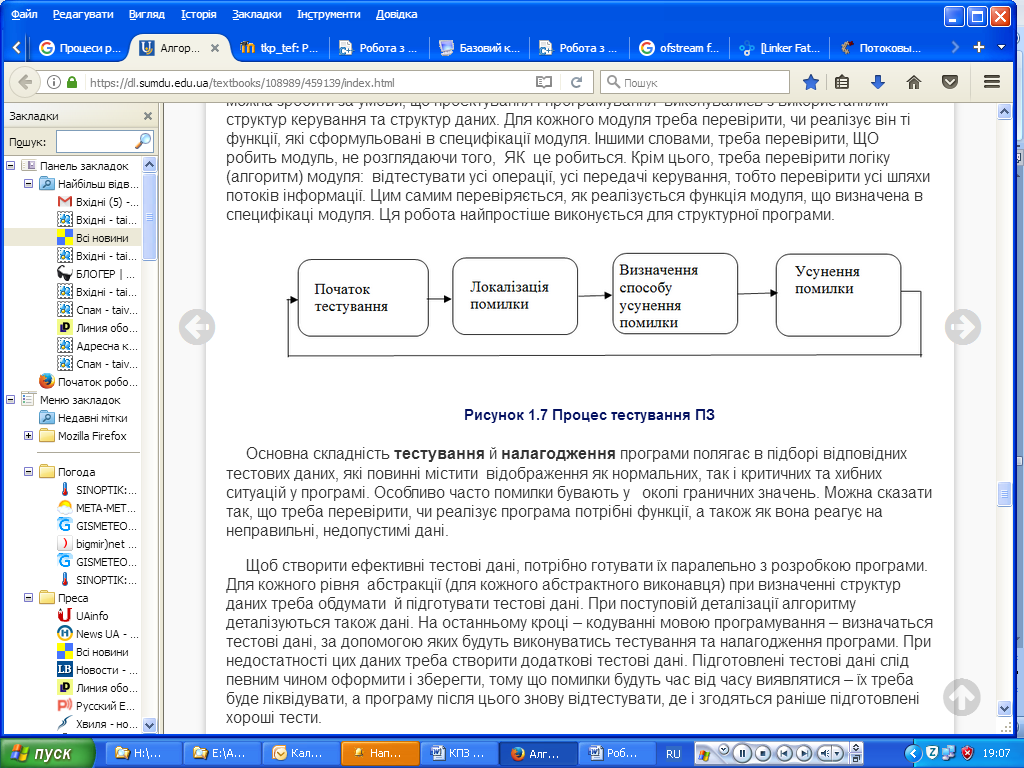


Рисунок 5 Процес тестування ПЗ

Основна складність тестування й налагодження програми полягає в підборі відповідних тестових даних, які повинні містити  відображення як нормальних, так і критичних та хибних ситуацій у програмі, тобто перевірити, чи реалізує програма потрібні функції, а також як вона реагує на неправильні, недопустимі дані. Особливо часто помилки бувають у   околі граничних значень.

Щоб створити ефективні тестові дані, потрібно готувати їх паралельно з розробкою програми. Для кожного рівня  абстракції (для кожного абстрактного виконавця) при визначенні структур даних треба обдумати  й підготувати тестові дані. При поступовій деталізації алгоритму деталізуються також дані. На останньому кроці – кодуванні мовою програмування – визначаться тестові дані, за допомогою яких будуть виконуватись тестування та налагодження програми. При недостатності цих даних треба створити додаткові тестові дані. Підготовлені тестові дані слід певним чином оформити і зберегти, тому що помилки будуть час від часу виявлятися – їх треба буде ліквідувати, а програму після цього знову відтестувати, де і згодяться раніше підготовлені хороші тести.

Існують різні методи тестування, які будуть розглядатися окремо.

Налагодження програми. Налагодження є наслідком успішного тестування - якщо тестовий варіант виявляє помилку, то процес налагодження знищує її. Процесу налагодження передує виконання тестового варіанту. Його результати оцінюються, реєструється невідповідність між очікуваним і реальним результатами. Процес налагодження намагається зіставити симптом з причиною, внаслідок чого призводить до виправлення помилки. Можливі два результати процесу налагодження:

1. причина знайдена, виправлена, знищена;
2. причина не знайдена.

У другому випадку відладчик може припускати причину. Для перевірки цієї причини він просить розробити додатковий тестовий варіант, який допоможе перевірити припущення. Запускається ітераційний процес корекції помилки.

Можливі різні способи прояву помилок:

1. програма завершується нормально, але видає неправильні результати;
2. програма зависає;
3. програма завершується з переривання;
4. програма завершується, видає очікувані результати, але збережені дані зіпсовані (це самий неприємний варіант).

Розрізняють дві групи методів налагодження:

* аналітичні;
* експериментальні.

Аналітичні методи базуються на аналізі вихідних даних для тестових прогонів. Експериментальні методи базуються на використанні допоміжних засобів налагодження (налагоджувальні друку, трасування), що дозволяють уточнити характер поведінки програми при тих чи інших вихідних даних.

Загальна стратегія налагодження – зворотне проходження від поміченого симптому помилки до вихідної аномалії (місцем в програмі, де помилка здійснена).

У простому випадку місце прояви симптому і помилковий фрагмент збігаються. Але найчастіше вони далеко відстоять один від одного.

Мета налагодження – знайти оператор програми, при виконанні якого правильні аргументи приводять до неправильних результатів. Якщо місце прояви симптому помилки не є шуканої аномалією, то один з аргументів оператора повинен бути невірним. Тому треба перейти до дослідження попереднього оператора, який виробив цей невірний аргумент. У підсумку покрокове зворотне простежування призводить до шуканого помилкового місця.

У різних методах простежування організовується по-різному. В аналітичних методах – на основі логічних висновків про поведінку програми.

Основна перевага аналітичних методів налагодження полягає в тому, що вихідна програма залишається без змін.

В експериментальних методах для простежування виконується:

1. Видача значень змінних у зазначених точках.
2. Трасування змінних (видача їх значень при кожній зміні).
3. Трасування потоків управління (імен процедур, що викликаються, міток, на які передається управління, номерів операторів переходу).

Перевага експериментальних методів налагодження полягає в тому, що основна рутинна робота з аналізу процесу обчислень перекладається на комп'ютер. Багато трансляторів мають вбудовані засоби налагодження для отримання інформації про хід виконання програми.

Недолік експериментальних методів налагодження – в програму вносяться зміни, при виключенні яких можуть з'явитися помилки. Втім, деякі системи програмування створюють спеціальний налагоджувальний екземпляр програми, а в основній екземпляр не втручаються.

**ЕТАП 7.  Використання. Супроводження**

Налагоджене ПЗ використовується для розв’язання задачі. Отримані результати аналізуються і, у випадку їх правильності, використовуються в практиці. При тривалому використанні ПЗ виникає потреба у його модифікації. Це можуть бути такі дії: заміна одних модулів (функцій) іншими, підключення нових модулів,  ліквідація виявлених помилок тощо. Ці дії виконуються в рамках супроводження ПЗ, хоча по суті справи це подальший розвиток програми. Труднощі полягають не тільки в необхідності виконання певного обсягу роботи, але й у тому, що цю роботу можуть виконувати не розробники програми, а інші люди, яким потрібно розібратись у тому, що зроблено попередниками, і внести зміни в програму, не порушуючи загальної концепції.  Ця робота може бути найбільш надійно виконана при хорошому проектуванні програми. А саме: повинна бути визначена структура програми, модулі мають бути функціонально чітко визначеними  і незалежними, документація має відповідати дійсному стану справ. Це забезпечить найпростішу заміну одного модуля іншим або підключення нових модулів. Після такої модифікації програма повинна бути заново відтестована й налагоджена.

***Контрольні запитання*.**

1. Які існують процеси, притаманні будь-якому проекту створення ПЗ?
2. Які етапи ЖЦ складають власне розробку ПЗ?
3. В чому полягає формалізація задачі при розробленні ПЗ?
4. Як специфікуються програми?
5. Назвіть етапи реалізації ПЗ.
6. Як пов‘язане тестування та налагодження програм?

***Рекомендована література***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник: http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf
2. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
3. Проектування інформаційних систем: Посібник // За редакцією Пономаренка В.С. – К.: Видавничий центр "Академія". 2002. ­ 488 с. URL: <http://www.dut.edu.ua/uploads/l_874_10304054.pdf>
4. ДСТУ ISO/IEC 15288:2005. Інформаційні технології. Процеси життєвого циклу системи (ISO/IEC 15288:2002, IDT) - К.: Держстандарт України, 2005. - К.: Держстандарт України, 2004.
5. ДСТУ ISO/IEC/IEEE 12207:2018 Інформаційні технології. Процеси життєвого циклу програмного забезпечення - К.: Держстандарт України, 2018.

***Для самостійного вивчення***: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.