**Лекція 15. Поняття тестування**

**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.*

**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

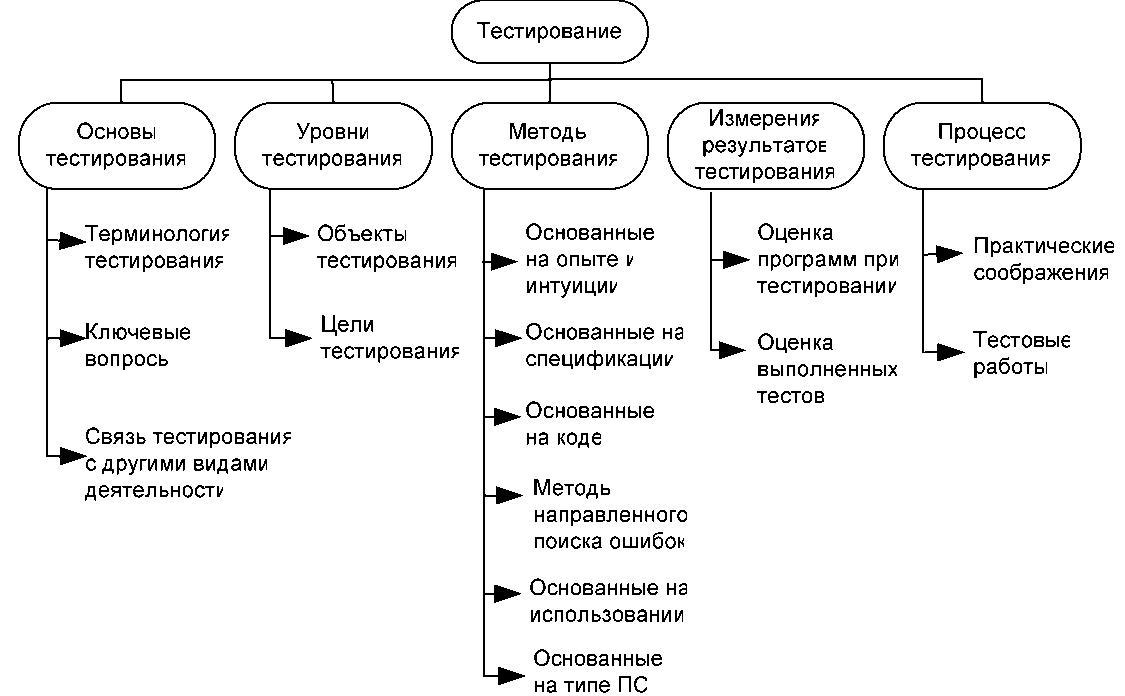
ДСТУ 2853-94 Програмні засоби ЕОМ. Підготовлення і проведення випробувань

**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.

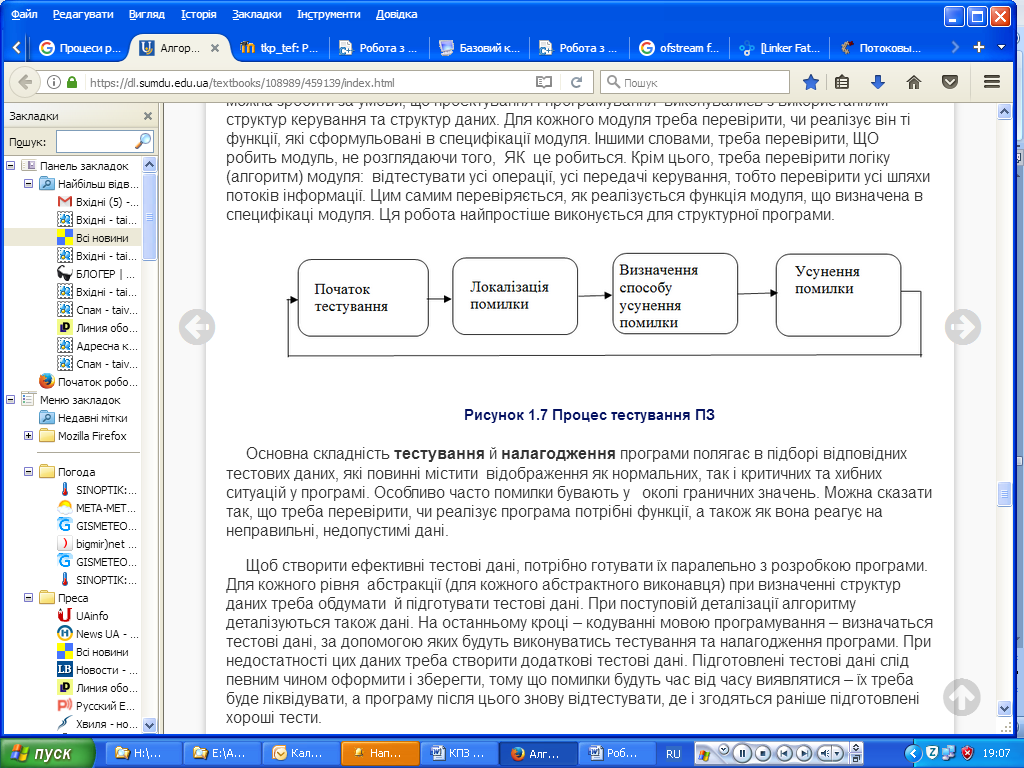


Рисунок 2. Процес тестування ПЗ

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.*

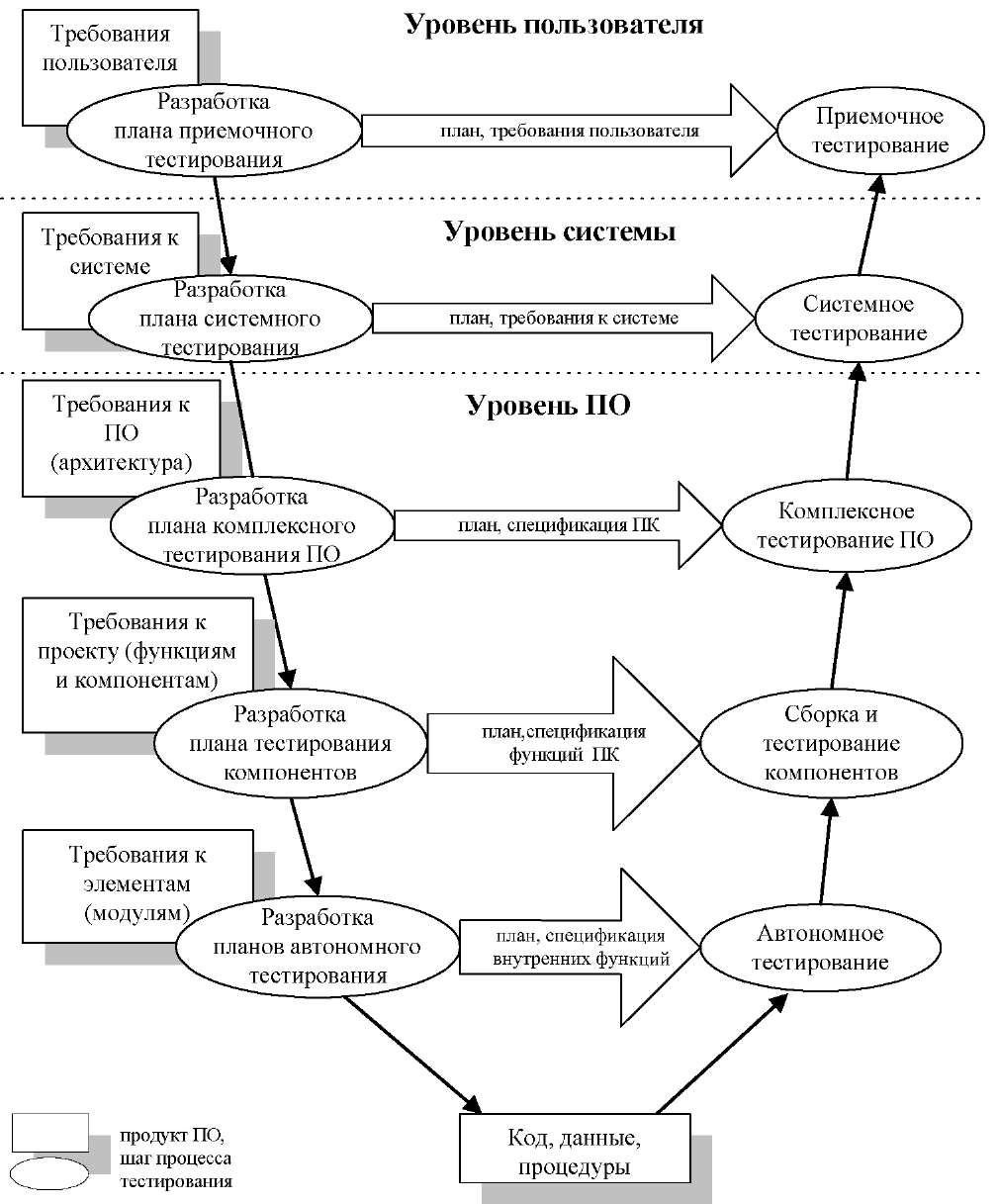
****

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)*

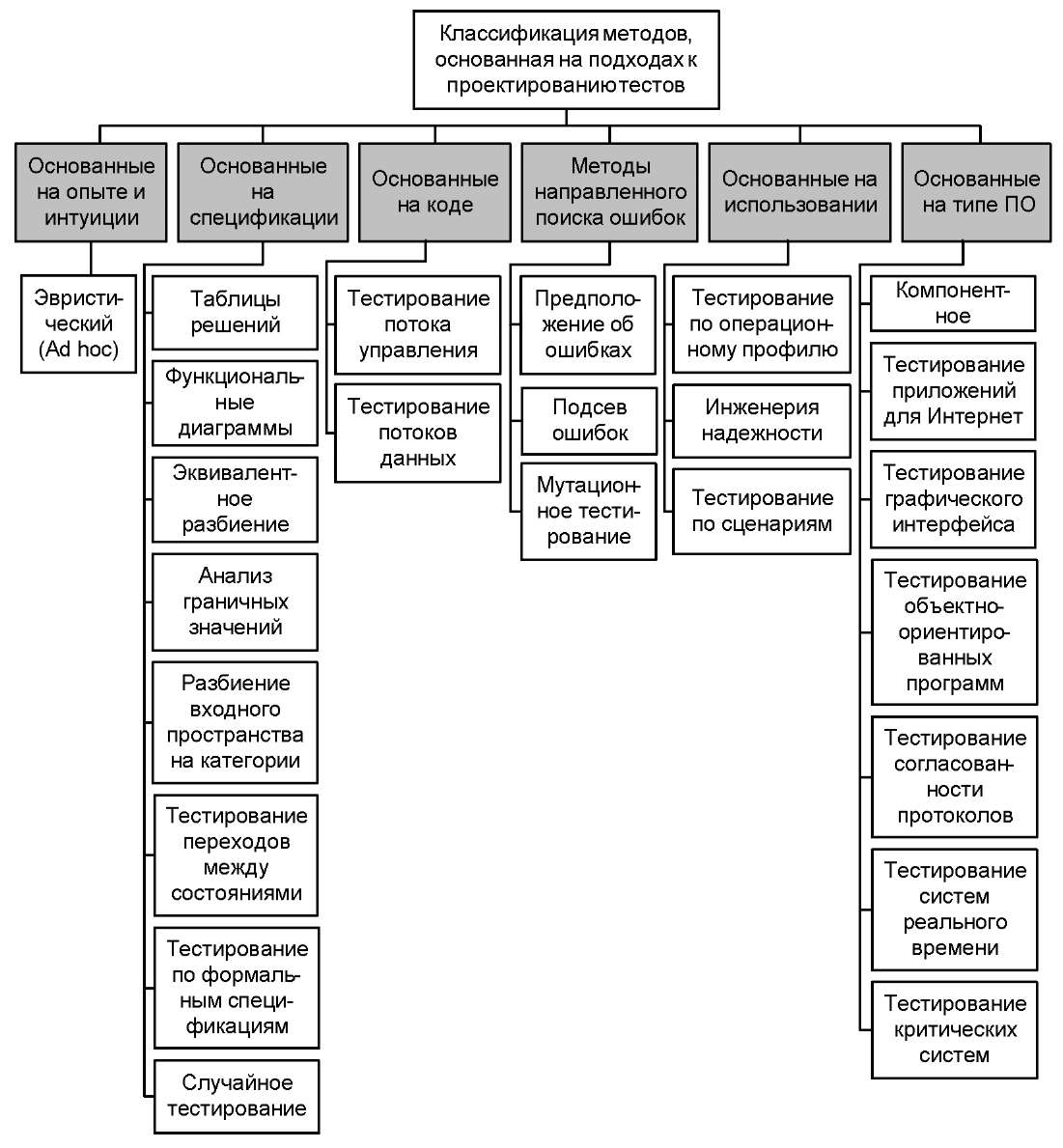


Рисунок 4 - Методи тестування

Тестування правильності. *Мета* – підтвердити, що функції, описані у специфікації вимог до ПЗ, відповідають очікуванням замовника.

Підтвердження правильності ПЗ виконується за допомогою тестів «чорного ящика», що демонструють відповідність вимогам. При виявленні відхилень від специфікації вимог створюється список недоліків. Як правило, відхилення і помилки, виявлені при підтвердженні правильності, вимагають зміни термінів розробки продукту.

Важливим елементом підтвердження правильності є перевірка конфігурації ПЗ. *Конфігурацією ПЗ* називають сукупність усіх елементів інформації, що виробляються в процесі конструювання ПЗ.

*Мінімальна конфігурація ПЗ* включає наступні базові елементи:

1. системну специфікація;
2. план програмного проекту;
3. специфікацію вимог до ПЗ; працюючий або паперовий макет;
4. попереднє керівництво користувача;
5. специфікація проектування;
6. лістинги вихідних текстів програм;
7. план і методику тестування; тестові варіанти і отримані результати;
8. керівництва по роботі та інсталяції;
9. ехе-код виконуваної програми;
10. опис бази даних;
11. керівництво користувача по налаштуванню;
12. документи супроводу; звіти про проблеми ПЗ; запити супроводу; звіти про конструкторських змінах;
13. стандарти та методики конструювання ПЗ.

Перевірка конфігурації гарантує, що всі елементи конфігурації ПЗ правильно розроблені, враховані і достатньо деталізовані для підтримки етапу супроводу в життєвому циклі ПС.

Розробник не може передбачити, як замовник буде реально використовувати ПЗ. Для виявлення помилок, які здатний знайти тільки кінцевий користувач, використовують процес, що включає альфа-і бета-тестування.

Альфа-тестування проводиться замовником в організації розробника. Розробник фіксує всі виявлені замовником помилки і проблеми використання.

Бета-тестування проводиться кінцевим користувачем в організації замовника. Розробник у цьому процесі участі не бере. Фактично, бета-тестування – це реальне застосування ПЗ у середовищі, яке не керується розробником. Замовник сам записує всі виявлені проблеми і повідомляє про них розробнику. Бета-тестування проводиться протягом фіксованого терміну (близько року). За результатами виявлених проблем розробник змінює ПЗ і тим самим готує продукт повністю на базі замовника.

Системне тестування. Даний вид тестування передбачає вихід за рамки області дії програмного проекту і проводиться не тільки програмним розробником. Класична проблема системного тестування – виявлення причини. Вона виникає, коли розробник одного системного елемента звинувачує розробника іншого елемента в причині виникнення дефекту. Для захисту від подібного звинувачення розробник програмного елемента повинен:

1. передбачити засоби обробки помилки, які тестують всі входи інформації від інших елементів системи;
2. провести тести, моделюючі невдалі дані або інші потенційні помилки інтерфейсу ПЗ;
3. записати результати тестів, щоб використовувати їх як доказ невинуватості у випадку «зазначення причини»;
4. взяти участь в плануванні та проектуванні системних тестів, щоб гарантувати адекватне тестування ПЗ.

*Системні тести повинні перевіряти, що всі системні елементи правильно об'єднані і виконують призначені функції.* Розглянемо основні типи системних тестів – тестування відновлення, тестування безпеки, стресове тестування та тестування продуктивності/ працездатності .

*Тестування відновлення*.

Комп'ютерні системи повинні відновлюватися після відмов і відновлювати обробку в межах заданого часу. У деяких випадках система повинна бути відмовостійкою, тобто відмови обробки не повинні бути причиною припинення роботи системи. В інших випадках системна відмова має бути усунута в межах заданого кванта часу, інакше замовнику наноситься серйозний економічний збиток.

Тестування відновлення використовує самі різні шляхи для того, щоб змусити ПЗ відмовити, і перевіряє повноту виконаного відновлення. При автоматичному відновленні оцінюються правильність повторної ініціалізації, механізми копіювання контрольних точок, відновлення даних, перезапуск. При ручному відновленні оцінюється, чи знаходиться середній час відновлення в допустимих межах.

*Тестування безпеки*.

Комп'ютерні системи дуже часто є мішенню незаконного проникнення. Під проникненням розуміється широкий діапазон дій: спроби хакерів проникнути в систему зі спортивного інтересу, помста колишніх адміністраторів системи тощо.

Тестування безпеки перевіряє фактичну реакцію захисних механізмів, вбудованих в систему, на проникнення.

У ході тестування безпеки випробувач грає роль взламника. Йому дозволено все:

* спроби дізнатися пароль за допомогою зовнішніх засобів;
* атака на систему за допомогою спеціальних утиліт, які аналізують захист;
* цілеспрямоване введення помилок в надії проникнути в систему в ході відновлення;
* перегляд несекретних даних в надії знайти ключ для входу в систему.

Звичайно, при необмеженому часі і ресурсах вдале тестування безпеки зламає будь-яку систему. Завдання проектувальника системи – зробити ціну проникнення більш високою, ніж ціна одержаної в результаті інформації.

*Стресове тестування*.

Стресові тести проектуються для нав'язування програмами ненормальних ситуацій. По суті, проектувальник стресового тесту запитує, як сильно можна розхитати систему, перш ніж вона відмовить?

Стресове тестування проводиться при ненормальних запитах на ресурси системи (за кількістю, частотою, розміром-об'ємом).

Приклади:

* генерується 10 переривань в секунду (при середній частоті 1,2 переривання в секунду);
* швидкість введення даних збільшується прямо пропорційно їх важливості (щоб визначити реакцію вхідних функцій);
* формуються варіанти, що вимагають максимуму пам'яті та інших ресурсів;
* генеруються варіанти, що викликають переповнення віртуальної пам'яті;
* проектуються варіанти, що викликають надмірний пошук даних на диску.

Різновид стресового тестування називається *тестуванням чутливості*. У деяких ситуаціях (зазвичай в математичних алгоритмах) дуже малий діапазон даних, що міститься в межах правильних даних системи, може викликати помилкову обробку або різке зниження продуктивності. Тестування чутливості виявляє комбінації даних, які можуть викликати нестабільність або неправильність обробки.

*Тестування продуктивності*.

У системах реального часу та вбудованих системах неприпустимо ПЗ, яке реалізує необхідні функції, але не відповідає вимогам продуктивності.

Тестування продуктивності перевіряє швидкість роботи ПЗ в комп'ютерній системі. Продуктивність тестується на всіх кроках процесу тестування. Навіть на рівні елемента при проведенні тестів «білого ящика» може оцінюватися продуктивність індивідуального модуля. Тим не менш, поки всі системні елементи не об'єднаються повністю, не може бути встановлена дійсна продуктивність системи. Іноді тестування продуктивності поєднують зі стресовим тестуванням. При цьому нерідко потрібен спеціальний апаратний і програмний інструментарій. Наприклад, часто потрібно точне вимірювання використовуваного ресурсу (процесорного циклу і т. д.). Зовнішній інструментарій регулярно відстежує інтервали виконання, реєструє події (наприклад, переривання) і машинні стану. За допомогою інструментарію випробувач може виявити стани, які призводять до деградації і можливих відмовлень системи.

**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.

***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.

***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)

***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. В чому полягає тестування продуктивності?
10. В чому сутність процесу налагодження програми?