Chương 1: Tổng quan về kiểm thử phần mềm

Kiểm thử phần mềm là quá trình đánh giá tính năng và chất lượng của một phần mềm để xác định xem nó đáp ứng được các yêu cầu và mong đợi của người dùng hay không. Mục tiêu chính của kiểm thử phần mềm là tìm ra lỗi, các vấn đề tiềm ẩn, và đảm bảo rằng phần mềm hoạt động một cách chính xác, đáng tin cậy và tuân thủ các yêu cầu đặt ra.

Quá trình kiểm thử phần mềm bao gồm các hoạt động sau:

1. Lập kế hoạch kiểm thử: Xác định phạm vi, mục tiêu và phương pháp kiểm thử. Tạo kế hoạch và lịch trình kiểm thử.
2. Thiết kế kiểm thử: Xây dựng kịch bản kiểm thử, tạo các bộ dữ liệu kiểm thử, và xác định các tiêu chí thành công.
3. Thực hiện kiểm thử: Chạy các bộ kiểm thử, thực hiện các thao tác kiểm thử, ghi lại kết quả và lỗi.
4. Phân tích kết quả: Xem xét và đánh giá kết quả kiểm thử, xác định các vấn đề, lỗi và hiệu chỉnh.
5. Báo cáo: Tạo báo cáo kiểm thử, bao gồm kết quả, lỗi phát hiện, đánh giá chất lượng và đề xuất cải tiến.

Chương 2: Kiểm thử phần mềm thủ công và tự động, Test Screnario & Test Case

Có nhiều phương pháp kiểm thử phần mềm khác nhau như kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử chấp nhận người dùng, kiểm thử tải và kiểm thử bảo mật. Kiểm thử phần mềm có thể được thực hiện bằng tay hoặc tự động, sử dụng các công cụ và kỹ thuật phù hợp.

Quá trình kiểm thử phần mềm đóng vai trò quan trọng trong việc cung cấp phần mềm chất lượng và đáng tin cậy cho người dùng, giảm thiểu các rủi ro và tăng cường trải nghiệm người dùng.

Kiểm thử phần mềm có thể được thực hiện bằng hai phương pháp chính là kiểm thử thủ công và kiểm thử tự động. Dưới đây là mô tả tổng quan về hai phương pháp này:

1. Kiểm thử phần mềm thủ công:
   * Kiểm thử phần mềm thủ công là quá trình kiểm thử mà con người thực hiện các hoạt động kiểm thử bằng tay, tức là thử nghiệm và kiểm tra phần mềm bằng các thao tác và kỹ thuật thủ công.
   * Nhân viên kiểm thử sẽ tiến hành kiểm tra chức năng, kiểm tra giao diện người dùng, kiểm tra sự tương tác và xác nhận các kết quả.
   * Kiểm thử thủ công cho phép nhân viên kiểm thử thấy được cách phần mềm hoạt động trong môi trường thực tế và phát hiện các vấn đề mà công cụ tự động không thể nhận biết.
   * Tuy nhiên, kiểm thử thủ công tốn thời gian và công sức, và có thể bị ảnh hưởng bởi sự khác biệt trong kỹ năng và sự chú ý của nhân viên kiểm thử.
2. Kiểm thử phần mềm tự động:
   * Kiểm thử phần mềm tự động là việc sử dụng công cụ và kịch bản tự động để thực hiện các hoạt động kiểm thử một cách tự động.
   * Các công cụ kiểm thử tự động giúp thực hiện các bước kiểm thử một cách nhanh chóng, lặp đi lặp lại và không bị ảnh hưởng bởi sự khác biệt cá nhân.
   * Kịch bản kiểm thử tự động có thể được lập trình hoặc sử dụng các công cụ quay video, kéo và thả hoặc ghi lại các hành động của người dùng.
   * Kiểm thử tự động thường được sử dụng cho các tác vụ lặp đi lặp lại, kiểm thử tích hợp, kiểm thử hiệu năng và kiểm thử tải.
   * Tuy nhiên, kiểm thử tự động có thể gặp khó khăn trong việc kiểm tra các khía cạnh phức tạp của giao diện người dùng và khó phát hiện các lỗi mới và không được định nghĩa trước.

Sự kết hợp giữa kiểm thử phần mềm thủ công và tự động thường được sử dụng để đạt được hiệu quả và độ chính xác cao trong quá trình kiểm thử phần mềm. Kiểm thử tự động được sử dụng để thực hiện các tác vụ lặp đi lặp lại và kiểm thử tích hợp, trong khi kiểm thử thủ công được sử dụng để kiểm tra các khía cạnh phức tạp và tìm kiếm lỗi mới.

1. Test Scenario (Kịch bản kiểm thử):
   * Test scenario là một tập hợp các bước hoặc sự kiện được xác định trước mà người kiểm thử sẽ thực hiện để kiểm tra một tính năng hoặc một phần của phần mềm.
   * Mỗi kịch bản kiểm thử tập trung vào một tình huống hoặc luồng logic cụ thể trong phần mềm.
   * Kịch bản kiểm thử giúp xác định các hoạt động, thao tác và dữ liệu cần thiết để thực hiện kiểm thử một cách hợp lý.
2. Test Case (Trường hợp kiểm thử):
   * Test case là một bộ test cụ thể, chi tiết và độc lập được thiết kế để kiểm tra một tính năng hoặc một phần của phần mềm.
   * Mỗi test case xác định các bước cụ thể mà người kiểm thử sẽ thực hiện, dữ liệu đầu vào và kết quả mong đợi để kiểm tra.
   * Mục tiêu của mỗi test case là kiểm tra một khía cạnh cụ thể của phần mềm và xác định xem nó hoạt động đúng theo mong đợi hay không.
   * Test case thường được mô tả chi tiết và có thể bao gồm các phần như tiền điều kiện, bước thực hiện, dữ liệu đầu vào, kết quả mong đợi và kết quả thực tế.

Tổ chức test scenario và test case tốt là quan trọng trong quá trình kiểm thử phần mềm. Kịch bản kiểm thử giúp định rõ phạm vi và luồng công việc của kiểm thử, trong khi các trường hợp kiểm thử chi tiết và độc lập giúp kiểm tra từng khía cạnh của phần mềm một cách chi tiết và chính xác.

7 nguyên lý kiểm thử phần mềm

1. Testing Shows the Presence of Defects (Kiểm thử chỉ ra sự tồn tại của lỗi):
   * Quá trình kiểm thử được thực hiện để phát hiện lỗi trong phần mềm.
   * Mục tiêu là tìm ra sự tồn tại của lỗi, chứ không phải chứng minh rằng không có lỗi tồn tại.
   * Không thể kiểm thử hoàn toàn để đảm bảo không có lỗi, nhưng kiểm thử giúp giảm rủi ro lỗi trong phần mềm.
2. Exhaustive Testing is Impossible (Kiểm thử toàn diện là không thể):
   * Kiểm thử toàn diện, kiểm thử tất cả các trường hợp có thể xảy ra, là không thể.
   * Với số lượng kịch bản và dữ liệu khả dĩ, việc kiểm thử toàn diện sẽ tốn nhiều thời gian và nguồn lực không khả thi.
   * Thay vào đó, kiểm thử được thiết kế dựa trên rủi ro và ưu tiên để đạt được phạm vi kiểm thử tối ưu.
3. Early Testing (Kiểm thử sớm):
   * Kiểm thử nên được thực hiện sớm trong quy trình phát triển phần mềm.
   * Việc kiểm thử sớm giúp phát hiện lỗi sớm và giảm chi phí sửa chữa.
   * Càng sớm phát hiện lỗi, càng dễ và rẻ hơn để khắc phục chúng.
4. Defect Clustering (Tập trung lỗi):
   * Trong phần mềm, lỗi thường tập trung ở một số phần nhất định.
   * Một số phần của phần mềm có nhiều lỗi hơn so với các phần khác.
   * Tìm hiểu và tập trung kiểm thử vào các khu vực có khả năng cao của lỗi giúp tăng hiệu suất của quá trình kiểm thử.
5. Pesticide Paradox (Hiệu ứng thuốc trừ sâu):
   * Kỹ thuật kiểm thử cố định có thể trở nên vô hiệu sau một thời gian.
   * Nếu sử dụng cùng một bộ kiểm thử lâu dài, các lỗi mới có thể không được phát hiện.
   * Cần cập nhật và tái thiết kế kiểm thử để tìm ra các lỗi mới và không được định nghĩa trước.
6. Testing is Context Dependent (Kiểm thử phụ thuộc vào ngữ cảnh):
   * Quá trình kiểm thử phải phụ thuộc vào ngữ cảnh của phần mềm, yêu cầu và môi trường.
   * Các phương pháp và kỹ thuật kiểm thử phải được tùy chỉnh để phù hợp với yêu cầu cụ thể và ngữ cảnh kiểm thử.
7. Absence-of-Errors Fallacy (Sự hiện diện lỗi không tuyệt đối):
   * Không có lỗi được tìm thấy trong quá trình kiểm thử không có nghĩa là phần mềm không có lỗi.
   * Kiểm thử không thể đảm bảo rằng không có lỗi tồn tại trong phần mềm.
   * Kiểm thử chỉ giới hạn trong phạm vi và thời gian cụ thể, và không thể đại diện cho mọi trường hợp sử dụng và môi trường.

Kiểm thử phần mềm chức năng và phi chức năng, black box, white box, gray box

1. Kiểm thử chức năng:
   * Kiểm thử chức năng tập trung vào việc kiểm tra tính năng và chức năng của phần mềm.
   * Mục tiêu của kiểm thử chức năng là xác định xem phần mềm có thực hiện đúng các chức năng và hoạt động như mong đợi hay không.
   * Kiểm thử chức năng kiểm tra các yêu cầu và phạm vi chức năng, bao gồm kiểm tra chức năng cơ bản, kiểm thử chức năng mở rộng, kiểm thử đặc tả và kiểm thử tương tự.
2. Kiểm thử phi chức năng:
   * Kiểm thử phi chức năng tập trung vào việc kiểm tra các yếu tố phi chức năng của phần mềm, bao gồm hiệu suất, bảo mật, khả năng mở rộng, khả năng khôi phục và tương tác người dùng.
   * Mục tiêu của kiểm thử phi chức năng là xác định xem phần mềm có đáp ứng các yêu cầu phi chức năng và đáp ứng được các yêu cầu chất lượng không liên quan đến chức năng.
   * Kiểm thử phi chức năng bao gồm kiểm thử hiệu suất, kiểm thử bảo mật, kiểm thử tương thích, kiểm thử khả năng khôi phục, kiểm thử khả năng mở rộng và kiểm thử tương tác người dùng.
3. Kiểm thử Black box (Hộp đen):
   * Kiểm thử Black box tập trung vào xem phần mềm hoạt động như thế nào từ bên ngoài, mà không quan tâm đến cấu trúc nội bộ của nó.
   * Kiểm thử viên xem phần mềm như một "hộp đen" mà không biết chi tiết về cách nó được triển khai.
   * Kiểm thử Black box dựa trên mô tả yêu cầu và xác định các kịch bản kiểm thử dựa trên đầu vào và kết quả dự kiến.
   * Mục tiêu của kiểm thử Black box là xác định xem phần mềm có hoạt động đúng theo yêu cầu và đáp ứng được các kịch bản kiểm thử hay không.
4. Kiểm thử White box (Hộp trắng):
   * Kiểm thử White box tập trung vào xem cấu trúc nội bộ của phần mềm và kiểm tra các thành phần, logic và luồng điều khiển.
   * Kiểm thử viên có kiến thức về cấu trúc mã nguồn và các thành phần của phần mềm.
   * Kiểm thử White box sử dụng các kỹ thuật như kiểm thử đường dẫn, kiểm thử điều kiện và kiểm thử đơn vị.
   * Mục tiêu của kiểm thử White box là xác định xem các thành phần và logic nội bộ của phần mềm hoạt động chính xác và tuân thủ các yêu cầu thiết kế.
5. Kiểm thử Gray box (Hộp xám):
   * Kiểm thử Gray box là sự kết hợp giữa kiểm thử Black box và White box.
   * Kiểm thử viên có một số thông tin về cấu trúc nội bộ và logic của phần mềm, nhưng không biết hoàn toàn.
   * Kiểm thử Gray box có thể sử dụng thông tin đó để tạo các kịch bản kiểm thử hiệu quả hơn, nhưng vẫn đánh giá phần mềm từ góc nhìn bên ngoài.
   * Mục tiêu của kiểm thử Gray box là kết hợp lợi thế của cả kiểm thử Black box và White box để đảm bảo tính chính xác và đáng tin cậy của phần mềm.

Bù Life Circle

1. Phát hiện (Discovery): Lỗi được phát hiện bởi kiểm thử viên hoặc người dùng cuối khi sử dụng phần mềm.
2. Ghi nhận (Reporting): Lỗi được ghi nhận và báo cáo vào hệ thống quản lý lỗi, có thể là một công cụ quản lý lỗi như JIRA hoặc Bugzilla.
3. Phân loại (Classification): Lỗi được phân loại theo mức độ ưu tiên và mức độ nghiêm trọng. Điều này giúp định rõ ưu tiên khắc phục lỗi trong quá trình phát triển phần mềm.
4. Tạo lại (Reproduction): Lỗi được xác nhận và tạo lại bởi kiểm thử viên hoặc nhóm phát triển. Quá trình này đảm bảo rằng lỗi có thể tái tạo và kiểm tra để đảm bảo tính nhất quán và khả năng lặp lại.
5. Xác nhận (Confirmation): Lỗi được xác nhận là thực sự là một lỗi và không phải là một lỗi giả. Quá trình này thường được thực hiện bởi kiểm thử viên hoặc nhóm phát triển.
6. Gán (Assignment): Lỗi được gán cho người phụ trách để xử lý và khắc phục. Người phụ trách có thể là một nhà phát triển hoặc thành viên trong nhóm phát triển.
7. Khắc phục (Fixing): Lỗi được sửa chữa bởi người phụ trách. Người phụ trách sẽ tìm hiểu và khắc phục lỗi trong mã nguồn hoặc trong quá trình phát triển phần mềm.
8. Kiểm tra (Testing): Lỗi đã được sửa chữa được kiểm tra lại để đảm bảo rằng lỗi đã được khắc phục một cách chính xác và không gây ra các tác động phụ.
9. Đóng (Closure): Khi lỗi đã được khắc phục và xác nhận là sửa, nó được đóng và đóng lại trong hệ thống quản lý lỗi.

Good bug report

1. Tiêu đề rõ ràng: Tiêu đề nên tóm tắt ngắn gọn vấn đề hoặc lỗi cụ thể mà người báo cáo gặp phải.
2. Mô tả chi tiết: Báo cáo nên cung cấp mô tả chi tiết về lỗi, bao gồm các bước để tái hiện lỗi và mô tả kết quả hiện tại không đúng.
3. Ngữ cảnh và điều kiện: Nếu có, cung cấp thông tin về ngữ cảnh và điều kiện trong quá trình xảy ra lỗi, ví dụ: phiên bản phần mềm, hệ điều hành, trình duyệt, dữ liệu đầu vào, vv.
4. Kết quả mong đợi: Đưa ra mô tả rõ ràng về kết quả mong đợi khi sử dụng phần mềm.
5. Kết quả thực tế: Mô tả kết quả thực tế bạn gặp phải khi tái hiện lỗi, bao gồm thông báo lỗi, hành vi không mong đợi và bất kỳ thông tin nào hữu ích khác.
6. Đính kèm tài liệu hỗ trợ: Nếu có, đính kèm tài liệu hỗ trợ như ảnh chụp màn hình, video, tệp tin log, vv., để giúp hiểu rõ hơn về lỗi.
7. Ưu tiên và ảnh hưởng: Xác định mức độ ưu tiên của lỗi và ảnh hưởng của nó đến chức năng hoặc trải nghiệm người dùng.

Một báo cáo lỗi tốt sẽ giúp đội phát triển hiểu và tái hiện lỗi một cách chính xác và nhanh chóng, tăng khả năng khắc phục lỗi và cải thiện chất lượng phần mềm.

CI (Continuous Integration) trong kiểm thử phần mềm là một phương pháp liên tục tích hợp các thành phần phần mềm, bao gồm mã nguồn, kiểm thử và triển khai. CI đảm bảo rằng các thay đổi trong mã nguồn được tích hợp và kiểm thử một cách tự động và liên tục, giúp đảm bảo tính ổn định và chất lượng của sản phẩm phần mềm. Dưới đây là một số khái niệm chính liên quan đến CI trong kiểm thử phần mềm:

1. Tích hợp liên tục (Continuous Integration): Tích hợp liên tục là quá trình tự động hóa việc tích hợp các thay đổi từ các thành viên trong nhóm phát triển vào một phiên bản chung của phần mềm. Điều này bao gồm việc hợp nhất mã nguồn, xây dựng và chạy các bộ kiểm thử cơ bản để đảm bảo tính ổn định và chất lượng của hệ thống.
2. Kiểm thử tự động (Automated Testing): Kiểm thử tự động trong CI là quá trình thực hiện các bộ kiểm thử tự động một cách liên tục để phát hiện và báo cáo lỗi. Điều này bao gồm việc xây dựng các kịch bản kiểm thử tự động, triển khai và thực thi chúng trong môi trường tự động.
3. Xây dựng liên tục (Continuous Build): Xây dựng liên tục là quá trình tự động hóa việc xây dựng và tạo ra phiên bản mới nhất của phần mềm sau mỗi lần tích hợp. Điều này đảm bảo rằng mã nguồn được biên dịch và xây dựng một cách liên tục, giúp phát hiện và sửa lỗi kịp thời.
4. Triển khai liên tục (Continuous Deployment): Triển khai liên tục là quá trình tự động hóa việc triển khai phiên bản mới nhất của phần mềm vào môi trường sản phẩm thực tế sau khi đã qua quá trình kiểm thử và xây dựng. Điều này đảm bảo rằng các thay đổi trong mã nguồn được triển khai nhanh chóng và liên tục.

CI trong kiểm thử phần mềm giúp tăng tốc độ phát triển, giảm rủi ro và đảm bảo tính ổn định và chất lượng của sản phẩm phần mềm. Nó đòi hỏi sự tự động hóa trong quá trình tích hợp, kiểm thử và triển khai, đồng thời cần có sự hợp tác và đồng thuận giữa các thành viên trong nhóm phát triển.

CM (Configuration Management) trong kiểm thử phần mềm là quá trình quản lý, kiểm soát và cấu hình các thành phần phần mềm trong quá trình phát triển và kiểm thử. CM đảm bảo sự quản lý hiệu quả của phiên bản phần mềm, mã nguồn, tài liệu, cấu hình hệ thống và các thành phần liên quan khác. Dưới đây là một số khía cạnh quan trọng của CM trong kiểm thử phần mềm:

1. Quản lý phiên bản: CM giúp quản lý và theo dõi các phiên bản khác nhau của phần mềm và các thành phần liên quan. Nó đảm bảo rằng mọi thay đổi được ghi lại và có sẵn để truy cập, phục hồi hoặc so sánh với phiên bản trước.
2. Quản lý mã nguồn: CM quản lý mã nguồn của phần mềm, bao gồm quản lý các phiên bản, nhánh và hợp nhất mã nguồn. Nó đảm bảo tính nhất quán và kiểm soát các thay đổi trong mã nguồn, đồng thời hỗ trợ quá trình phân tích lỗi và theo dõi các thay đổi liên quan.
3. Quản lý tài liệu: CM đảm bảo quản lý hiệu quả các tài liệu liên quan đến phần mềm, bao gồm tài liệu yêu cầu, tài liệu thiết kế, tài liệu kiểm thử và tài liệu hướng dẫn sử dụng. Điều này đảm bảo rằng mọi thành viên trong nhóm phát triển có thể truy cập và sử dụng tài liệu cần thiết.
4. Quản lý cấu hình hệ thống: CM giúp quản lý cấu hình hệ thống và các thành phần liên quan, bao gồm cấu hình môi trường, cấu hình cơ sở dữ liệu và các cấu hình khác. Điều này đảm bảo tính nhất quán và khả năng tái tạo môi trường kiểm thử.
5. Quản lý thay đổi: CM theo dõi và quản lý các thay đổi trong phần mềm và các thành phần liên quan. Nó cung cấp lịch sử các thay đổi, giúp theo dõi và đánh giá tác động của các thay đổi đối với quá trình kiểm thử và chất lượng phần mềm.

CM trong kiểm thử phần mềm đóng vai trò quan trọng trong đảm bảo tính nhất quán, kiểm soát và kiểm thử hiệu quả của phần mềm. Nó giúp đảm bảo rằng các thành phần phần mềm được quản lý và kiểm soát một cách có hệ thống và đáng tin cậy.