Тестування, верифікація та валідація ПЗ

Концидайло Андрій Михайлович

Quality Assurance Engineer
asp_kam1@student.ztu.edu.ua
@AndyFox96

Туренко Єлизавета Єгорівна

Quality Assurance Engineer
kipz_tee@ztu.edu.ua
@patric_kosichka

Оцінювання студента.

Відвідування.

Лекція 1. Знайомство з тестуванням.

Що таке забезпечення якості (QA) у тестуванні програмного забезпечення?

Що таке якість?

Якість надзвичайно важко визначити, і це просто заявлено: «Придатний для використання або призначення». Йдеться про задоволення потреб і очікувань клієнтів щодо функціональності, дизайну, надійності, довговічності та ціни продукту.

Що таке гарантія?

Гарантія - це не що інше, як позитивна декларація про продукт або послугу, яка додає впевненості. Це впевненість у тому, що продукт чи послуга працюватимуть добре. Це забезпечує гарантію того, що продукт працюватиме без будь-яких проблем відповідно до очікувань або вимог.

Як забезпечити якість: повний процес

Методологія забезпечення якості має певний цикл, який називається циклом PDCA або циклом Демінга. Фази цього циклу:

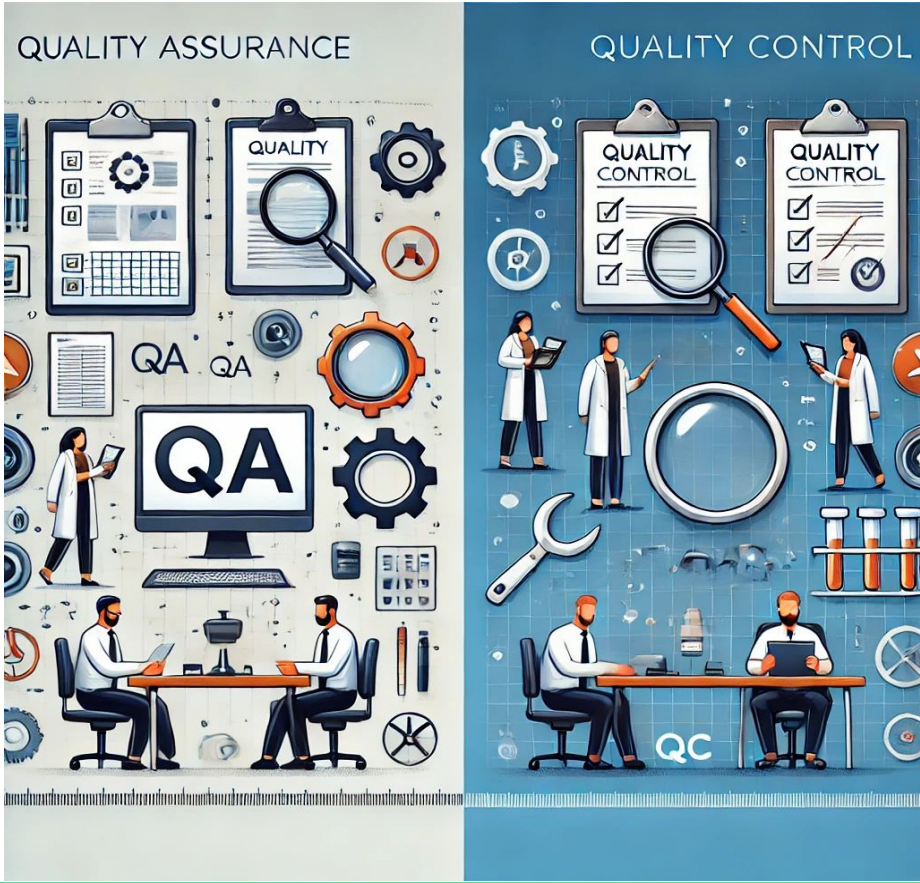
План – організація повинна спланувати та встановити цілі, пов’язані з процесом, і визначити процеси, необхідні для отримання високоякісного кінцевого продукту.
Робити – розробка та тестування процесів, а також «робити» зміни в процесах
Перевірка – моніторинг процесів, модифікація процесів і перевірка їх відповідності заздалегідь визначеним цілям
Дійте – Тестер із забезпечення якості повинен виконувати дії, необхідні для покращення процесів

Що таке контроль якості?

Контроль якості, в народі скорочений як QC. Це процес розробки програмного забезпечення, який використовується для забезпечення якості продукту чи послуги. Він не стосується процесів, які використовуються для створення продукту; скоріше він перевіряє якість «кінцевих продуктів» і кінцевий результат.

Різниця між контролем якості та забезпеченням якості?

Іноді QC плутають із QA. Контроль якості полягає в дослідженні товару чи послуги та перевірці результату. Забезпечення якості в розробці програмного забезпечення полягає в дослідженні процесів і внесенні змін до процесів, які призвели до кінцевого продукту.

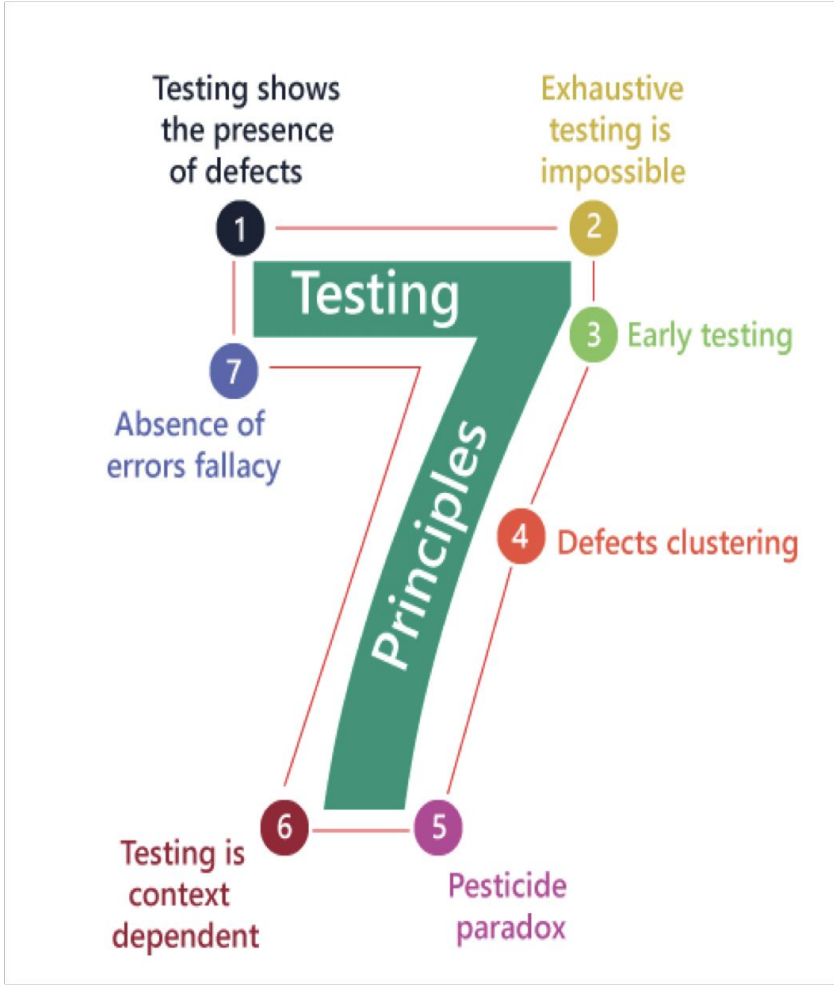


Тип тестування

Деякі тестування включають виконання компонента або системи, що тестується; таке тестування називається ***динамічним.*** *Інше тестування не передбачає виконання компонента або системи, що тестується; таке тестування називається* ***статичним тестуванням.***

Принципи тестування

1. Тестування показує наявність дефектів, а не їх відсутність
2. Вичерпне тестування неможливо
3. Раннє тестування економить час і гроші
4. Баги групуються разом
5. Остерігайтеся парадоксу пестицидів
6. Тестування залежить від контексту
7. Відсутність помилок є помилкою

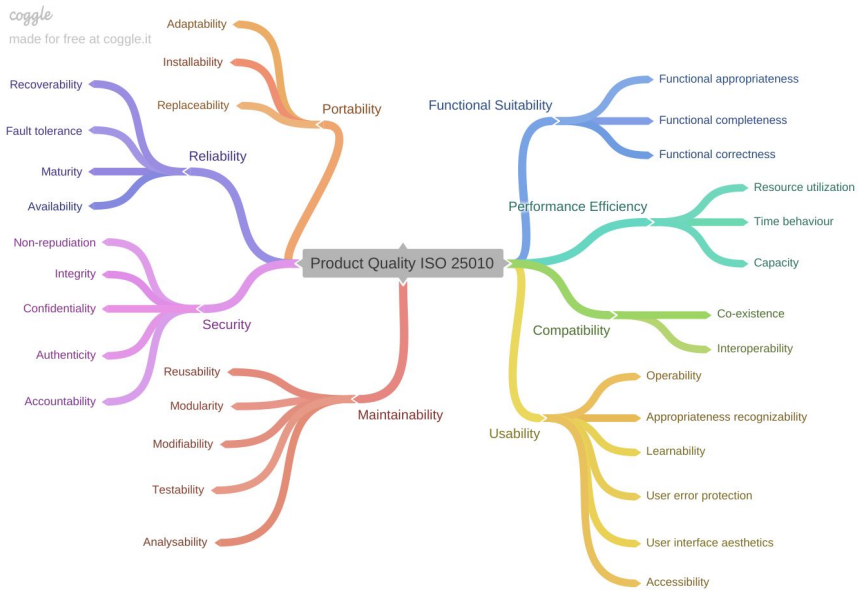


Види тестування

Функціональні види тестування

Нефункціональні види тестування

Пов'язані зі змінами види тестування



Функціональні види тестування

- Функціональне тестування (Functional testing)
- Тестування інтерфейсу користувача (GUI Testing)
- Тестування безпеки (Security and Access Control Testing)
- Тестування взаємодії (Interoperability Testing)

Функціональне тестування (Functional testing)

Функціональне тестування — це тип тестування програмного забезпечення, спрямований на перевірку функцій або функціональних вимог програми. Метою є переконатися, що система виконує завдання, визначені у технічних вимогах, і все працює так, як це передбачено. Функціональне тестування може бути як ручним, так і автоматизованим.

Тестування інтерфейсу користувача (GUI Testing)

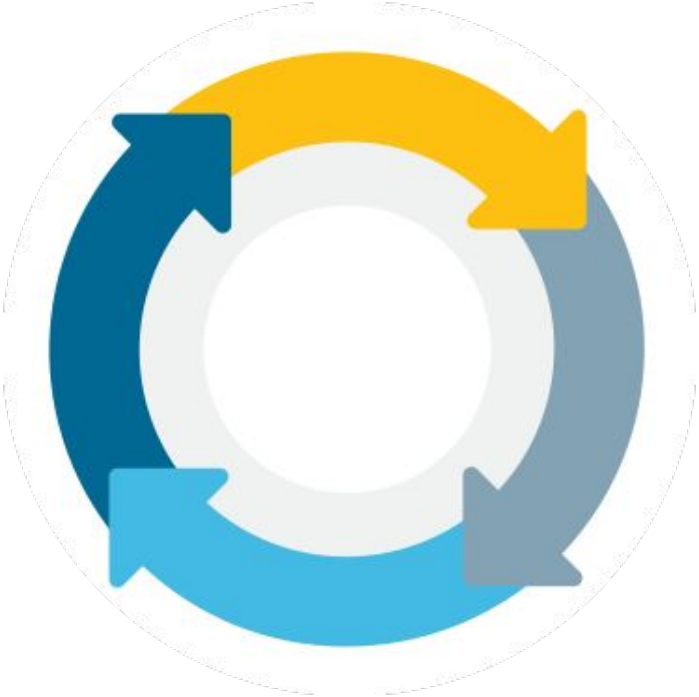
Тестування інтерфейсу користувача (GUI Testing, або Graphical User Interface Testing) — це процес перевірки функціональності графічного інтерфейсу програмного забезпечення. Основна мета полягає у забезпеченні коректної роботи всіх візуальних елементів, таких як кнопки, меню, поля для введення тексту, лейбли, панелі навігації та інші, а також правильність їх взаємодії з користувачем.

Тестування безпеки (Security and Access Control Testing)

Тестування безпеки — це процес ідентифікації потенційних слабких місць в системі з метою виявлення можливих вразливостей, що можуть бути використані зловмисниками. Даний тип тестування забезпечує конфіденційність, цілісність та доступність інформації в системі.

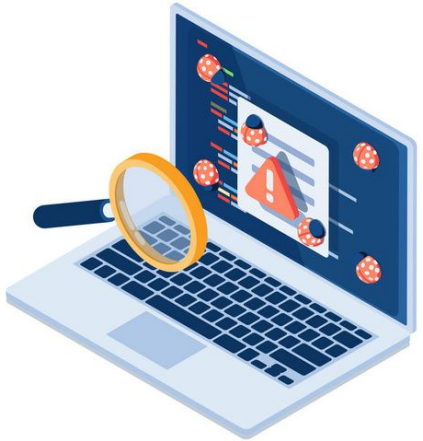
Тестування взаємодії (Interoperability Testing)

Тестування взаємодії (Interoperability Testing) — це процес перевірки здатності системи або компонента спілкуватися та працювати взаємозамінно з іншими системами або компонентами. Ця форма тестування важлива в сучасному світі, де програмні продукти часто інтегровані з іншими застосунками, базами даних, мережами тощо.



Нефункціональні види тестування

- Види тестування продуктивності
- Тестування установки (Installation testing)
- Тестування зручності користування (Usability Testing)
- Тестування на відмову та відновлення (Failover and Recovery Testing)
- Конфігураційне тестування (Configuration Testing)



Види тестування продуктивності

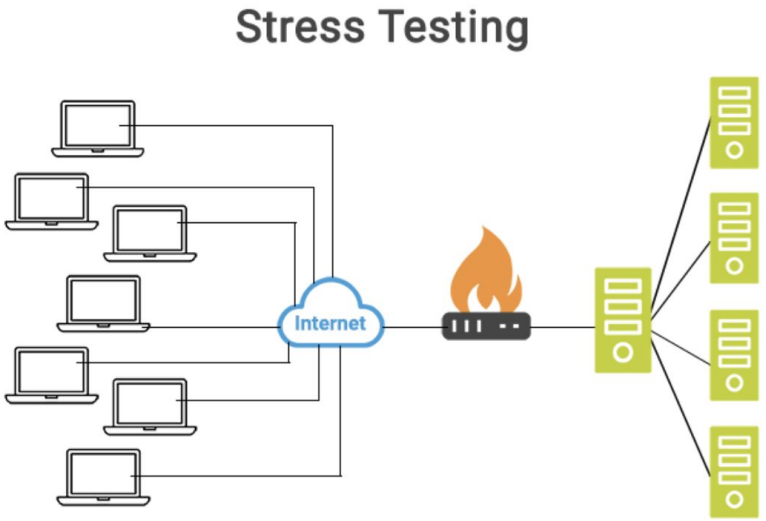
- навантажувальне тестування (Performance and Load Testing)
- стресове тестування (Stress Testing)
- тестування стабільності чи надійності (Stability / Reliability Testing)
- об'ємне тестування (Volume Testing)

Навантажувальне тестування (Performance and Load Testing)

Це автоматизоване тестування, яке імітує роботу певної кількості бізнес-користувачів на загальному (спільному для них) ресурсі.

Стресове тестування (Stress Testing)

Тестування на стрес (Stress Testing) дозволяє перевірити, наскільки працездатним є застосунок і система в цілому в умовах стресу, а також оцінити здатність системи до відновлення, тобто повернення до нормального стану після припинення впливу стресового навантаження. В даному контексті стресом може бути збільшення інтенсивності виконання операцій до дуже високих значень або аварійна зміна конфігурації сервера. Також одним із завдань під час стрес-тестування може бути оцінка деградації продуктивності, тому цілі стрес-тестування можуть перекриватися з цілями тестування продуктивності.

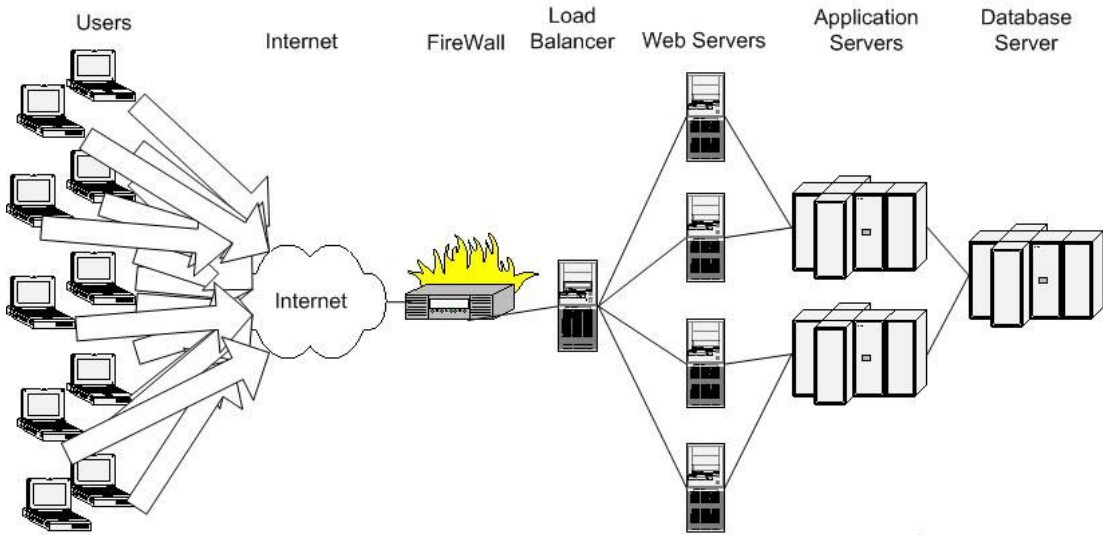


Тестування стабільності чи надійності (Stability / Reliability Testing)

Тестування стабільності або надійності (Stability / Reliability Testing) має на меті перевірку працездатності додатка під час тривалого (багатогодинного) тестування з середнім рівнем навантаження.

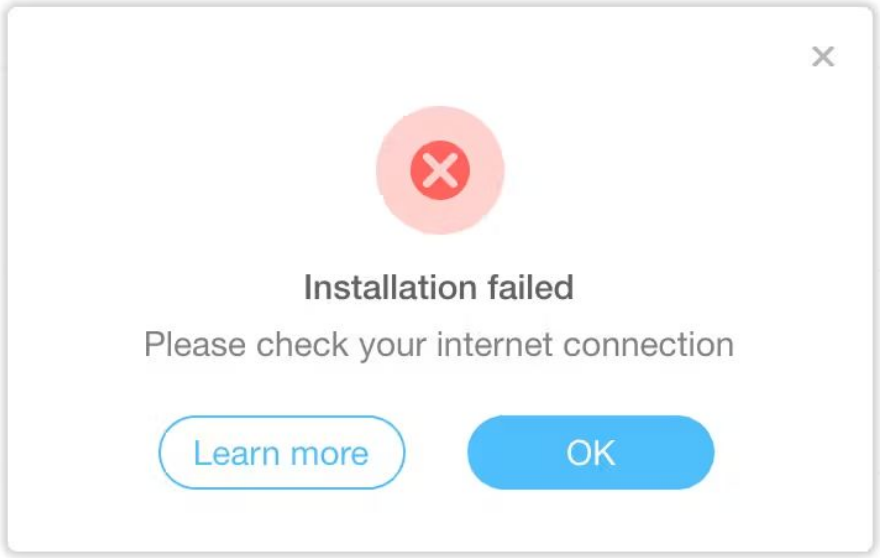
Об'ємне тестування (Volume Testing)

Обсягове тестування (Volume Testing) має на меті отримання оцінки продуктивності при збільшенні обсягів даних у базі даних застосунку.



Тестування установки (Installation testing)

Тестування установки (Installation Testing) — це тип тестування, який зосереджений на перевірці того, чи правильно програмний продукт встановлюється на цільову систему. Мета — забезпечити, що користувач може без проблем встановити та запустити програму на своєму обладнанні.



Тестування зручності користування (Usability Testing)

Тестування зручності користування (Usability Testing) — це процес оцінки продукту з точки зору кінцевого користувача з метою виявлення проблем в інтерфейсі, дизайні, навігації та загалом користувацькому досвіду (UX). Цей вид тестування допомагає визначити, наскільки легко та ефективно користувачі можуть виконувати необхідні завдання з допомогою продукту.

Тестування на відмову та відновлення (Failover and Recovery Testing)

Тестування на відмову та відновлення (Failover and Recovery Testing) — це процес перевірки здатності системи або компонента відновитися після несподіваних відмов або збоїв. Це допомагає забезпечити, що система може продовжувати функціонувати або швидко відновити свою роботу після таких інцидентів.

Конфігураційне тестування (Configuration Testing)

Конфігураційне тестування (Configuration Testing) — це вид тестування, де основна увага приділяється перевірці продукту в різних конфігураціях апаратного та програмного забезпечення. Ціль полягає в тому, щоб переконатися, що програмний продукт буде коректно працювати в різних умовах, що можуть включати різні версії операційних систем, браузерів, мережевих налаштувань та ін.

Пов'язані зі змінами види тестування

- Димове тестування (Smoke Testing)
- Регресійне тестування (Regression Testing)
- Повторне тестування (Re-testing)
- Тестування складання (Build Verification Test)
- Санітарне тестування або перевірка узгодженості/справності (Sanity Testing)

Димове тестування (Smoke Testing)

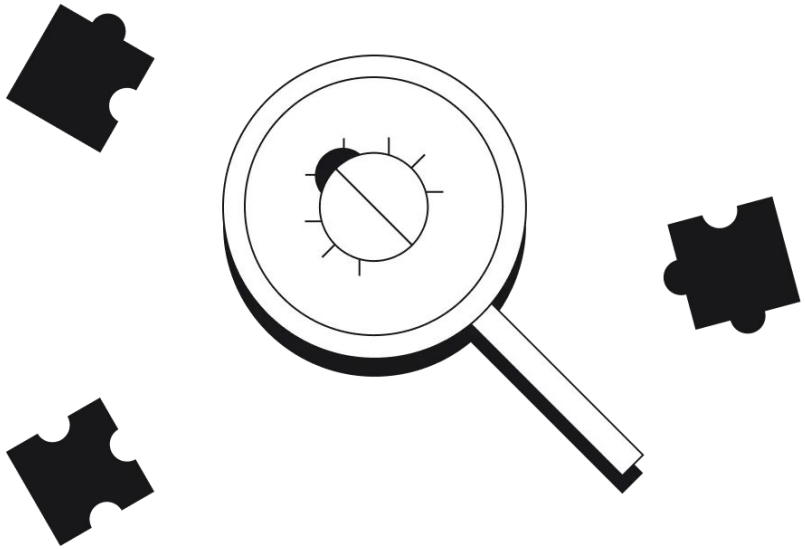
Димове тестування (Smoke Testing) — це базовий вид тестування, який виконується для визначення, чи готова дана збірка програмного забезпечення до подальшого, більш детального тестування. Цей тип тестування допомагає команді швидко зрозуміти, чи працює основна функціональність програми, і чи варто витрачати ресурси на детальніше тестування.

Регресійне тестування (Regression Testing)

Регресійне тестування (Regression Testing) — це тип тестування, що застосовується для перевірки, чи нові зміни в коді (як-от додавання нових функцій, виправлення помилок, оптимізації) не зашкодили існуючій функціональності системи. Мета регресійного тестування — виявити випадки, коли зміни в одній частині програми призводять до збоїв у вже тестованих частинах.

Повторне тестування (Re-testing)

Повторне тестування (Re-testing) — це процес перевірки конкретних дефектів або проблем, які були виявлені та виправлені, для забезпечення їхнього вирішення. Повторне тестування фокусується на одному або декількох конкретних випадках та виконується з метою переконатися, що виправлені дефекти дійсно були виправлені та більше не виникають.

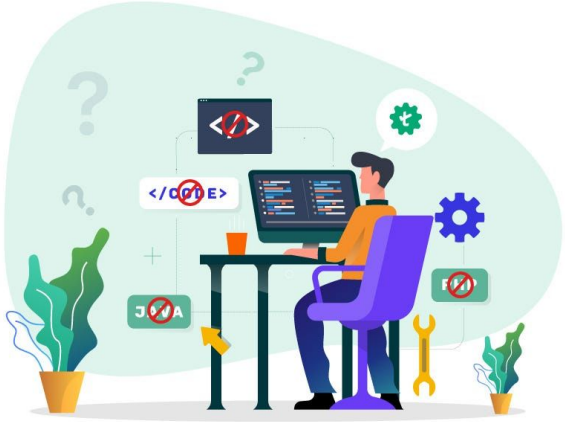


Тестування складання (Build Verification Test)

Тестування складання, або Build Verification Test (BVT), — це набір тестових випадків, які запускаються після кожного нового збирання продукту з метою перевірки, чи "зібраний" продукт гідний для подальшого тестування. Ці тести слід забезпечити мінімальну перевірку основних функціональних можливостей та характеристик програмного продукту.

Санітарне тестування або перевірка узгодженості/справності (Sanity Testing)

Санітарне тестування, або Sanity Testing, — це поверхневий тип тестування, який виконується для перевірки деяких базових функцій програмного продукту після внесення невеликих змін в код. Це допомагає переконатися, що зміни не призвели до серйозних регресій або виходу системи з ладу в ключових аспектах.



Підходи до функціонального тестування

1. Тестування чорного ящика
2. Тестування білого ящика
3. Тестування сірого ящика

Тестування чорної ящика

Тестування чорного ящика приймає масив вхідних даних і шукає генерацію визначених виходів. Ідея цієї назви полягає в тому, що вміст коду, що тестується, невідомий досліднику і, за визначенням, тестувальнику, який займається лише перевіркою функцій.

Тестування білого ящика

Тести білого ящика знаходяться на іншому кінці спектру. Вони засновані на точному знанні того, що відбувається з тестованим кодом, і тести виконуються насамперед для перевірки надійності коду, а не його абсолютної функціональності.

Тестування білого ящика виконується на початку процесу розробки за допомогою модульних тестів і на ранніх етапах фази інтеграції. Тестування чорного ящика є типовим для останніх етапів, де важлива реакція на конкретні сценарії роботи.

Тестування сірого ящика

Тестування сірого ящика — це підхід в тестуванні програмного забезпечення для тестування програмного продукту або програми з частковим знанням внутрішньої структури програми. Метою тестування сірого ящика є пошук і виявлення дефектів через неправильну структуру коду або неправильне використання програм.

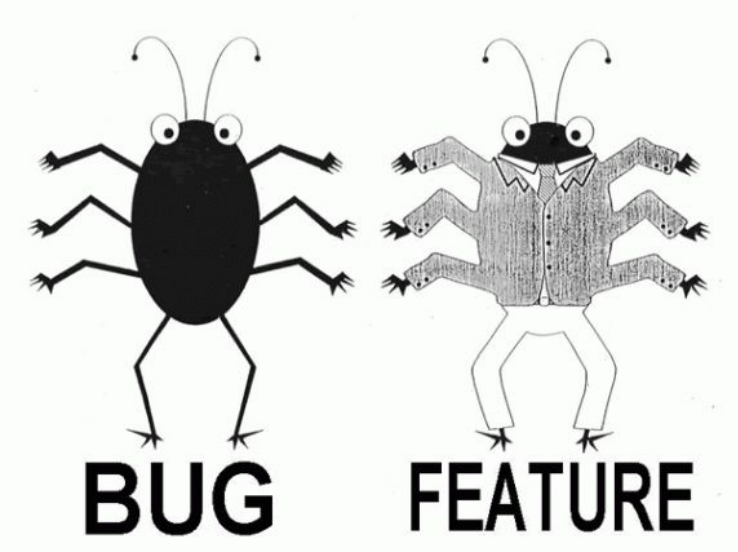
Техніки тест дизайну

Еквівалентний Поділ (Equivalence Partitioning — EP)
Аналіз Граничних Значень (Boundary Value Analysis — BVA)
Причина / Наслідок (Cause/Effect — CE)
Передбачити помилку (Error Guessing — EG)
Вичерпне тестування (Exhaustive Testing — ET)
Попарне тестування (Pairwise Testing)

Incident? Defect? Bug?

Хибнонегативні (False negatives) - це тести, які не виявляють дефектів, які вони мали б виявити; помилкові спрацьовування повідомляються як дефекти, але насправді не є дефектами.

Помилкові спрацьовування (False positives) - можуть виникати через помилки у способі виконання тестів, дефекти тестових даних, тестового середовища чи іншого тестового програмного забезпечення чи інших причин.



Bug Report

ID/ім’я : будьте короткими та використовуйте правильні терміни. Найкраще вказати назву функції, у якій ви виявили проблему. Хорошим прикладом може бути «КОШИК – Неможливо додати новий товар у кошик»

Опис/резюме: якщо ви вважаєте, що назва недостатня, поясніть помилку кількома словами. Поділіться ним легкою для розуміння мовою. Майте на увазі, що ваш опис може бути використаний для пошуку у вашій програмі відстеження помилок, тому використовуйте правильні слова.

Середовище: залежно від вашого браузера, операційної системи, рівня масштабування та розміру екрана веб-сайти можуть по-різному поводитися в одному середовищі. Переконайтеся, що ваші розробники знають ваше технічне середовище.

Вихідна інформація: інформація яка допоможе швидше знайти місце помилки та саму помилку (кеш, посилання на сторінку, помилка в консолі ...)

Візуальний доказ: зображення варте тисячі слів. Хоча цього може бути недостатньо, візуальний елемент, наприклад знімок екрана чи відео, допоможе вашим розробникам краще та швидше зрозуміти проблему.

Дії для відтворення: знімок екрана є доказом того, що у вас виникла проблема, але майте на увазі, що ваш розробник може не вдатися відтворити помилку. Обов’язково якомога докладніше опишіть кроки, які ви виконали до того, як виявили помилку.

Очікувані та фактичні результати (Expected vs. actual results) - поясніть, яких результатів ви очікували – будьте якомога конкретнішими. Просто сказати, що «додаток працює не так, як очікувалося», не має користі. Також корисно описати те, що ви насправді пережили.

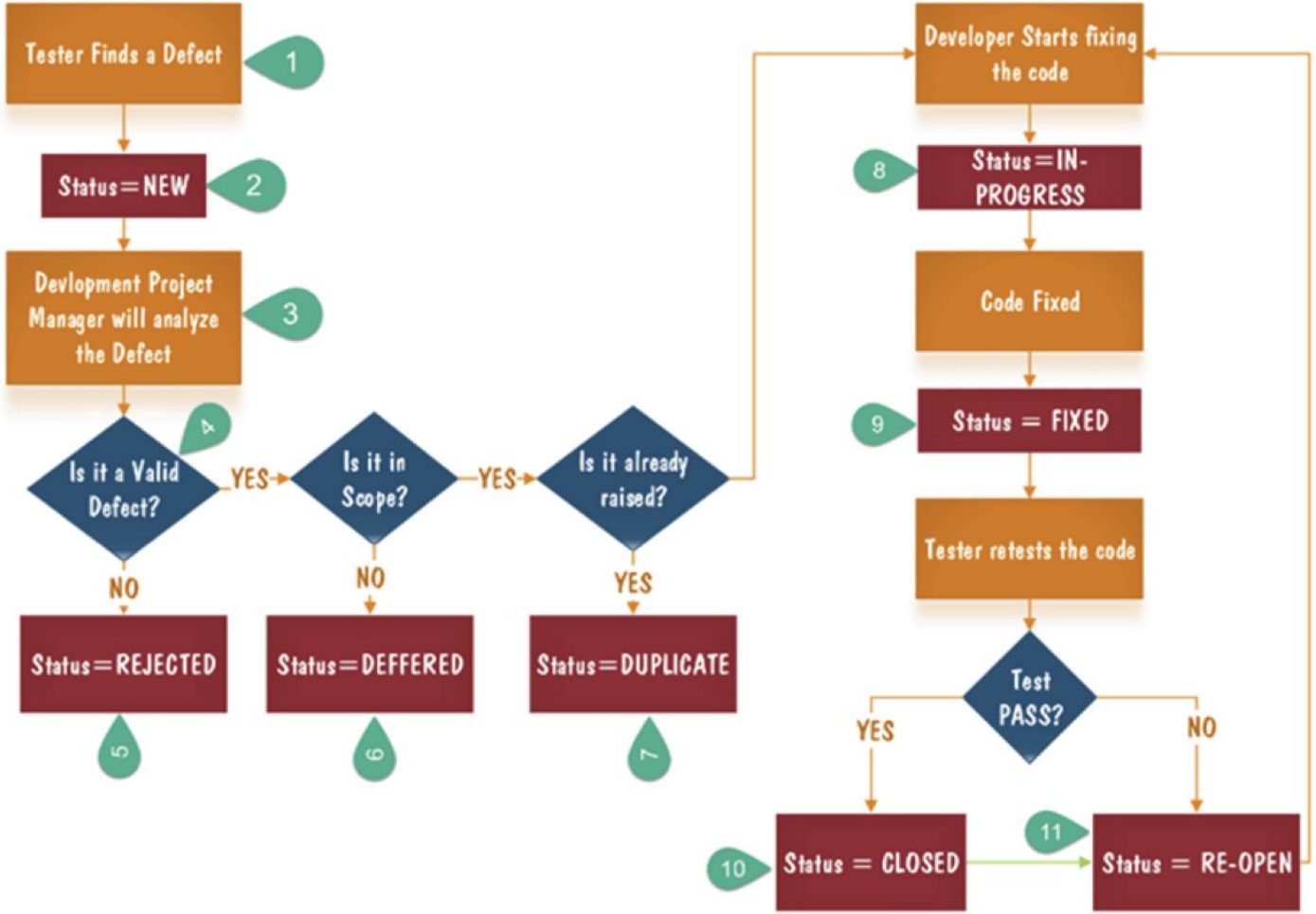
Severity and Priority: Ви також можете включити додаткову інформацію, таку як серйозність (критична, велика, другорядна, тривіальна, покращення), пріоритет (високий, середній, низький), ім’я особи, яка повідомляє, призначену особу або термін виконання.

Вплив помилки на додаток.

Вплив помилки на бізнес клієнта.

Тестери вибирають ступінь серйозності помилки, а керівник проекту або керівник проекту вибирає пріоритет помилки.

Життєвий цикл помилки



Verification and Validation

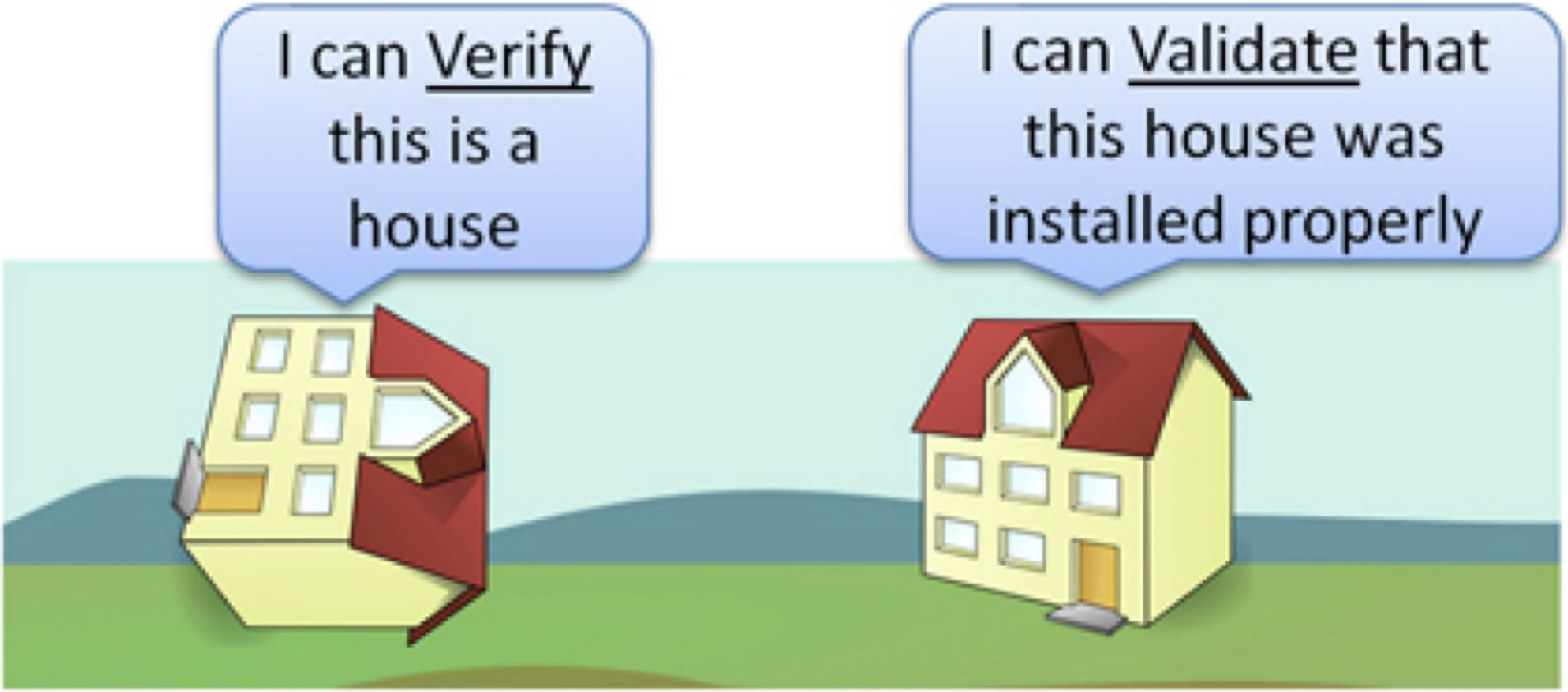
Верифікація – це статична діяльність, яка потребує тестування

- **Документи**
- **Дизайн**
- **Код**
- **Тестові випадки**
- **Контрольні списки**

Перевіряє, чи система розроблена правильно, не гарантує корисність, передує перевірці

Валідація – процес оцінки якості, перевірка відповідності програмного забезпечення потребам зацікавлених сторін.

Перевіряє, чи правильно розвивається система, динамічний тип тестування


I can Verify this is a house
I can Validate that this house was installed properly