ІНДЗ

Тема

Методи тестування програмного забезпечення

Виконували

Шпак Максим

Волошко Максим

Черкаси 2020,

Зміст

Вступ…………………………………………………………………………....3

Основні визначення…………………………………………………………....4

Види тестування……………………………………………………………….6

Функціональне тестування якості………………………………………….…8

Розробка тестових випадків………………………………………………….11

Список використаної літератури…………………………………………….15

*Вступ*

Історія тестування програмного забезпечення відображає еволюцію розробки самого програмного забезпечення. Протягом тривалого часу розробка програмного забезпечення приділяла основну увагу великомасштабним науковим програмам, а також програмам міністерства оборони, пов'язаним з системами корпоративних баз даних, які проектувалися на базі універсальної ЕОМ або мінікомп'ютера. Тестові сценарії записувалися на папір. З їх допомогою перевірялися цільові потоки управління, обчислення складних алгоритмів та маніпулювання даними. Остаточний набір тестових процедур міг ефективно протестувати всю систему повністю. Тестування зазвичай починалося лише після завершення плану-графіка проекту і виконувалося тим же персоналом.

«Тестування - процес, що підтверджує правильність програми і демонструє, що помилок у програмі немає.» Основний недолік подібного визначення полягає в тому, що воно абсолютно неправильно; фактично це майже визначення антоніма слова «тестування». Люди з деяким досвідом програмування вже, ймовірно, розуміє, що неможливо продемонструвати відсутність помилок у програмі. Тому визначення описує нездійсненне завдання, а так як тестування найчастіше все ж виконується з успіхом, принаймні з деяким успіхом, то таке визначення логічно некоректно. Правильне визначення тестування таке: Тестування - процес виконання програми з наміром знайти помилки.

Неможливо гарантувати відсутність помилок в нетривіальною програмі; в кращому випадку можна спробувати показати наявність помилок. Якщо програма правильно поводиться для солідного набору тестів, немає підставі стверджувати, що в ній немає помилок; з усією визначеністю можна лише стверджувати, що не відомо, коли ця програма не працює. Причиною багатьох нещасть розробників є програмні помилки, через які на їх багатостраждальні голови звалюються і давно прострочені проекти, і безсонні ночі. Помилки можуть зробити життя розробників дійсно нещасною, бо, досить кільком помилкам украстися в їх програми, як замовники припиняють цими програмами користуватися, а самі вони можуть втратити роботу.

Надійність неможливо внести в програму в результаті тестування, вона визначається правильністю етапів проектування. Найкраще рішення проблеми надійності - з самого початку не допускати помилок у програмі.

*Основні визначення*

Хоча в тестуванні можна виділити кілька різних процесів, такі терміни, як тестування, налагодження, доказ, контроль та випробування, часто використовуються як синоніми і, на жаль, для різних людей мають різний зміст. Хоча стандартних, загальноприйнятих визначень цих термінів немає, спроба сформулювати їх була зроблена на симпозіумі з тестування програм. Класифікацію різних форм тестування я почну з того, що дам ці визначення, злегка доповнивши і розширивши їх список.

Тестування (testing), як ми вже з'ясували, -процес виконання програми (або частини програми) з наміром (або метою) знайти помилки.

Доказ (proof) - спроба знайти помилки в програмі безвідносно до зовнішньої для програми середовищі. Більшість методів докази передбачає формулювання тверджень про поведінку програми і потім висновок і доказ математичних теорем про правильність програми. Докази можуть розглядатися як форма тестування, хоча вони і не передбачають прямого виконання програми. Багато дослідників вважають доказ альтернативою тестуванню - погляд багато в чому помилковий.

Контроль (verification) - спроба знайти помилки, виконуючи програму в тестовій, або модельованої, середовищі.

Випробування (validation) - спроба знайти помилки, виконуючи програму в заданій реальному середовищі.

Атестація (certification) - авторитетне підтвердження правильності програми, аналогічне атестації електротехнічного обладнання Underwriters Laboratories. При тестуванні з метою атестації виконується порівняння з деяким наперед визначеним стандартом.

Налагодження (debugging) не є різновидом тестування. Хоча слова «налагодження» і «тестування» часто використовуються як синоніми, під ними маються на увазі різні види діяльності. Тестування - діяльність, спрямована на виявлення помилок; налагодження спрямовано встановлення точної природи відомої помилки, а потім - на виправлення цієї помилки. Ці два види діяльності пов'язані - результати тестування є вихідними даними для налагодження.

Тестування модуля, або автономне тестування (module testing, unit testing) - контроль окремого програмного модуля, зазвичай в ізольованому середовищі (т. е. ізольовано від всіх інших модулів). Тестування модуля іноді включає також математичне доказ.

Тестування сполученні (integration testing) - контроль сполученні між частинами системи (модулями, компонентами, підсистемами).

Тестування зовнішніх функцій (external function testing) - контроль зовнішнього поводження системи, певного зовнішніми специфікаціями.

Комплексне тестування (system testing) - контроль та/або випробування системи по відношенню до вихідних цілям. Комплексне тестування є процесом контролю, якщо воно виконується в моделюється середовищі, і процесом випробування, якщо виконується в середовищі реальної, життєвої.

Тестування прийнятності (acceptance testing) - перевірка відповідності програми вимогам користувача.

Тестування налаштування (installation testing) - перевірка відповідності кожного конкретного варіанту установки системи з метою виявити будь-які помилки, що виникли в процесі налаштування системи.

*Види тестування ПЗ*

Тестування ПЗ - це процес його дослідження з метою отримання інформації про якість. Метою тестування є виявлення дефектів у ПЗ. За допомогою тестування не можна довести відсутність дефектів і коректність функціонування аналізованої програми. Тестування складних програмних продуктів є творчим процесом, що не сводящимся до слідування строгим і чітким процедурам.

Тестування програмного забезпечення охоплює цілий ряд видів діяльності, дуже аналогічний послідовності процесів розробки програмного забезпечення. Сюди входять постановка задачі для тесту, проектування, написання тестів, тестування тестів і, нарешті, виконання тестів і вивчення результатів тестування. Вирішальну роль відіграє проектування тесту.

Склад і зміст документації, супутньої процесу тестування, визначається закордонним стандартом IEEE 829-2008 Standard for Software Test Documentation.

Існує кілька підстав, за якими прийнято виробляти класифікацію видів тестування.

. По об'єкту тестування

· Функціональне тестування (functional testing)

· Навантажувальне тестування (performance/load/stress testing)

· Тестування зручності використання (usability testing)

· Тестування інтерфейсу користувача (UI testing)

· Тестування безпеки (security testing)

· Тестування локалізації (localization testing)

· Тестування сумісності (compatibility testing)

. За знанням про тестованої системі

· Тестування методом «чорного ящика» (black box)

· Тестування методом «білого ящика» (white box)

· Тестування методом «сірого ящика» (grey box)

. За рівнем автоматизації

· Ручне тестування (manual testing)

· Автоматизоване тестування (automated testing)

. За ступенем ізольованості

та інші

*Функціональне тестування та тестування якості*

Функціональне тестування проводиться для перевірки виконання системою функціональних вимог.

Навантажувальне тестування проводиться для аналізу роботи системи при різних рівнях навантаження (великі обсяги даних або кількість користувачів). За допомогою навантажувального тестування можна експериментально визначити вимоги до ресурсів, масштабованість і надійність створеної системи. З погляду замовника системи, тестування навантаження є одним із способів перевірки роботи системи в умовах, наближених до реальних.

Основними показниками продуктивності інформаційної системи, вимірюваними в ході навантажувального тестування, є:

· Час відгуку (час виконання операції)

· Число операцій, які виконуються в одиницю часу (наприклад, transactions per second, TPS).

Основним результатом навантажувального тестування є вимірювання продуктивності інформаційної системи, які можуть бути використані для локалізації вузьких місць і подальшої оптимізації. У процесі навантажувального тестування може бути побудована «крива деградації» - графік, що показує залежність продуктивності системи (наприклад, в одиницях часу відгуку) від робочого навантаження (наприклад, від числа віртуальних користувачів).

Стресовий (stress) тестування проводиться в умовах недостатніх системних ресурсів і дозволяє оцінити рівень надійності роботи системи під навантаженням.

Тестування зручності використання має на меті оцінити прийнятність користувальницького інтерфейсу додатку (час, витрачений на досягнення мети, отриманий результат, легкість доступу до потрібної інформации, інтерпретація відповідей системи і т.д.)

Щоб охопити всі аспекти зручності використання, поряд з фахівцями щодо забезпечення якості в цьому вигляді тестування можуть брати участь фахівці з маркетингу і психологи, а також майбутні користувачі системи. У процесі тестування користувачам під керівництвом модератора пропонується вирішити з використанням системи ряд завдань. Для подальшого аналізу результатів тестування необхідна чітка фіксація всіх подій, що відбуваються: використання клавіатури і миші, вираз обличчя респондента, переходи між екранами, мова модератора і респондента.

Тестування інтерфейсу користувача (UI testing) передбачає перевірку відповідності ПЗ вимогам до графічного інтерфейсу користувача. Розрізняють такі види тестування графічного інтерфейсу користувача:

· Тестування на відповідність стандартам графічних інтерфейсів;

· Тестування з різними дозволами екрану;

· Тестування локалізованих версій: перевірка довжини назв елементів інтерфейсу і т.п .;

· Тестування графічного інтерфейсу користувача на різних цільових пристроях (для мобільних додатків, можливо з використанням емуляторів).

У ході тестування безпеки (security testing) проводиться оцінка уразливості системи по відношенню до атак. Тестування безпеки перевіряє фактичну реакцію захисних механізмів, вбудованих в систему, на спроби їх злому і обходу. У ході тестування безпеки випробувач грає роль потенційного порушника і намагається перевірити наступні аспекти безпеки системи:

· Тестування механізмів контролю доступу - допомагає виявити дефекти, в результаті яких користувачі можуть отримувати несанкціонований доступ до об'єктів і функцій програми;

· Тестування авторизації користувачів - виявляє дефекти, пов'язані з авторизацією окремих користувачів і г8рупп користувачів і з перевіркою їх автентичності;

· Тестування процедур перевірки коректності введення - має на меті виявлення помилок в процедурах перевірки даних, що надходять в систему ззовні;

· Тестування криптографічних механізмів захисту - використовується для виявлення дефектів, пов'язаних з шифруванням і розшифрування даних, використанням цифрових підписів і перевіркою цілісності даних;

· Тестування правильності обробки помилок - включає в себе перевірку таких аспектів, як вивід на екран фрагментів коду за ощибки, вплив помилок на роботу всього програми, аналіз помилок в коді їх обробки;

· Тестування конфігурації сервера - допомагає виявити помилки, пов'язані з розкриттям конфігурації апаратних і програмних засобів, а також з некоректними налаштуваннями параметрів безпеки серверного ПЗ.

Завдання проектувальника системи полягає в тому, щоб зробити витрати на організацію атаки вище, ніж ціна одержуваної в результаті інформації.

У процесі тестування локалізації (localization testing) перевіряються різні аспекти, пов'язані з регіональними особливостями (перевірка роботи різних мовних версій, систем вимірювань, форматів дат, нумерації днів тижня, порядку сортування і т.д.)

Тестування сумісності (compatibility testing) - перевірка сумісності системи з різними варіантами програмно-апаратного оточення (операційними системами, різними браузерами, мережевим ПЗ, СУБД, стороннім ПЗ, апаратною платформою).

*Розробка тестових випадків (test case)*

Тестування на різних рівнях виробляється протягом усього життєвого циклу розробки і супроводу ПЗ. Рівень тестування визначає те, над чим виробляються тести: над окремим модулем, групою модулів або системою, у цілому. Проведення тестування на всіх рівнях системи - це основа успішної реалізації і здачі проекту. Рівні тестування: • компонентне або модульне тестування (Component Testing or Unit Testing) • інтеграційне тестування (Integration Testing) • системне тестування (System Testing) • приймальне тестування (Acceptance Testing) Компонентне (модульне) тестування (Component or Unit Testing) перевіряє функціональність і шукає дефекти в частинах додатка, які доступні і можуть бути протестовані окремо (модулі програм, об'єкти, класи, функції тощо). Звичайно компонентне (модульне) тестування проводиться викликаючи код, який необхідно перевірити і за підтримкою середовищ розробки, таких як фреймворки (frameworks - каркаси) для модульного тестування або інструменти для налагодження. Усі знайдені дефекти, як правило виправляються в коді без формального їхнього опису в системі менеджменту помилок/дефектів - багів (Bug Tracking System). Один з найбільш ефективних підходів до компонентного (модульного) тестування - це підготовка автоматизованих тестів до початку основного кодування (розробки) програмного забезпечення. Це називається «розробка від тестування» (test-driven development) або «підхід тестування спочатку» (test first approach). При цьому підході створюються й інтегруються невеликі частини коду, напроти яких запускаються тести, написані до початку кодування. Розробка ведеться доти поки всі тести не будуть успішними. Інтеграційне тестування (Integration Testing) призначене для перевірки зв'язку між компонентами, а також взаємодії з різними частинами системи (операційною системою, устаткуванням або зв'язком між різними системами). Рівні інтеграційного тестування: • компонентний інтеграційний рівень (Component Integration testing). Перевіряється взаємодія між компонентами системи після проведення компонентного тестування; • системний інтеграційний рівень (System Integration Testing). Перевіряється взаємодія між різними системами після проведення системного тестування. Підходи до інтеграційного тестування: • знизу нагору (Bottom Up Integration). Усі низькорівневі модулі, процедури або функції збираються разом і потім тестуються. Після чого збирається наступний рівень модулів для проведення інтеграційного тестування. Даний підхід вважається корисним, якщо всі або практично всі модулі, рівня, що розробляється готові. Також даний підхід допомагає визначити за результатами тестування рівень готовності додатка; • зверху вниз (Top Down Integration). Спочатку тестуються усі високорівневі модулі, і поступово один за іншим додаються низькорівневі. Усі модулі більш низького рівня симулюються заглушками з аналогічною функціональністю, потім в міру готовності вони заміняються реальними активними компонентами; • великий вибух ("Big Bang" Integration). Всі або практично всі розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, і потім проводиться інтеграційне тестування. Такий підхід дуже гарний для збереження часу. Однак якщо тест кейси і їхні результати записані не вірно, то сам процес інтеграції сильно ускладниться, що стане перешкодою для команди тестування при досягненні основної мети інтеграційного тестування. Основною задачею системного тестування є перевірка як функціональних, так і не функціональних вимог у системі в цілому. При цьому виявляються дефекти, такі як невірне використання ресурсів системи, непередбачені комбінації даних користувальницького рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або невірна функціональність, незручність використання тощо. Для мінімізації ризиків, пов'язаних з особливостями поведінки в системі в будь-якому середовищі, під час тестування рекомендується використовувати оточення максимальне наближене до того, на яке буде встановлений продукт після видачі. Можна виділити два підходи до системного тестування: • на базі вимог (requirements based).Для кожної вимоги пишуться тестові випадки (test cases), що перевіряють виконання даної вимоги; • на базі випадків використання (use case based). На основі представлення про способи використання продукту створюються випадки використання системи (Use Cases). • По конкретному випадку використання можна визначити один або більш сценаріїв. На перевірку кожного сценарію пишуться тест кейси (test cases), які мають бути протестовані. Формальний процес приймального тестування, що перевіряє відповідність системи вимогам і проводиться з метою: • визначення чи задовольняє система приймальним критеріям; • винесення рішення замовником або іншою уповноваженою особою приймається додаток чи ні. Приймальне тестування виконується на підставі набору типових тестових випадків і сценаріїв, розроблених на підставі вимог до даного додатка. Рішення про проведення приймального тестування приймається, коли: • продукт досяг необхідного рівня якості; • замовник ознайомлений із планом приймальних робіт (Product Acceptance Plan) або іншим документом, де описаний набір дій, пов'язаних із проведенням приймального тестування, дата проведення, відповідальні особи тощо. Фаза приймального тестування триває доти, поки замовник не виносить рішення про відправлення додатка на доробку або видачі додатка. Тестовий випадок (test case) - сукупність вхідних даних тесту, умови виконання і очікуваних результатів, які розроблені для конкретної мети. Тестовий випадок - це найменша одиниця тестування, яку можна самостійно виконати від початку до кінця. Шаблони тестового випадку і зразок їх заповнення представлені у додатку В. Розглянемо особливості заповнення полів шаблону тестування. Ідентифікатор тестового випадку - включає номер версії тесту. Власник тесту – ПІБ особи, що експлуатує тест (воно може не співпадати з ПІБ автора тесту). Дата останнього перегляду – ця інформація визначає актуальність тесту. Назва тесту - опис назви тесту, що дозволяє його легко знайти і зрозуміти його призначення. Не рекомендується вживати назви, що не несуть ніякого сенсового навантаження, наприклад, "xxxLLL0123.tst". Місцезнаходження тесту – повна назва шляху, розташування на диску ЕОМ. Технічна вимога, що тестується - унікальний ідентифікатор, який відображається в документах технічних вимог. Мета тестування - формулювання того, що має досягти тест. Конфігурація засобів тестування - специфікація вводу / виводу, умови випробувань. Налаштування на прогін тесту - процедура подібна методиці тестування. Вона передбачає опис дій тестувальника і очікуваних результатів. Якщо настроювання автоматизовані, це виглядає так: run setupSC03.pl. Методика тестування - опис дій тестувальника і очікуваних результатів. Взаємозалежність тестових випадків – ідентифікація будь-якого тестового випадку. Для того, щоб виконання даного тесту починалося при означених умовах, необхідно здійснити прогін попередніх тестів. Очистка тесту – якщо система була переведена в нестійкий стан або дані були зруйнованими, очистка дозволяє усунути подібні ситуації.

Список використаної літератури:

1)Процеси життєвого циклу програмного забезпечення. ДКА України СОУ-Н ДКА 0061:2012

2)Марголин Л. Н. Компьютерные методы обработки информации. Интернет издание.

3)Майерс Г. Надежность программного обеспечения. М:, Мир, 1980

4)Куликов С. С. Тестирование программного обеспечения. Базовый курс. Интернет-издание 2017

5)Канер С., и др. Тестирования программного обеспечения. К:, Диасофт, 2001.

6)Майерс Г. Искусство тестирования программ. М:, Финансы и статистика, 1982

7)Савин Р. Тестирование Дот Ком. М:, «Дело», 2007.

8)Бахтизин В.В. Автоматизация тестирования программного обеспечения. Минск, БГУИР 2012