ЛАБОРАТОРНА РОБОТА №1

ПОНЯТТЯ ТЕСТУВАННЯ. СТАДІЇ, ВИДИ ТЕСТУВАННЯ. ПЛАН ТЕСТУВАННЯ

Мета роботи: Використання технік тестування (згідно стандарту BS 7925-2) при тестуванні програмного забезпечення(аналіз граничних значень, розподілу на еквівалентні класи вхідних даних). Тестовий сценарій (test case)(типові форми, зміст).

Порядок виконання роботи:

1. Ознайомитися з поняттям тестування ПЗ, розглянути стадії та типи тестування - Додаток №1.
2. Ознайомитись з двома прикладами планів тестування (RUP та стандарт IEEE 829) та з наведеними порадами щодо складання плану - Додаток №2.
3. Скласти план тестування програми згідно зі своїм варіантом, враховуючи наведені поради та прийнятий порядок оформлення розділів (за приклад взяти план за RUP або IEEE 829).
4. Згідно зі своїм вариантом, провести тестування програмного забезпечення по наступній схемі:

4.1 Зробити короткий опис продукта згідно моделі якості ПЗ (FURPS)-1-2 сторінки.

4.2 Ознайомитися з поняттям тестового сценарію, вимогами до його написання (див. Додаток №3).

4.3 Створити простий базовий тест та оформити його тестові сценарії згідно з шаблоном, наведеним в Додатку №4.

4.4 Провести інвентаризацію типів вхідних даних програми та комбінуючи техніку аналіза граничних значень і виділення еквівалентних класів вхідних даних з’ясувати, якій набір тестових даних використати під час проведення тестів.

Додаток №1

ТЕСТУВАННЯ. ВИДИ ТЕСТУВАННЯ.

Тестування ПЗ – процес виявлення дефектів в програмному забезпеченні.

Будь-який метод тестування не дозволяє однозначно та повно встановити абсолютну коректність функціонування програми.

Метою будь-якого процесу тестування є забезпечення якості програмного забезпечення, враховуючи всі або найбільш критичні фактори.

ISO 9126: Якість (програмних засобів) можна визначити як сукупну характеристику програмного забезпечення, що досліджується, враховуючи наступні складові:

- надійність
- супроводжуваність
- практичність
- ефективність
- мобільність
- функціональність

Більш повний перелік атрибутів та критеріїв можна знайти в самому стандарті ISO 9126 Міжнародної організації стандартизації.

Склад та зміст документації, що супроводжує процес тестування визначається стандартом IEEE 829-1998 Standard for Software Test Documentation.

Стадії тестування ПЗ.

Модульне тестування (юніт-тестування) – найнижчий рівень тестування. Тестується мінімально можливий компонент програмного комплексу (юніт), наприклад, окремий клас чи функція, при цьому не береться до уваги взаємодія цього “юніту” з іншими компонентами системи. Юніт-тестування виконує зазвичай програміст.

Для більшості популярних мов програмування високого рівня існують інструменти та бібліотеки для юніт-тестування.

- JUnit (JUnit.org) — для Java
- NUnit— для мов платформи . NET: C#, Visual Basic .NET и др.
- для C++ : CPPUnit.
- DUnit— для Delphi

Інтеграційне тестування — перевіряє, чи є проблеми в інтерфейсах і взаємодії між індивідуальними компонентами - наприклад, чи дані, що передаються, є коректними.

Інтеграційне тестування в якості вхідних параметрів використовує модулі, над якими проводилося юніт-тестування, групує їх та тести виконуються вже на рівні таких об’єднаних модулів, використовуючи технологію тестування “чорного ящика”.

Системне тестування — тестується повна інтегрована система на її відповідність вихідним вимогам:

а) Альфа-тестування — імітація реальної роботи із системою штатними програмістами, або реальна робота із системою потенційними користувачами/замовником на стороні розроблювача. Часто альфа-тестування застосовується для закінченого продукту як внутрішнє приймальне тестування. Виявлені помилки можуть бути передані тестувальникам для додаткового дослідження в оточенні, подібному тому, у якому буде використовуватися ПЗ.

б) Бета-тестування — у деяких випадках розповсюджують версію з обмеженнями (чи по функціональності чи по часу роботи) для деякої групи користувачів, щоб переконатися, що продукт містить досить мало помилок. Іноді бета-тестування виконується для того, щоб отримати відгуки про продукт від його майбутніх користувачів.

Статичне та динамічне тестування

Динамічне тестування – об'єднує всі типи тестування, які розраховані на те що код виконується.

Статичне тестування – програма не виконується, замість будь-якої перевірки самої функціональності вичитують програмний код, перевіряється логіка роботи, чи відповідає написаний код стандартам та т.ін. До статичного тестування відноситься і тестування вимог до ПЗ.

Тестування "чорного ящика" або "білого ящика"

При тестуванні за принципом "білого ящика" (white-box testing) тестувальник має доступ до коду та може писати код пов'язаний з бібліотеками ПЗ, що тестується. Цей метод є типовим для юніт-тестування, він забезпечує що компоненти системи функціонують та до певної міри стабільні.

При тестуванні за принципом "чорного ящика" (black-box testing) тестувальник має доступ до ПЗ тільки через інтерфейс, внутрішня структура та логіка функціонування окремих юнітів є невідомими.

Тестування "сірого ящика" – розумне поєднання перших двох типів (доступна або "видимої" є лише частина програмного коду).

Типи тестування в залежності від типів вимог до ПЗ:

- Functional (Feature) — функціональне
- UI / GUI (+Usability) — інтерфейсу користувача (графічного)
- Interface — інтерфейсу з іншими системами
- Security — вимог безпеки
- Performance — вимоги до продуктивності
- Configuration — тестування конфігурацій
- Reliability — тестування надійності
- Documentation — документації, вбудованої системи довідки

Додаток №2

ПЛАН ТЕСТУВАННЯ

Тест план (**Test Plan**) – документ, що описує повний обсяг робіт з тестування, починаючи з опису об’єкта, стратегії, розкладу, критеріїв початку та закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також можливих ризиків з варіантами їх розв’язку.

Кожна методологія чи процес мають свої формати оформлення планів тестування. Серед найбільш поширених є шаблони від **RUP (Rational Unified Process)** та з **стандарту IEEE 829** (див. приклади в окремих файлах)

Обидва приклади складені за майже однаковими принципами.

Тест план має відповідати на наступні питання:

1. Що треба тестувати (об’єкт тестування: система, обладнання..._

2. Що буде тестуватися (перелік функцій і компонентів системи)

3. Що не буде тестуватися (подібній п.2 перелік функцій, але які не будуть тестуватися).

4. Як буде відбуватися тестування (стратегія тестування – види тестування по відношенню до об’єкта тестування)

5. Розклад тестування (послідовність проведення робіт: підготовка, тестування, аналіз результатів, у відповідності до запланованих фаз розробки проекта)

6. Критерії початку та закінчення тестування

7. Ризики з варіантами їх розв’язку

Додаток №3

ТЕСТОВИЙ СЦЕНАРІЙ (TEST CASE).

IEEE Standard 610 (1990) визначає тестовий сценарій (тест-кейс) як: набір вхідних вимог, умов виконання та очікуємих результатів, що розроблений з певними цілями, такими як виконання певного програмного коду або для перевірки на відповідність певній програмній вимозі (product requirement).

Тестовий сценарій (тест-кейс) - це послідовність дій, виконання яких дає можливість зробити висновок чи відповідає функція, що тестується вимогам до неї. Будь який тест-кейс має містить перелік дій, що треба виконати з відповідними результатами по кожній дії (пари «дія» – «результат»).

Метою написання тест-кейсів є визначення конкретної послідовності дій, яка допоможе визначити найбільшу кількість багів, які необхідно виправити та згодом перевірити, чи виправлені ці баги.

Тест-кейс має включати (обов'язкові поля):

- **Унікальний ідентифікатор (№___** : <Назва тестового сценарію>)
- **Мета** (Purpose) – з метою перевірки якої функції, властивості системи пишеться цей сценарій
- **Передумови** його виконання (Preconditions) – опис дії, які приводять систему в стан, що є придатним для проведення тестування
- **Послідовність дій**, які треба виконати для перевірки мети тест-кейса (Actions – Expected result)
- **Результат –** результат виконання тест-кейса (Pass – тест-кейс пройдений та помилок не виявлено, Fail – при виконанні тест-кейсу була виявлена помилка, або кілька помилок.

Тест-кейс також може включати (або ці поля можуть не зазначатися):

- **Тип (Type)** – у відповідності до типу вимоги, що тестується (може бути зрозуміло з назви)
- **Пріоритет** – важливість виконання даного тест-кейса для висновків чи відповідає програмний продукт вимогам до нього - суб'єктивна міра (Високий, Середній, Низький)
- **Час на виконання** – скільки часу (як ви плануєте) буде витрачено на виконання цього тест-кейсу
- **Постумови (Postconditions)** – умови, що приводять систему до початкового стану (досить рідко після завершення тесту необхідно привести систему до стану перед тестом, часто це є неможливим).

Часто тестові сценарії поєднуються в набори тестів (test suites), коли результат попереднього тест-кейса береться за початок для наступного. Наприклад, набор тестів для тестування програми, що оперує з базою даних, може включати 3 тест-кейси: один, який записує дані до бази, другий, що маніпулює з цими даними, та третій, який зчитує результуючі дані з бази та очищає їх. Ці три тест-кейси будуть виконуватись один за одним.

Вимоги, яким має відповідати тест-кейс:

Точність (Accurate) – перевіряти те, що саме має бути перевірене

Економічність, лаконічність (Economical) – не має містити зайвих дій

Повторюваність (Repeatable) – може виконуватися багато разів

Відслідковуємість (Traceable) – має зберігатися зв'язок с програмною вимогою, що перевіряється

Відповідність (Appropriate) – відповідність як до середовища, так і до тестерів, що будуть його виконувати

Об'єктивність, незалежність від автора, (Self standing) – тест-кейс має бути зрозумілим всім, хто задіяні в процесі тестування/розробки ПЗ, а не лише авторові.

Додаток №4

Шаблон тестового сценарію

<table>
<tr><td colspan="2">№___ : <Назва тестового сценарію></td></tr>
<tr><td colspan="2">Мета: <…></td></tr>
<tr><td colspan="2">Тип: <Functional, GUI, Documentation, Performance…></td></tr>
<tr><td>Приорітет: <High, Medium, Low></td><td>Час на виконання: < ___ хв.></td></tr>
<tr><td>Дата: <…></td><td>Власник: <…></td></tr>
<tr><td colspan="2">Передумови: 1. _____
2. _____</td></tr>
<tr><td>Дії</td><td>Очикуємий результат</td></tr>
<tr><td>1. <…></td><td>1. <…></td></tr>
<tr><td>2. <…></td><td>2. <…></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Післяумови: _____</td></tr>
</table>