Тестування програмного забезпечення

Викладач: к.т.н. Шитікова Олена Вікторівна

ВВЕДЕННЯ

Тестування програмного забезпечення – процес дослідження, випробування програмного продукту, який має дві різні цілі:

– продемонструвати розробникам та замовникам, що програма відповідає вимогам;

– виявити ситуації, в яких поведінка програми є невірною, небажаною або не відповідає специфікації.

ВВЕДЕННЯ

Мета дисципліни: створити підґрунтя для ознайомлення з концепціями тестування програмного забезпечення: тестування вебпроєктів, зручності використання, кросбраузерне тестування, функціональне тестування, технічне тестування тощо.

Завдання дисципліни: ознайомитись з методами та засобами тестування програмного забезпечення, підходами проведення різноманітних видів тестування.

ВВЕДЕННЯ

У результаті вивчення навчальної дисципліни студент повинен ***знати:***

- життєвий цикл звітів, розстановку пріоритетів та їх обробку;
- загальне уявлення про інструментарій тестувальника (система відслідковування помилок, система управління проєктами, інструменти автоматизації та інше);
- основи функціонального тестування;
- дослідне тестування;

ВВЕДЕННЯ

та вміти:

- чітко визначати мету тестування;
- створювати баг-репорти, проводити спостереження та аналіз результатів тестування;
- розробляти та працювати з необхідною тестовою документацією;
- аналізувати, планувати та проводити різні види тестування;
- тестувати вебдодатки;
- перевіряти повноту вимог, їх несуперечливість, дублювання;
- перевіряти різні аспекти якості;
- тощо.

Структура курсу

Тема 1. «Вступ. Тестування програмного забезпечення»

Лабораторна робота № 1 – Баг-репорт

Тема 2. «Види тестування»

Лабораторна робота № 2 - Види тестування

Тема 3. «Вебтестування та чек-листи»

Тема 4. «Кросбраузерне тестування»

Лабораторна робота № 3 - Вебтестування

Тема 5. «Тестування зручності використання»

Лабораторна робота № 4 - Тестування зручності використання

Тема 6. «Тестування вебпроєктів»

Структура курсу

Тема 7. «Технічне тестування»

Лабораторна робота № 5 - Технічне тестування

Тема 8. «Функціональне тестування»

Лабораторна робота № 6 – Функціональне тестування

Тема 9. «Тест-дизайн, тест-кейси, техніки тест-дизайну»

Лабораторна робота № 7 – Тест-дизайн, тест-кейси

Тема 10. «Тест-плани»

Тема 11. «Звіти про тестування»

Тема 12. «Мобільне тестування»

Тема 13. «Тестування ігор»

Підсумковий контроль – залік (диференційований)

Лекція 1
Вступ: Тестування програмного забезпечення

Історія розвитку тестування програмного забезпечення

Перші програмні системи розроблялися в рамках програм наукових досліджень або програм для потреб міністерств оборони.

1960-ті – «вичерпне» тестуванню, яке мало проводитися з використанням усіх шляхів у коді або всіх можливих вхідних даних.

На початку 1970-х тестування ПЗ позначалося як «процес, спрямований на демонстрацію коректності продукту» або як «діяльність по підтвердженню правильності роботи ПЗ».

Історія розвитку тестування програмного забезпечення

У другій половині 1970-х тестування уявлялося, як виконання програми з наміром знайти помилки, а не довести, що вона працює. Успішний тест – це тест, який виявляє раніше невідомі проблеми.

У 1980-х до тестування додалося таке поняття, як попередження дефектів. Проєктування тестів – найбільш ефективний з відомих методів попередження помилок.

У середині 1980-х з'явилися перші інструменти для автоматизованого тестування.

Історія розвитку тестування програмного забезпечення

На початку 1990-х в поняття «тестування» стали включати планування, проектування, створення, підтримку і виконання тестів і тестових оточень, і це означало перехід від тестування до забезпечення якості, що охоплює весь цикл розробки ПЗ.

У середині 1990-х з розвитком інтернету і розробкою великої кількості вебдодатків особливу популярність здобуває «гнучке тестування».

У 2000-х з'явилося ще більш широке визначення тестування, коли до нього було додано поняття «оптимізація бізнес-технологій» (business technology optimization, BTO).

Особливості та вимоги до професії тестувальника (Tester)

Обов'язки тестувальника:

- контроль якості розроблюваних продуктів;
- планування тестування та всіх необхідних видів робіт;
- виявлення та аналіз помилок і проблем, що виникають у користувачів при роботі з програмними продуктами;
- розробка сценаріїв тестування;
- документування знайдених дефектів;
- складання технічної документації (найчастіше англійською моваю на рівні Upper Intermediate);
- розробка автотестів та їх регулярний прогін (при необхідності і наявності відповідних знань).

Особливості та вимоги до професії тестувальника (Tester)

Вимоги до тестувальника:

- вища освіта;
- аналітичні здібності;
- знання англійської мови на рівні, достатньому для читання і написання технічних текстів.

Додаткові вимоги:

- досвід в організації та проведенні різних видів тестування;
- вміння тестувати вебдодатки;
- знання Windows, macOS;
- знання мобільних платформ (iOS, Android);
- базові знання HTML, CSS, SQL;
- знання основ мов програмування.

Основні поняття і терміни

Баг та атрибути бага

Існує кілька визначень терміна «Баг» (bug):

- дефект, помилка в програмі або в системі, яка видає несподіваний або невірний результат;
- відхилення фактичного результату (actual result) від очікуваного результату (expected result).

Атрибути багів

- *номер бага в системі* (bug number);
- *серйозність* (severity) – це технічна категорія, яка визначає критичність багу з точки зору тестувальника: особливість, помилка в тексті, дрібна проблема, значна проблема, падіння продукту, проблема блокуючого характеру:
  - критичний (critical):
    - ❖ критичний системний збій (crash);
    - ❖ втрата даних (data loss);
    - ❖ проблема з безпекою (security issue);

Атрибути багів

- значний (major):
  - ❖ сайт «зависає» (site hangs);
  - ❖ баг блокує кодування, тестування або використання вебсайту (blocker);
- помірний (minor):
  - ❖ функціональні проблеми (functional bugs);
- косметичний (cosmetic):
  - ❖ косметична проблема (cosmetic problem)
  - ❖ normal;
  - ❖ trivial;

Атрибути багів

– *пріоритет* (priority) – пріоритет, з яким проблема повинна бути виправлена – також є показником важливості бага для бізнесу компанії:

- immediate;
- urgent;
- high;
- normal;
- low;

Атрибути багів

– *короткий опис* (summary) – це максимально інформативний і стислий опис проблеми;

– *опис* (description) – корисна інформація про баг: опис, коментарі, нюанси;

– *кроки відтворення* (steps to reproduce) – конкретні кроки для відтворення проблеми;

– *прикріплення* (attachment) – будь-яка інформація, яка допоможе відтворити ситуацію (скріншоти, відео, лог-файл);

– *додаткова інформація* (операційна система, браузер + версія, мобільний пристрій).

Тестування та мета тестування

Тестування програмного забезпечення (Software Testing) – це перевірка відповідності між реальною та очікуваною поведінкою програми, яка здійснюється на кінцевому наборі тестів, обраному певним чином.

У більш широкому сенсі, ***тестування*** – це одна з технік контролю якості, що включає в себе активності з:

- планування робіт (Test Management),
- проєктування тестів (Test Design),
- виконання тестування (Test Execution)
- аналіз отриманих результатів (Test Analysis).

Тестування та мета тестування

Верифікація (Verification) – це процес оцінки системи або її компонентів, метою якого є визначення того, чи задовольняють результати поточного етапу розробки умовам, сформованим на початку цього етапу [IEEE].

Валідація (Validation) – це визначення відповідності ПЗ, що розробляється, очікуванням і потребам користувача, вимогам до системи [BS7925-1].

Спільне та відмінності процесів тестування, верифікації, валідації

Верифікація (Verification)	Валідація (Validation)
це статична практика перевірки документів, дизайну, архітектури, коду, тощо. 1)включає перевірку планів, специфікацій вимог, специфікацій дизайну, коду, тест-кейсів, чек-листів тощо 2)завжди проходить без запуску коду. 3)використовує методи – reviews, walkthroughs, inspections тощо 4)відповідає на питання “Чи робимо ми продукт правильно?” 5)допоможе визначити, чи є програмне забезпечення **високої якості**, але воно не гарантує, що система **корисна**. Перевірка пов’язана з тим, що система добре спроєктована і безпомилкова. 6)відбувається до **Валідації**	це процес оцінки кінцевого продукту. Перевіряє, чи відповідає програмне забезпечення очікуванням і вимогам клієнта. Це динамічний механізм перевірки та тестування фактичного продукту 1) завжди включає в себе запуск коду програми. 2) використовує методи, такі як тестування Black Box, тестування White Box і нефункціональне тестування. 3) відповідає на питання “Чи робимо ми правильний продукт?” 4) перевіряє, чи відповідає програмне забезпечення вимогам і очікуванням клієнта. 5) може знайти помилки, які процес **Верифікації** не може зловити відбувається після **Верифікації**

Спільне та відмінності процесів тестування, верифікації, валідації

На практиці, відмінності ***верифікації*** та ***валідації*** мають велике значення:

– замовника цікавить більше валідація (задоволення власних вимог);

– виконавця хвилює не тільки дотримання всіх норм якості (верифікація) при реалізації продукту, а й відповідність всіх особливостей продукту бажанням замовника.

Приклад: Перевірка вебформи.

Верифікація: перевіряємо наявність полів. Всі поля повинні відповідати специфікації. Їх наявність визначено макетами. Необхідна інформація вноситься в ТЗ або до мокапів. Також перевіряється, що поля всі робочі, в них можливо ввести різні дані згідно найменувань.

Валідація: перевіряємо інформацію, що вводиться в поля та її відповідність специфікації.

Тестування та мета тестування

Інцидент (Test Incident) – будь-яка подія, спостереження, знайдене в рамках тестування, що вимагає дослідження.

Звіт щодо інциденту (Incident Report) – документ, що описує подію, яка відбулася під час тестування, і яку необхідно досліджувати.

Звіт про запит на зміну (Improvement Report) – документ, що описує пропозицію про вдосконалення продукту. Включає в себе детальний опис пропозиції та обґрунтування внесення змін до програмного забезпечення.

Тестування та мета тестування

План тестування (Test Plan) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладів, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх дозволу.

Тестовий випадок (Test Case) – це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини.

Баг/Дефект-репорт (Bug Report) – це документ, що описує ситуацію або послідовність дій, яка призвела до некоректної роботи об'єкта тестування, із зазначенням причин і очікуваного результату.

Мета тестування – це знаходження багів до того, як їх знайдуть користувачі.

Системи відслідковування помилок та життєвий цикл дефекту

Система відслідковування помилок (Bug Tracking System) – це прикладна програма, розроблена, щоб допомогти розробникам програмного забезпечення (програмістам, тестувальникам та ін.) враховувати й контролювати помилки і неполадки, знайдені в програмах, побажання користувачів, а також стежити за процесом усунення цих помилок і виконання або невиконання побажань.

Системи відслідковування помилок та життєвий цикл дефекту

Баг-трекер (bug tracker) – система обліку та відстеження помилок, яка дозволяє:

- створювати;
- зберігати;
- переглядати;
- модифікувати інформацію про баги.

Сучасні системи відслідковування помилок

1. Mantis Bug Tracking System – вільно розповсюджувана система відслідковування помилок у програмних продуктах (bug tracker). Забезпечує взаємодію розробників з користувачами (тестувальниками). Дозволяє користувачам заводити повідомлення про помилки й відстежувати подальший процес роботи над ними з боку розробників. Система має гнучкі можливості конфігурації, що дозволяє настроювати її не тільки для роботи над програмними продуктами, але і в якості системи обліку заявок для технічної підтримки.

Сучасні системи відслідковування помилок

2. Redmine – відкритий серверний вебдодаток для управління проєктами і завданнями (у тому числі, для відстеження помилок). Redmine написаний мовою Ruby і являє собою додаток на основі широко відомого вебфреймворку Ruby on Rails.

Redmine надає наступні можливості:

- ведення декількох проєктів;
- гнучка система доступу, заснована на ролях;
- система відслідковування помилок;
- діаграми Ганта і календар;
- ведення новин проєкту, документів і управління файлами;

Сучасні системи відслідковування помилок

• оповіщення про зміни за допомогою RSS-потоків і електронної пошти;

• вікі (wiki) для кожного проєкту;

• форуми для кожного проєкту;

• облік тимчасових витрат;

• настроюються довільні поля для інцидентів, тимчасових витрат, проєктів і користувачів;

• створення записів про помилки за отриманими листами;

• можливість самостійної реєстрації нових користувачів;

• багатомовний інтерфейс;

• підтримка СУБД MySQL, PostgreSQL, SQLite, Oracle.

Сучасні системи відслідковування помилок

3. Atlassian JIRA – комерційна система відслідковування помилок, призначена для організації спілкування з користувачами, хоча в деяких випадках систему можна використовувати для управління проєктами. Розроблено компанією Atlassian Software Systems. Назву системи (JIRA) отримано шляхом усічення слова «Gojira», японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla. JIRA створювалася для заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проєктами. Для кожного з проєктів створює і веде схеми безпеки і схеми оповіщення.

Життєвий цикл дефекту