**Практична робота № ЛР07-3**. **Процес тестування програмного забезпечення. Підготовка тестів для перевірки ПЗ (аналіз функцій).**

**Мета:** Навчиться готувати тести для перевірки програмного забезпечення.

**Завдання:**

1. Опрацювати теоретичні відомості.
2. Провести розподіл функцій власного проекту на основні та другорядні із визначенням критерію віднесення (див. Табл.1) та області можливої нестійкості продукту.
3. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва практичної роботи.
* Прізвище, група
* Назва проекту.
* Результати роботи оформлюються у вигляді таблиці:

|  |  |  |  |
| --- | --- | --- | --- |
| Функція | О/Д | За яким критерієм віднесена | Стійкість |
|  |  |  |  |

В графі "О/Д" записуємо "О" для основних, "Д" для другорядних.

В графі " Стійкість " визначаємо область можливої нестійкості функцій, або встановлюємо позначку "стійка".

По закінченню практичну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com) . Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**QA<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, **QA4101Р**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**QA <Номер групи>-ЛР<Номер лабораторної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-41 – 24.03.2025**

**ІПЗ-42 – 24.03.2025**

**Теоретичні відомості**

Підготовка тестів складається з кроків.

*Крок 1. Дослідження.*

*Крок 2. Проектування тестів.*

*Крок 3. Виконання тестів.*

*Крок 4. Визначення критерію покриття..*

***Крок 1. Дослідження***

* Формування списку функцій (ієрархії функцій).
* Розбивка функцій на основні й другорядні.
* Виявлення областей можливої нестійкості продукту (підданих відмовам функцій і даних).

Таблиця 1. Визначення основних та другорядних функцій

| **Визначення** | **Критерії віднесення** |
| --- | --- |
| *Основна функція* - будь-яка зовнішня функція, відмови в якій означають невідповідність продукту своєму призначенню. | 1. Функції, що є визначальними для використання продукту по призначенню. 2. Функції, часто використовувані звичайними користувачами в сеансі роботи. 3. Функції, які можуть використовуватися рідко, але їхні відмови приводять до значних негативних наслідків. |
| *Другорядна функція -* функція, без якої продукт може використовуватися. | Рідко використовувані функції, відмови при виконанні яких не можуть мати істотних наслідків. |

Приклади областей можливої нестійкості функцій

* функції обробки зовнішніх подій;
* функції, що інтенсивно використовують оперативна пам'ять;
* дуже складні функції;
* функції, що використовують засоби Windows і/або її параметри, що змінюють (настроювання параметрів);
* функції, що маніпулюють конфігурацією Windows;
* функції аналізу вхідних даних і обробки помилок;
* функції, що підмінюють базові функції Windows (наприклад, відновлення вилучених файлів);
* функції або групи функцій, що використовують багато паралельних процесів;
* функції, що працюють із багатьма файлами одночасно;
* функції, що працюють із файлами, розташованими в мережі.

Приклади областей можливої нестійкості при обробці даних:

* *документи:* довгі, багато одночасно відкритих;
* *запису:* довгі, багато записів, порожні, складні;
* *списки:* довгі, порожні або багатоколонкові;
* *поля:* уведення великої кількості символів; дуже більші значення числових полів;
* *об'єкти:* багато, великі, складні й ін.

***Крок 2. Проектування тестів.***

| **Визначення мети** | **Критерій проходу** | **Критерій відмови** |
| --- | --- | --- |
| *Функціональність* - здатність продукту виконувати необхідні функції. | Функція виконується відповідно до її призначення. | Хоча б одна функція не виконується відповідно до її призначення. |
| Будь-яке неправильне виконання функції не приводить до серйозних наслідків при коректному використанні. | Неправильне виконання приводить до серйозних наслідків навіть при коректному використанні. |
| *Стійкість* - здатність продукту функціонувати без відмов при підвищених навантаженнях (за часом, пам'яті). | Робота продукту при підвищених навантаженнях не приводить до руйнування Windows.  Робота продукту не приводить до втрати даних, зависанню, відмові. При тестуванні не було виявлено відмов або будь-яких дефектів при виконанні основних функцій. | Робота продукту руйнує Windows.  Робота продукту може привести до втрати даних, зависанню, відмові. При тестуванні хоч однієї з основних функцій спостерігалися відмови. |

***Крок 3. Виконання тестів***

Завдання:

* тестування всіх основних функцій;
* тестування ідентифікованих областей потенційної нестійкості;
* вибіркове тестування другорядних функцій;
* реєстрація відмов;
* фіксація будь-яких виявлених проблем (для подальшого вивчення).

***Крок 4. Визначення критерію покриття***

* протестовані всі основні функцій;
* протестовані обрані другорядні функції;
* протестовані обрані області можливої нестійкості (п'ять - десять функцій, які можуть привести до нестійкої роботи продукту, протестувати їх на тестових даних).

Розподіл часу

* 80% часу - на основні функції,
* 10% - на другорядні й
* 10% - на області потенційної нестійкості.

**Еквівалентна розбивка**

Критерії :

* тести включають значення тих самих вхідних даних;
* при запуску тестів виконуються ті самі операції;
* від виконання тестів очікуються однакові результати;
* або жоден з тестів не викликає виконання блоку обробки помилок, або всі тести викликають виконання цього блоку (у припущенні, що програма взагалі містить блоки обробки помилок).

Приклад опису класів еквівалентності

|  |  |  |
| --- | --- | --- |
| **Вхідні події** | **Припустимі класи еквівалентності** | **Неприпустимі класи еквівалентності** |
| Уведення числа | Числа від 0 до 9000 | Числа менше 0 і числа більше 9000  Пробіл  Нечислові символи  Вираз, що обчислюється, результат обчислення якого перебуває поза припустимим інтервалом |
| Уведення першої букви прізвища | Перший символ повинен бути заголовною буквою | Перший символ мала літера  Перший символ - не буква |

**Розбивка вхідного простору на категорії** виконується у послідовності

1. *Декомпозиція функції на функціональні елементи, які можуть тестуватися незалежно.*
2. *Ідентифікація параметрів і умов середовища, що впливають на поводження функції.*
3. *Виділення категорій інформації, що характеризує кожний параметр і умова середовища.*
4. *Розбивка кожної категорії на чіткі альтернативи, які включають різні види значень, можливі для категорії.*
5. *Формування формальної специфікації тесту для кожного функціонального елемента.*

Приклад формування формальної специфікації тесту для кожного функціонального елемента

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Категорія** | **Альтернативи** |
| Поле вибору | Зміст | Текст  Піктограма  Значок з текстом |
|  | Стан | Поле не зазначене/обране  Поле не зазначене/не обране  Поле зазначене/обране  Поле зазначене/не обране |
|  | Тип поля | Однозначний вибір  Багатозначний вибір  Розширений вибір |
| Меню | Спосіб вибору | Вибір курсором  Виділена буква/(комбінація букв)  Функціональна клавіша  Вибір за допомогою клавіатури |

Що надає такий підхід:

* дозволяє охопити відразу обоє основних аспекту тестування - перевірку повноти реалізації функцій і виявлення різних класів дефектів у найбільш уразливих частинах програми;
* функціональний аналіз є невід'ємною частиною методу й виконується паралельно зі специфікацією функціональних вимог (або відразу після), забезпечуючи своєчасне усунення дефектів специфікацій;
* хоча специфікація тесту повинна охоплювати всі можливі категорії інформації й варіанти сполучення альтернатив, - застосовуючи механізм обмежень і керуючись практичними міркуваннями, можна управляти обсягом тестування;
* процес перебору альтернатив і «відсівання» неможливих або небажаних їхніх сполучень може бути автоматизований, що позбавить тестувальника від рутинної роботи.

**Питання для самоконтролю:**

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?
17. Назвіть рівні тестування (Test Levels).
18. Над чим проводяться тести?
19. Навести типові цілі тестування (Objectives for testing).
20. Приймальне тестування (Acceptance / qualification testing)
21. Пояснити, навіщо використовують альфа і бета-тестування (Alpha and beta testing), хто його проводить, навести приклади.
22. Функціональні тести / тести відповідності (Conformance testing / Functional testing / Correctness testing).
23. Проаналізувати процес регресійного тестування (Regression testing).
24. Проаналізувати процес навантажувального тестування (Stress testing).
25. Проаналізувати процес порівняльного тестування (Back-to-back testing), обґрунтувати доцільність його проведення.
26. Тестування спроможності системи на відновлення (Recovery testing).
27. Проаналізувати процес конфігураційного тестування (Configuration testing), обґрунтувати доцільність і умови його проведення.
28. Описати основні етапи тестування зручності і простоти використання (Usability testing), обґрунтувати доцільність проведення такого тестування.
29. У чому полягає відмінність інтеграційного і модульного видів тестування від системного?
30. Що саме аналізує системне тестування?
31. Який метод тестування застосовується під час тестування системи в цілому?
32. На які категорії можна поділити тести системного тестування?
33. Які частини програмного комплексу тестуються в кожному з названих випадків?
34. Що таке регресійне тестування?
35. Опишіть порядок дій під час виконання регресійного тестування.
36. З якою метою можуть комбінуватися рівні тестування? Чого можна досягти за рахунок комбінування?