**Лекція 11. Базова термінологія**

**Основні визначення.**

***Тестування* -** складова процесу програмної інженерії, один з методів подальшого поліпшення якості розробленого програмного забезпечення системи за допомогою виявлення дефектів, що залишилися, не виявлених раніше всіма іншими видами перевірок.

**Тестування -** процес *виконання* програмної системи (або її елементів) з метою *перевірки її відповідності встановленим вимогам* і *виявлення дефектів.*

Тестування програмних систем складається з динамічної верифікації поведінки програм на кінцевому (обмеженому) наборі тестів (set of test cases), обраних відповідним чином з зазвичай виконуваних дій прикладної області і забезпечують перевірку відповідності очікуваної поведінки системи.

У даному визначенні тестування виділені слова, що визначають основні питання, яким адресується дана область знань:

***Динамічність (dynamic)***: цей термін означає тестування завжди припускає до виконання програми, що тестується із заданими вхідними даними. При цьому, величини, що задаються на вхід тестуються, програмного забезпечення, не завжди достатньо для визначення тесту. Складність і недетермінованість систем призводить до того, що система може по-різному реагувати на одні й ті ж вхідні параметри, в залежності від стану системи. У даній галузі знань термін "вхід" (input) буде використовуватися в рамках угоди про те, що вхід може також специфікувати стан системи, в тих випадках, коли це необхідно. Крім динамічних технік перевірки якості, тобто тестування, існують також і статичні техніки, що розглядаються в галузі знань "Software Quality".

***Обмеженість (finite):*** навіть для простих програм теоретично можлива настільки велика кількість тестових сценаріїв, що вичерпне тестування може зайняти багато місяців і навіть роки. Саме тому, з практичної точки зору, всебічне тестування вважається нескінченним. Тестування завжди передбачає компроміс між обмеженими ресурсами і заданими термінами, з одного боку, і практично необмеженими вимогами з тестування, з іншого.

***Вибір (selection):*** багато пропонованої техніки тестування відрізняються один від одного в тому, як вибираються сценарії тестування. Інженери з програмного забезпечення повинні володіти уявленням про те, що різні критерії вибору тестів можуть давати різні результати, з точки зору ефективності тестування. Визначення відповідного набору тестів для заданих умов є дуже складною проблемою. Звичайно, для вибору відповідних тестів спільно застосовують техніки аналізу ризиків, аналіз вимог і відповідну експертизу в області тестування і заданої прикладної області.

***Очікувана поведінка (expected behaviour):*** хоча це не завжди легко, все ж таки необхідно вирішити, яка поведінка програми буде прийнятною, а яка — ні. В іншому випадку, зусилля з тестування — марні. Спостережувана поведінка може розглядатися в контексті користувача очікувань (маючи на увазі "тестування для перевірки" — testing for validation), специфікації ( "тестування для атестації" — testing for verification) або, нарешті, в контексті передбачуваної поведінки на основі неявних вимог або обгрунтованих очікувань. Загальний погляд на тестування програмного забезпечення останні роки активно еволюціонував, стаючи все більш конструктивним, прагматичним і наближеним до реалій сучасних проектів розробки програмних систем. Тестування більш не розглядається як діяльність, що починається тільки після завершення фази конструювання.

**Мета тестування: *основна* –** виявити дефекти і перевірити функціональну придатність**; *додаткова* –** перевірити зручність застосування та користування, продуктивність тощо.

**Стандарти**

# ДСТУ 2844-94 Програмні засоби забезпечення якості. Терміни та визначення

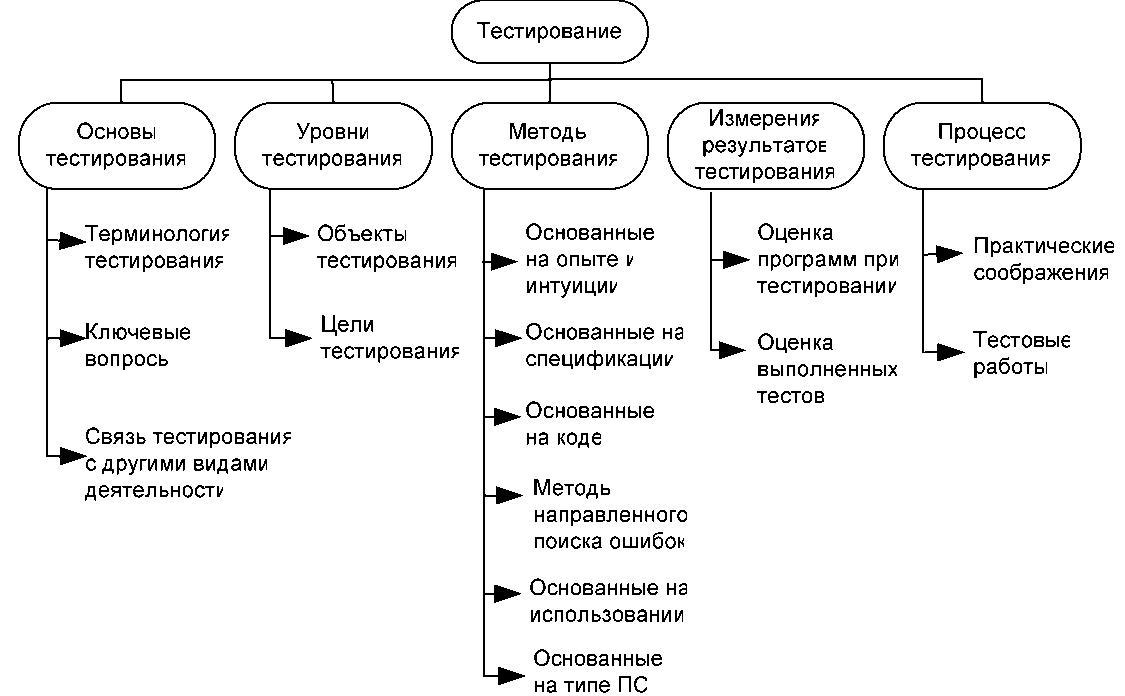
ДСТУ 2853-94 Програмні засоби ЕОМ. Підготовлення і проведення випробувань

**Основні підходи до виконання тестування**

* деструктивний (негативне, руйнівне тестування)
* конструктивний (позитивне або демонстраційне).

***Ключові питання тестування в* SWEBOK.**

* + *Критерії вибору тестів/Критерії адекватності тестів*
  + *Ефективність тестування/Мети тестування*
  + *Тестування для виявлення дефектів*
  + *Проблема оракула.*
  + *Теоретичні й практичні обмеження тестування.*
  + *Проблема нездійсненних шляхів.*
  + *Тестопридатність*

Рисунок 1. Область знань «Тестування ПЗ» в SWEBOK-2004

***Зв'язок тестування з іншими видами діяльності***

Тестування тісно пов‘язане з процесами верифікації і валідації.

* Верифікація — перевірка, перевіряємість, спосіб підтвердження, перевірка за допомогою доказів, будь-яких теоретичних положень, алгоритмів, програм та процедур *шляхом їх зіставлення* з еталонними чи емпіричними даними, алгоритмами та програмами; підтвердження відповідності кінцевого продукту наперед визначеним еталонним вимогам. Верифікація — проводиться практично завжди, виконується методом перевірки (звірення) характеристик продукції з заданими вимогами, результатом є висновок про відповідність (чи невідповідності) продукції.
* Валідація - підтвердження на основі надання об'єктивних свідоцтв того, що вимоги, що призначені для конкретного використання чи застосування, виконані. Валідація — проводиться за необхідності, шляхом аналізу заданих умов застосування та оцінки відповідності характеристик продукції цим вимогам, результатом є висновок про можливість застосування продукції до конкретних вимог.

**Основні методи тестування.**

Зазвичай, для проведення тестування застосовуються методи структурного («білий ящик») та функціонального («чорний ящик») тестування.

*При функціональному тестуванні* вихідний код програми не доступний. Суть полягає в перевірці відповідності поведінки програми її зовнішній специфікації. Критерієм повноти тестування вважається перебір всіх можливих значень вхідних даних, що здійснити на практиці надзвичайно важко.

*При структурному тестуванні* текст програми відкритий для аналізу. Суть даного методу полягає в перевірці внутрішньої логіки ПЗ. Повним тестуванням у цьому випадку буде таке, що приведе до перебору всіх можливих варіантів передач керування програми. Число таких шляхів може досягати десятків тисяч, відповідно, виникає питання про створення тестів, що забезпечують дане покриття. Здійснити повне всеохоплююче тестування навіть простої програми вкрай важко, а часом і неможливо в силу обмеженості часу й ресурсів. Отже, необхідно мати певні критерії за якими мають обиратися контрольні приклади та критерії зупинки процесу тестування.

Структурне тестування, або тестування «білого ящика», - це методика аналізу вихідного коду програми. Існує три різновиди структурного тестування: тестування на основі потоку керування програми, на основі потоку даних та мутаційне тестування.

*При використанні тестування на основі потоку керування програми* тестується логіка програми, яка може бути представлена у вигляді графа керування: вершинами є оператори, а гілками - переходи між ними.

*При тестування на основі потоку даних* увага приділяється взаємозв'язкам між змінними. Виділяються вершини, у яких змінна ініціалізується та в яких використовується, і вивчаються переходи й взаємозв'язки між такими вершинами.

*Мутаційне тестування* полягає у внесенні несправностей у вихідний код програми та порівняння роботи вихідної програми та програми мутанта.

Оскільки здійснити вичерпне структурне тестування вкрай важко, необхідно вибрати такі критерії його повноти, які допускали б їхню просту перевірку й полегшували б цілеспрямований підбір тестів.

**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.

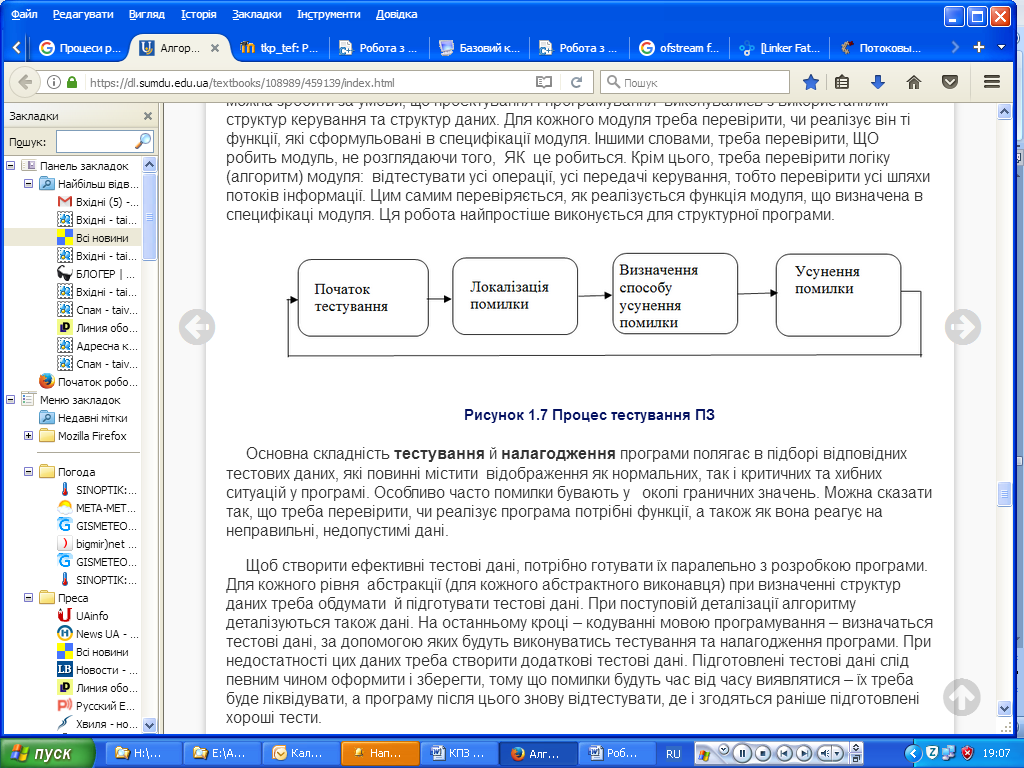


Рисунок 2. Процес тестування ПЗ

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.

**Види й рівні тестування**

* *Модульне (IEEE 1008-87 "Standard for Software Unit Testing“)*
* *Інтеграційне (ДСТУ 2941* Системи оброблення інформації. Розроблення систем. Терміни та визначення*)*
* *Тестування ПЗ*
* *Системне (ISO/IEC 12207):*
* *по цілям якості, (акцент на нефункціональні вимоги: надійність, стійкість, продуктивність, переносимость і ін.),*
* *зовнішнім інтерфейсам з іншими системами, середовищем, апаратним забезпеченням.*

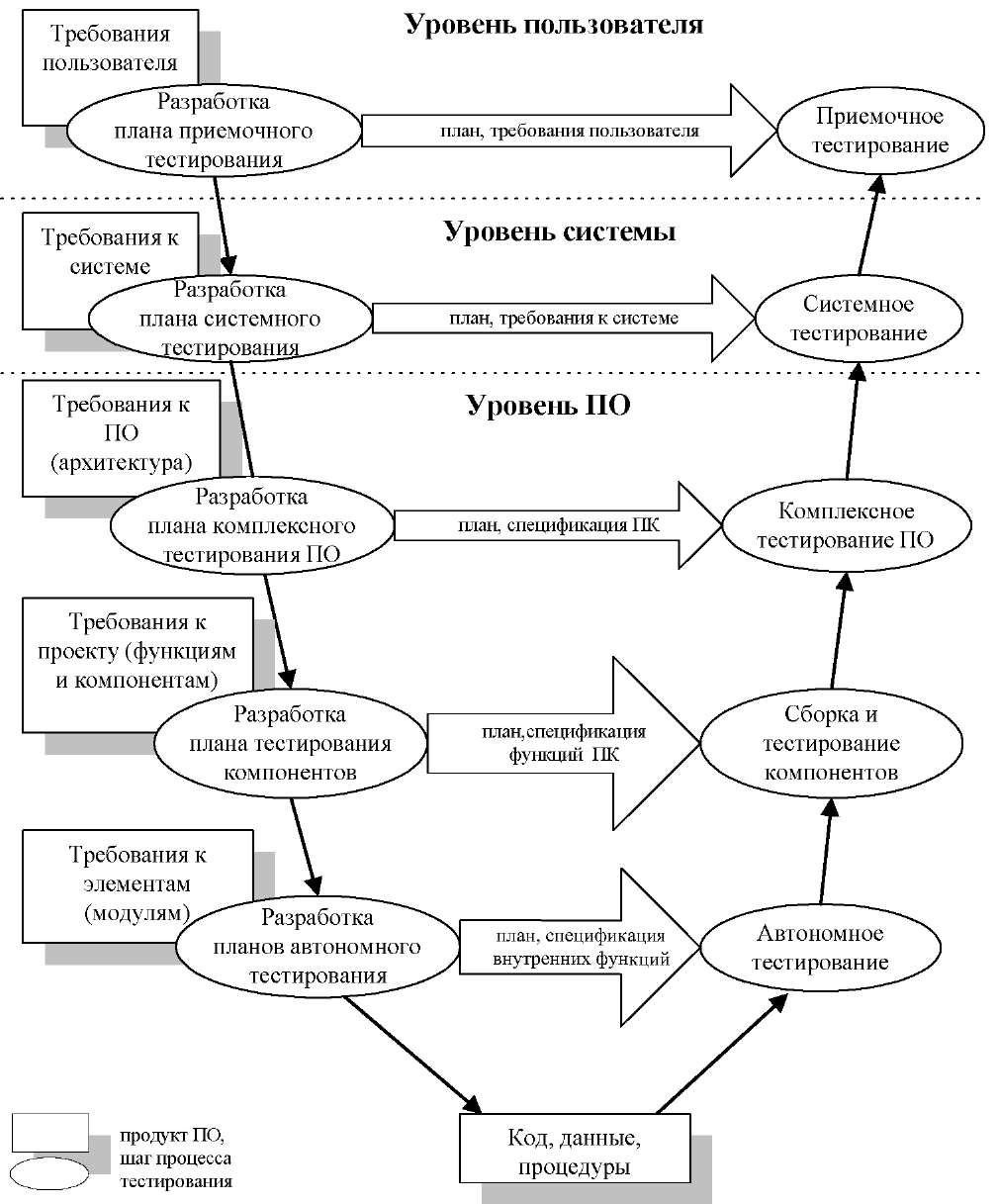
****

Рисунок 3 - Види й рівні тестування

**Види тестування.**

* *Функціональне тестування* (*на відповідність* або тестування *коректності)*
* *Тестування безпеки* (Security testing)
* *Тестування зручності застосування (ергономічності)*
* *Тестування технічних характеристик*
* *Тестування на надійність.*
* *Тестування продуктивності (Performance testing)* види:
  + *навантажувальне тестування (load testing)*
  + *тестування на стійкість (stress testing*);
  + *тестування обсягу (volume testing)* .
* *Тестування конфігурації (Configuration testing).*
* *Порівняльне тестування ( Back-to-back testing).*
* *Тестування відновлення (Recovery testing).*
* *Керована тестами розробка ( Test-driven development)*

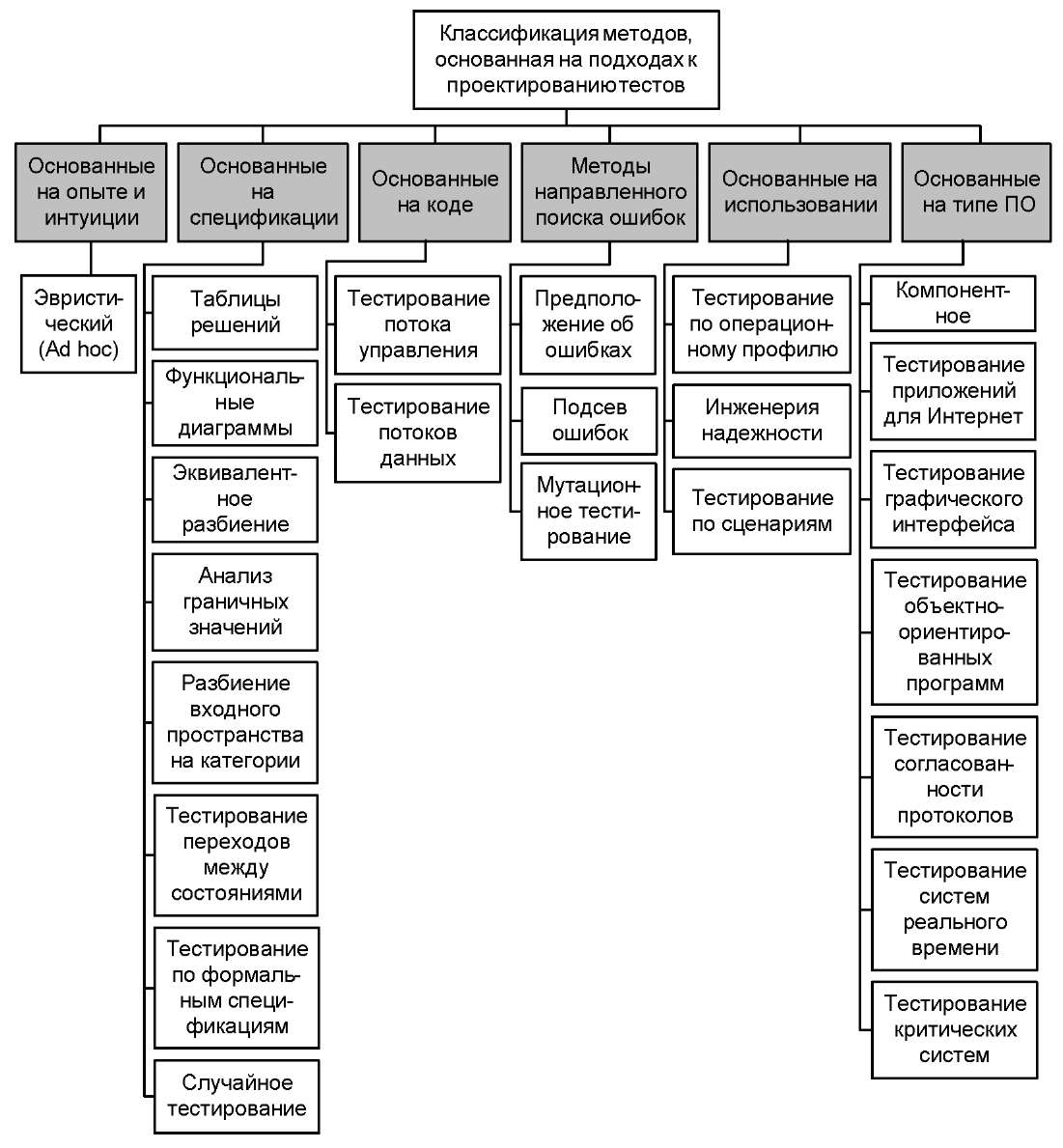


Рисунок 4 - Методи тестування

**Налагодження програми**. Налагодження є наслідком успішного тестування. Це означає, що якщо тестовий варіант виявляє помилку, то процес налагодження знищує її.

Отже, 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. Таким чином, запускається ітераційний процес корекції помилки.

Можливі різні *види 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 викликаються процедур, 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 транслятори 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.

**Поняття дефекту, збою та відмови**

У літературі, присвяченій програмній інженерії, зустрічається безліч термінів, що описують порушення функціонування програмних систем — недоліки (faults), дефекти (defects), збої (failures), помилки (errors) та ін. Відповідна термінологія описана в IEEE Std. 610—90. Важливо чітко розділяти причини порушення роботи прикладних систем, звичайно описувану термінами недолік або дефект, і як спостерігається небажаний ефект, що викликається цими причинами — збій. Термін помилка, залежно від контексту, може описувати і як причину збою, і сам збій. Тестування дозволяє виявити дефекти, що призводять до збоїв.

Необхідно розуміти, що причина збою не завжди може бути однозначно визначена. Не існує теоретичних критеріїв, що дозволяють гарантовано визначити, який саме дефект призводить до нинішнього збою.

**Типи дефектів, їх класифікація та статистика виникнення**

У літературі з тестування зустрічається велика кількість різних класифікацій дефектів. Ефективність тестування може бути досягнута в тому випадку, якщо ми розуміємо, які типи дефектів можуть бути знайдені в процесі тестування програмної системи та як змінюється їх частота в часі (маючи на увазі історичну перспективу розвитку системи, а не її збоїв у процесі роботи). Ця інформація дозволяє прогнозувати якість системи і допомагає вдосконалювати процес розробки, в цілому.

Міжнародний стандарт ANSI/IEEE-729-83 поділяє всі помилки в розробці програм на наступні типи.

**Помилка (error)** — стан програми, при якому видаються неправильні результати, причиною яких є вади (flaw) в операторах програми або в технологічному процесі її розробки, що призводить до неправильної інтерпретації вихідної інформації, отже, і до невірного рішення.

**Дефект (fault)** в програмі — наслідок помилок розробника на будь-якому з етапів розробки, яка може міститися у вихідних або проектних специфікаціях, текстах кодів програм, експлуатаційної документації тощо. У процесі виконання програми може бути виявлений дефект або збій.

**Відмова *(failure)*** — це відхилення програми від функціонування або неможливість програми виконувати функції, визначені вимогами та обмеженнями, що розглядається як подія, яке сприяє переходу програми в неробочий стан через помилки, прихованих в ній дефектів або збоїв у середовищі функціонування. Відмова може бути результатом наступних причин:

* помилкова специфікація або пропущена вимога, яка означає, що специфікація точно не відображає того, що припускав користувач;
* специфікація містить вимогу, яку неможливо виконати на цій апаратурі та програмному забезпеченні;
* проект програми містить помилки (наприклад, база даних спроектована без засобів захисту від несанкціонованого доступу користувача);
* програма може бути неправильною.

Таким чином, відмови, як правило, є результатами однієї або більше помилок у програмі, а також наявності різного роду дефектів.

Розглянемо процес тестування, виходячи з рекомендацій стандарту 180 / ІЕС 12207, і наведемо типи помилок, які виявляються на кожному процесі ЖЦ.

**Процес розробки вимог.** Під час визначення початкової концепції системи та вихідних вимог до системи виникають помилки аналітиків при специфікації верхнього рівня системи та побудові концептуальної моделі предметної області.

*Характерними помилками цього процесу є:*

* неадекватність специфікації вимог кінцевим користувачам;
* некоректність специфікації взаємодії ПЗ із середовищем функціонування або з користувачами;
* невідповідність вимог замовника до окремих і загальних властивостей ПЗ;
* некоректність опису функціональних характеристик;
* незабезпеченість інструментальними засобами всіх аспектів реалізації вимог замовника та ін.

**Процес проектування.** Помилки під час проектування компонентів можуть виникати при описі алгоритмів, логіки управління, структур даних, інтерфейсів, логіки моделювання потоків даних, форматів введення-виведення та ін. В основі цих помилок лежать дефекти специфікацій аналітиків і недоробки проектувальників.

*До них відносяться помилки, пов’язані:*

* з визначенням інтерфейсу користувача із середовищем;
* з описом функцій (неадекватність цілей і завдань компонентів, які виявляються під час перевірки комплексу компонентів);
* з визначенням процесу обробки інформації та взаємодії між процесами (результат некоректного визначення взаємозв’язків компонентів і процесів);
* з некоректним завданням даних та їх структур за описом окремих компонентів і ПЗ в цілому;
* з некоректним описом алгоритмів модулів;
* з визначенням умов виникнення можливих помилок у програмі;
* з порушенням прийнятих для проекту стандартів і технологій.

**Етап кодування.** На даному етапі виникають помилки, які є результатом дефектів проектування, помилок програмістів і менеджерів у процесі розробки і налагодження системи.

*Причиною помилок є:*

* безконтрольність значень вхідних параметрів, індексів масивів, параметрів циклів, вихідних результатів, ділення на 0 та ін.;
* неправильна обробка нерегулярних ситуацій під час аналізу кодів повернення від викликаних підпрограм, функцій та ін.;
* порушення стандартів кодування (погані коментарі, нераціональне виділення модулів і компонент тощо);
* використання одного імені для позначення різних об’єктів або різних імен одного об’єкта, погана мнемоніка імен неузгоджене внесення змін до програми різними розробниками та ін.

**Процес тестування.** При цьому процесі помилки допускаються програмістами і тестувальники при виконанні технології збирання та тестування, вибору тестових наборів та сценаріїв тестування та ін. Відмови в програмному забезпеченні, викликані такого роду помилками, повинні виявлятися, усуватися і не відбиватися на статистиці помилок компонентів і програмного забезпечення в цілому.

**Процес супроводження.** На процесі супроводу виявляються помилки, причиною яких є недоробки і дефекти експлуатаційної документації, недостатні показники модифікованості і легкості читання, а також некомпетентність осіб, відповідальних за супроводження або вдосконалення ПЗ. Залежно від суті внесених змін на цьому етапі можуть виникати практично будь—які помилки, аналогічні раніше перерахованих помилок на попередніх етапах.

*Всі помилки, які виникають у програмах, прийнято поділяти на наступні класи:*

* логічні і функціональні помилки;
* помилки обчислень і часу виконання;
* помилки вводу—виводу і маніпулювання даними;
* помилки інтерфейсів;
* помилки обсягу даних та ін.

*Логічні помилки* є причиною порушення логіки алгоритму, внутрішньої неузгодженості змінних і операторів, а також правил програмування. Функціональні помилки — наслідок неправильно певних функцій, порушення порядку їх застосування чи відсутності повноти їх реалізації і т.д.

*Помилки обчислень* виникають через неточності вихідних даних і реалізованих формул, похибок методів, неправильного застосування операцій обчислень або операндів. Помилки виконавчі пов’язані з незабезпеченням необхідної швидкості обробки запитів або часу відновлення програми.

*Помилки вводу—виводу і маніпулювання даними* є наслідком неякісної підготовки даних для виконання програми, збоїв при занесення їх до бази даних або при вибірці з неї.

*Помилки інтерфейсу* ставляться до помилок взаємозв’язку окремих елементів один з одним, що проявляється під час передачі даних між ними, а також при взаємодії із середовищем функціонування.

*Помилки обсягу* відносяться до даних і є наслідком того, що реалізовані методи доступу та розміри баз даних не відповідають реальним обсягам інформації системи або інтенсивності їх обробки.

Наведені основні класи помилок властиві різним типам компонентів ПЗ і проявляються вони в програмах по-різному. Так, під час роботі з БД виникають помилки подання і маніпулювання даними, логічні помилки в завданні прикладних процедур обробки даних та ін. У програмах обчислювального характеру переважають помилки обчислень, а в програмах керування та обробки — логічні і функціональні помилки. У ПЗ, яке складається з безлічі різнопланових програм, що реалізують різні функції, можуть міститись помилки різних типів. Помилки інтерфейсів і порушення обсягу характерні для будь-якого типу систем.

Аналіз типів помилок у програмах є необхідною умовою створення планів тестування та методів тестування для забезпечення правильності ПЗ.

На сучасному етапі розвитку засобів підтримки розробки ПЗ (об’єктно—орієнтовані методи та засоби проектування моделей і програм) проводиться таке проектування, при якому ПЗ захищається від найбільш типових помилок і тим самим запобігає появі програмних дефектів.

Зв’язок помилки з відмовою. Наявність помилки в програмі, як правило, призводить до відмови ПЗ при його функціонуванні. Для аналізу причинно—наслідкових зв’язків "помилка—відмова" виконуються наступні дії:

* ідентифікація недоліків у технологіях проектування та програмування;
* взаємозв’язок вад процесу проектування і допускаються людиною помилок;
* класифікація відмов, недоліків і можливих помилок, а також дефектів на кожному етапі розробки;
* зіставлення помилок людини, які допускаються на певному процесі розробки, і дефектів в об’єкті, як наслідків помилок специфікації проекту, моделей програм;
* перевірка і захист від помилок на всіх етапах ЖЦ, а також виявлення дефектів на кожному етапі розробки;
* зіставлення дефектів та відмов у ПЗ для розробки системи взаємозв’язків і методики локалізації, збору і аналізу інформації про відмови і дефекти;
* розробка підходів до процесів документування та випробування ПЗ.

Кінцева мета причинно-наслідкових зв’язків "помилка—відмова" полягає у визначенні методів і засобів тестування та виявлення помилок певних класів, а також критеріїв завершення тестування на множині наборів даних; у визначенні шляхів удосконалення організації процесу розробки, тестування та супроводження ПЗ.

Наведемо наступнукласифікацію типів відмов:

* апаратний, при якому загальносистемне ПЗ непрацездатне;
* інформаційний, викликаний помилками у вхідних даних і передачі даних по каналах зв’язку, а також при збої пристроїв введення (наслідок апаратних відмов);
* ергономічний, викликаний помилками оператора при його взаємодії з машиною;
* програмний, за наявності помилок у компонентах та ін.

Деякі помилки можуть бути наслідком недоробок при визначенні вимог, проекту, генерації вихідного коду та документації. З іншого боку, вони виявляються в процесі розробки програми або під час розробки інтерфейсів окремих елементів програми (порушення порядку параметрів, менше або більше параметрів і т.д.).

Джерела помилок. Помилки можуть бути виявлені в процесі розробки проекту, компонентів, коду і документації. Як правило, вони виявляються під час виконання або супроводу програмного забезпечення в найнесподіваніших і різних її точках.

Причиною появи помилок — нерозуміння вимог замовника; неточна специфікація вимог у документах проекту та ін. Це призводить до того, що реалізуються деякі функції системи, які будуть працювати не так, як пропонує замовник. У зв’язку з цим проводиться спільне обговорення замовником і розробником деяких деталей вимог для їх уточнення.

Команда розробників системи може також змінити синтаксис і семантику опису системи. Однак деякі помилки можуть бути не виявлені (наприклад, неправильно задані індекси або значення змінних цих операторів).

Кожна організація, яка розробляє ПЗ загальносистемного призначення, стикається з проблемами знаходження помилок. Тому доводиться класифікувати типи виявлених помилок і визначати своє ставлення до усунення цих помилок.

На основі багаторічної діяльності в галузі створення ПЗ різні фірми створили свою класифікацію помилок, засновану на виявленні причин їх появи в процесі розробки, у функціях і в області функціональної діяльності ПЗ.

***Кількісні оцінки під час тестування***

Часто техніки тестування плутають з цілями тестування. Ступінь покриття тестами — не те ж саме, що висока якість тестованої системи. Однак, ці питання пов’язані. Чим вище ступінь покриття, тим більше ймовірність виявлення прихованих дефектів. Коли ми говоримо про результати тестування, ми повинні підходити до їх оцінки, як до оцінки самої тестованої системи. Саме кількісні оцінки результатів тестування (але не самих тестів, наприклад, покриття ними можливих сценаріїв роботи системи) висвітлюються нижче.

Вимірювання є інструментом аналізу якості. Вимірювання результатів тестування стосується оцінки якості одержуваного продукту — програмної системи. Історія вимірювань демонструє прогрес досягнення прийнятної якості.

Дані вимірювання можуть базуватися на розмірі програм (наприклад, у термінах кількості рядків коду або функціональних точок) або їх структурі (наприклад, з точки зору оцінки її складності в тих чи інших архітектурних термінах). Структурні виміри можуть також включати частоту звернень одних модулів програми до інших.

***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.

***Контрольні запитання*.**

1. Надайте визначення тестування та сформулюйте його мету?
2. З якими видами діяльності тісно пов‘язане тестування?
3. В чому полягає функціональне тестування?
4. В чому полягає структурне тестування?
5. Що таке конфігурація ПЗ і які базові елементи входять до базової конфігурації?
6. Чим різниться Альфа-тестування та Бета-тестування?
7. В чому призначення системних тестів?
8. Назвіть види основних системних тестів.
9. Описати область знань «тестування програмного забезпечення», виділити її основні проблеми.
10. Надати визначення поняттю тестування.
11. Недоліки та збої (Faults vs Failures).
12. Пояснити, яким чином треба вимірювати та аналізувати результати тестування (Test—related measures).
13. Навести типи дефектів, класифікувати дефекти, оцінити статистику їх виникнення.
14. Надати визначення поняттю щільності дефектів (Fault density).
15. В чому полягає тестування продуктивності?
16. В чому сутність процесу налагодження програми?

***Література***

1. Лавріщева К.М. Програмна інженерія. Електронний підручник: – URL: <http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf>
2. Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие / Д.Сайкс, Д.Макгрегор. –К.: Диасофт, 2002. –432 с.
3. Соммервил И. Инженерия программного обеспечения / И.Соммервил.–М. : Издательский дом «Вильямс», 2002. –623 с.
4. Тамре Л. Введение в тестирование программного обеспечения / Л. Тамре. –М.: Издательский дом «Вильямс», 2003. –368 с.
5. Блэк Р. Ключевые процессы тестирования / Р.Блэк. – М.: Лори, 2006. –544 с.
6. Винниченко И.В. Автоматизация процессов тестирования / И.В.Винниченко. – СПб.: Питер, 2005. –208 с.
7. Криспин Л. Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд / Л.Криспин, Д.Грегори. –М.: Издательский дом «Вильямс», 2010. –464 с.
8. Майерс Г. Искусство тестирования программ / Г. Майерс. — Пер. с англ.— М.: Финансы и статистика, 1982. — 172 с.
9. Фолк Д. Тестирование программного обеспечения / Д.Фолк, Е.К. Нгуен, С.Канер. – К.: Диасофт, 2003. –400 с.
10. ДСТУ 2873-94 Системи оброблення інформації. Програмування. Терміни та визначення
11. ДСТУ 2853-94 Програмні засоби ЕОМ. Підготовлення і проведення випробувань
12. ДСТУ ISO/IEC 12207:2016 Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення (ISO/IEC 12207:2008, IDT)