1. 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 починається 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 невідомо, коли ця програма не працює. Звичайно, якщо є причини вважати даний набір тестів здатним з великою ймовірністю виявити всі можливі помилки, то можна говорити про деяке рівні впевненості у правильності програми, що встановлюється цими тестами.

Більшість людей, поставивши мету (наприклад, показати, що помилок немає), орієнтується у своїй діяльності на досягнення цієї мети. Тестувальник підсвідомо не дозволить собі діяти проти мети, т. Е. Підготувати тест, який виявив би одну з решти в програмі помилок. Оскільки ми всі знаємо, що досконалість у проектуванні та кодуванні будь-якої програми недосяжно і тому кожна програма містить деяку кількість помилок, найбільш плідним застосуванням тестування буде знайти деякі з них. Якщо ми хочемо добитися цього і уникнути психологічного бар'єру, що заважає нам діяти проти поставленої мети, наша мета повинна полягати в тому, щоб знайти якомога більше помилок. Поява персональних комп'ютерів сприяло стандартизації цієї галузі, оскільки програми стали спочатку створюватися для роботи із загальною операційною системою. Впровадження персональних комп'ютерів відкрило нову еру і призвело до швидкого і бурхливому зростанню комерційних розробок. Комерційні програми жорстко боролися за першість і виживання. Користувачі комп'ютерів брали вижило програмне забезпечення як стандарти defacto. Пакетна обробка замінилася системами, що працюють в реальному часі.

Тестування систем реального часу зажадало іншого підходу до проектування тестування через те, що робочі потоки могли викликатися в будь-якому порядку. Ця особливість призвела до появи величезної кількості процедур тестування, здатних підтримати нескінченне число перестановок і сполучень.

Сформулюємо основоположний висновок:

- Якщо ваша мета - показати відсутність помилок, то ви їх знайдете не надто багато.

- Якщо ж ваша мета - показати наявність помилок, ви знайдете значну їх частину.

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.