Chapter 4: Software Testing

**Giới thiệu**

Kiểm thử phần mềm bao gồm việc kiểm chứng động (dynamic), đó là một chương trình cung cấp các hành vi dự kiến trên một tập hữu hạn các trường hợp thử nghiệm, phù hợp được lựa chọn từ các miền thực hiện thường là vô hạn.

Các vấn đề quan trọng trong việc mô tả các kiến thức kiểm thử phần mềm (KA):

- Dynamic: Thuật ngữ này có nghĩa là kiểm thử luôn luôn bao gồm thực thi chương trình khi lựa chọn đầu vào (input). Để được chính xác, giá trị đầu vào đơn (alone) không phải lúc nào cũng đủ để xác định 1 bài test, vì một hệ thống không xác định phức tạp có thể phản ứng với các đầu vào cùng với các hành vi khác nhau, tùy thuộc vào trạng thái hệ thống. Tuy nhiên trong KA này, thuật ngữ đầu vào sẽ phải được duy trì, với quy ước bao hàm rằng ý nghĩa của nó cũng bao gồm một trạng thái đầu vào quy định trong những trường hợp quan trọng. Các kỹ thuật tĩnh khác nhau và bổ trợ cho kiểm thử động. Các kỹ thuật tĩnh được bao gồm trong Software Quality KA. Điều đáng chú ý là thuật ngữ không thống nhất giữa các cộng đồng khác nhau và một số sử dụng thuật ngữ “kiểm thử” trong mối liên hệ với các kỹ thuật tĩnh.

- Finite: Ngay cả trong các chương trình đơn giản, rất nhiều trường hợp kiểm thử về mặt lý thuyết được kiểm tra toàn diện có thể yêu cầu nhiều tháng hoặc nhiều năm để thực hiện. Đó là lý do tại sao trong thực tế, một tập đầy đủ các kiểm thử nhìn chung có thể là vô hạn, và kiểm thử được tiến hành trên một tập hợp tất cả các kiểm thử có khả thi, có thể được xác định theo các tiêu chí rủi ro và ưu tiên. Kiểm thử luôn bao hàm sự cân bằng giữa các tài nguyên hạn chế và tiến độ trên một mặt và yêu cầu kiểm thử vốn dĩ không giới hạn trên các mặt khác.

- Selected: Nhiều đề xuất kiểm thử khác nhau về cơ bản là cách một tập kiểm thử được lựa chọn, và các kỹ sư phần mềm phải được nhận thức rằng các tiêu chí lựa chọn khác nhau có thể mang lại mức độ khác nhau về tính hiệu quả. Làm thế nào để xác định các tiêu chí lựa chọn phù hợp nhất trong điều kiện nhất định là một vấn đề phức tạp; trong thực tế, các kỹ thuật phân tích rủi ro và chuyên môn công nghệ phần mềm được áp dụng.

- Expected: Phải có khả năng, mặc dù không phải lúc nào cũng dễ dàng, để quyết định xem các kết quả quan sát của chương trình thử nghiệm được chấp nhận hay không; nếu không, các nỗ lực thử nghiệm là vô ích. Các hành vi quan sát có thể được so sánh với nhu cầu của người sử dụng (thường được gọi là thử nghiệm để xác nhận), chống lại một đặc điểm kỹ thuật (thử nghiệm để xác minh), hoặc, có lẽ, đối với các hành vi được mong đợi từ các yêu cầu bao hàm hoặc mong đợi (xem thử nghiệm Chấp nhận trong các yêu cầu phần mềm KA ).

Trong những năm gần đây, kiểm thử không còn được coi là một hoạt động mà chỉ bắt đầu sau khi giai đoạn coding được hoàn thành với mục đích hạn chế của việc phát hiện lỗi. Kiểm thử phần mềm trở nên phổ biến trong suốt quá trình phát triển và bảo trì. Kế hoạch kiểm thử phần mềm nên bắt đầu với giai đoạn đầu của quy trình yêu cầu phần mềm, các kế hoạch và quy trình thử nghiệm phải được phát triển một cách có hệ thống và liên tục - và có thể tinh chế - như tiến hành phát triển phần mềm. Những hoạt động lập kế hoạch kiểm thử và thử nghiệm thiết kế cung cấp các đầu vào hữu ích cho các nhà thiết kế phần mềm và giúp họ làm nổi bật những khuyết điểm, thiết sót.

Đối với nhiều tổ chức, phương pháp tiếp cận đến chất lượng phần mềm là một trong những sự ngăn chặn: tức là ngăn chặn vấn đề tốt hơn là sửa chữa chúng. Kiểm thử có thể được nhìn thấy, sau đó, được dùng như một phương tiện để cung cấp thông tin về các tính năng và chất lượng các thuộc tính của phần mềm và cũng để xác định lỗi trong những trường hợp ngăn chặn lỗi không có hiệu quả. Sự thật hiển nhiên là phần mềm có thể chứa lỗi, thậm chí sau khi hoàn thành việc kiểm thử bao quát. Các lỗi phần mềm có sau đó sẽ được giải quyết bằng bảo trì sửa chữa. Mục bảo trì phần mềm có trong phần Software Mainteance KA (chương 5).

Trong Kỹ thuật quản lý chất lượng phần mềm, được phân loại thành 2 kỹ thuật đáng chú ý là Tĩnh (không thực thi mã) và động (thực thi mã). KA này tập trung và các kỹ thuật động. Kiểm thử phần mềm cũng có liên quan đến xây dựng phần mềm.

**BREAKDOWN OF TOPICS FOR SOFTWARE TESTING**

**1. Software Testing Fundamentals (kiểm thử phần mềm cơ bản)**

1.1. Thuật ngữ liên quan đến kiểm thử

1.1.1. Định nghĩa về kiểm thử và thuật ngữ liên quan

*1.1.2. Faults vs. Failures (Lỗi và chịu lỗi)*

1.2. Key Issues (Các vấn đề chính)

1.2.1. Tiêu chí lựa chọn kiểm thử/Kiểm tra mức độ đầy đủ tiêu chí

Tiêu chí lựa chọn thử nghiệm có nghĩa là lựa chọn trường hợp kiểm thử hoặc xác định một tập các trường hợp kiểm thử là đủ cho một mục đích cụ thể.

1.2.2. Hiệu quả kiểm thử/Mục tiêu kiểm thử

Hiệu quả kiểm thử được xác định bằng cách phân tích một tập hợp các chương trình thực thi. Lựa chọn kiểm thử được thực hiện có thể được hướng dẫn bởi các mục tiêu khác nhau.

1.2.3. Kiểm thử phát hiện khiếm khuyết

Trong kiểm thử để phát hiện khiếm khuyết, một thử nghiệm thành công là một trong những nguyên nhân dẫn đến hệ thống thất bại.

1.2.4. The Oracle Problem

Oracle là bất kỳ người hoặc máy móc dùng để quyết định xem liệu một chương trình có thực hiện một cách chính xác trong một thử nghiệm nhất định và phù hợp kết quả là đạt hoặc thất bại.

1.2.5. Giới hạn lý thuyết và thực tiễn của kiểm thử

1.2.6. Vấn đề về đường dẫn không khả thi

Đường dẫn không khả thi là các đường dẫn luồng điều khiển không thể được thực thi bởi bất kỳ dữ liệu đầu vào.

1.2.7. Khả năng kiểm thử

Thuật ngữ "khả năng kiểm thử phần mềm" có 2 nghĩa: Một mặt nó đề cập đến sự dễ dàng mà một tiêu chuẩn thử nghiệm nhất định có thể được thỏa mãn; Mặt khác, nó được định nghĩa là khả năng một tập các trường hợp kiểm thử sẽ làm phơi bày lỗi nếu phần mềm bị lỗi.

1.3. Mối quan hệ của kiểm thử với các hoạt động khác

- Kiểm thử vs kỹ thuật quản lý chất lượng phần mềm tĩnh

- Kiểm thử vs Bằng chứng về tính đúng đắn và xác minh hình thức

- Kiểm thử vs Gỡ lỗi

- Kiểm thử vs Xây dựng chương trình

2. Test Levels

Kiểm thử phần mềm thường được thực hiện ở các cấp độ khác nhau trong suốt quá trình phát triển và bảo trì. Mức có thể được phân biệt dựa trên các đối tượng thử nghiệm, được gọi là các target, hoặc về mục đích, được gọi là các objective (từ cấp thử nghiệm).

2.1. Mục tiêu của kiểm thử (Target of the Test)

Mục tiêu của thử nghiệm có thể khác nhau: một mô-đun duy nhất, một nhóm các mô-đun như (liên quan theo mục đích, sử dụng, hành vi, hoặc cơ cấu), hay toàn hệ thống. Ba giai đoạn thử nghiệm có thể được phân biệt: đơn vị, tích hợp và hệ thống.

2.1.1. Kiểm thử đơn vị

Đơn vị kiểm thử xác minh các chức năng một cách riêng biệt của các yếu tố phần mềm mà có thể kiểm chứng một cách riêng biệt. Tùy thuộc vào hoàn cảnh, đây có thể là các chương trình con riêng lẻ hoặc một bộ phận lớn các đơn vị có liên kết chặt chẽ.

2.1.2. Kiểm thử tích hợp

Kiểm thử tích hợp là quá trình xác minh sự tương tác giữa các thành phần phần mềm. Thường được sử dụng với các phần mềm có cấu trúc phân cấp. Kiểm thử tích hợp thường diễn ra ở từng giai đoạn của sự phát triển.

2.1.3. Kiểm thử hệ thống

Kiểm thử hệ thống là có liên quan với kiểm thử hành vi của toàn bộ hệ thống. Kiểm thử đơn vị và kiểm thử tích hợp thường đã xác định nhiều lỗi phần mềm. Kiểm thử hệ thống thường được coi là thích hợp cho việc đánh giá phi chức năng hệ thống yêu cầu - chẳng hạn như bảo mật, tốc độ, độ chính xác và đột in cậy. Giao diện bên ngoài cho các ứng dụng khác, tiện ích, các thiết bị phần cứng, hoặc các môi trường hoạt động cũng thường được đánh giá ở mức này.

2.2. Mục đích của kiểm thử

Kiểm thử được tiến hành trong bối cảnh mục tiêu cụ thể, trong đó được ghi nhận nhiều hơn hoặc ít hơn một cách rõ ràng và có độ chính xác khác nhau. Trong đó nêu rõ mục đích của kiểm thử một cách chính xác, hỗ trợ về mặt định lượng đo lường và kiểm soát trong quá trình kiểm thử.

Kiểm thử có thể được dùng để xác minh các tính chất khác nhau. Trường hợp kiểm thử có thể được thiết kế để kiểm tra các thông số kỹ thuật chức năng được thực hiện một cách chính xác, hiệu suất, độ tin cậy, khả dụng...

2.2.1. Nghiệm thu/Năng lực kiểm thử

Nghiệm thu/Năng lực kiểm thử xác định một hệ thống đáp ứng tiêu chuẩn nghiệm thu của nó, thường là bằng cách kiểm tra các hành vi hệ thống mong muốn đối với các yêu cầu của khách hàng.

2.2.2. Cài đặt kiểm thử

Thông thường, sau khi hoàn thành hệ thống và chấp nhận kiểm thử, các phần mềm được xác minh sau khi cài đặt trong môi trường mục tiêu. Cài đặt kiểm thử có thể được xem như là hệ thống thử nghiệm được tiến hành trong môi trường hoạt động cùa cấu hình phần cứng và hoạt động hạn chế khác.

2.2.3. Kiểm thử Alpha và Beta

Trước khi phần mềm được phát hành, nó đôi khi được trao cho một nhóm nhỏ, nhóm người dùng tiềm năng được chọn để sử dụng (alpha test) và một nhóm lớn hơn của bên đại diện (beta test).

2.2.4. Độ tin cậy và đánh giá

Kiểm thử cải thiện độ tin cậy bằng cách xác định và sửa lỗi. Ngoài ra, các biện pháp thống kế về độ tin cậy có thể được bắt nguồn bằng cách tạo ra một cách ngẫu nhiên trường hợp kiểm tra theo hoạt động của phần mềm.

2.2.5. Kiểm thử hồi quy

Kiểm thử hồi quy là kiểm thử lại chọn lọc của một hệ thống hay thành phần để xác minh rằng những sửa đổi không gây ra hiệu ứng ngoài ý muốn và hệ thống hay thành phần vẫn tuân thủ các yêu cầu của quy định.

2.2.6. Kiểm thử hiệu suất

Kiểm thử hiệu suất xác nhận rằng các phần mềm đáp ứng được yêu cầu thực hiện quy định và đánh giá các đặc tính hiệu suất - VD, năng lực và thời gian đáp ứng.

2.2.7. Kiểm thử bảo mật

Kiểm thử bảo mật tập trung vào việc xác minh rằng phần mềm được bảo vệ khỏi các cuộc tấn công từ bên ngoài. Đặc biệt, kiểm thử bảo mật xác minh tính bảo mật, tính toàn vẹn và tính sẵn sàng của hệ thống và dữ liệu của nó.

2.2.8. Kiểm thử Stress

Kiểm thử stress thử nghiệm các bài tập phần mềm thiết kế tải trọng tối đa, hoặc vượt ra ngoài nó, với mục tiêu xác định các giới hạn hành vi.

2.2.9. Kiểm thử Back-to-Back

IEEE/ISO/IEC Standard 24765 định nghĩa kiểm thử back-to-back là "kiểm thử, trong đó hai hoặc nhiều hơn các biến thể của một chương trình được thực hiện với các đầu vào tương tự, các kết quả được so sánh, và các lỗi được phân tích trong trường hợp có sự khác biệt".

2.2.10. Kiểm thử phục hồi

Kiểm thử phục hồi nhằm mục đích kiểm tra khả năng khởi động lại của phần mềm sau khi hệ thống bị treo hoặc crash.

2.2.11. Kiểm thử giao diện

Các nhược điểm giao diện rất phổ biến trong các hệ thống phức tạp. Kiểm thử giao diện nhằm xác minh các thành phần giao diện chính xác để cung cấp những trao đổi chính xác của dữ liệu và kiểm soát thông tin. Thông thường các trường hợp thử nghiệm được tạo ra từ các đặc tả giao diện. Mục tiêu cụ thể của thử nghiệm giao diện là để mô phỏng việc sử dụng các API của các ứng dụng của người dùng cuối. Điều này liên quan đến việc tạo ra các thông số của các cuộc gọi API, thiết lập các điều kiện môi trường bên ngoài, và định nghĩa của dữ liệu nội bộ có ảnh hưởng đến các API.

2.2.12. Kiểm thử cấu hình

Trong trường hợp phần mềm được xây dựng để phục vụ người dùng khác nhau, kiểm thử cấu hình dùng để kiểm chứng các phần mềm theo quy định cấu hình khác nhau.

2.2.13. Kiểm thử tính khả dụng và tương tác người - máy

Nhiệm vụ chính của phần này là đánh giá như thế nào là dễ dàng cho người dùng cuối để tìm hiểu và sử dụng phần mềm. Nói chung, nó có thể liên quan đến việc thử nghiệm các chức năng phần mềm hỗ trợ việc sử dụng, tài liệu hướng dẫn nhằm hỗ trợ người sử dụng, và khả năng của hệ thống để phục hồi từ lỗi người sử dụng

**5. Test process**

Khái niệm về kiểm thử, chiến lược, kỹ thuật và đo lường cần phải được tích hợp vào một quá trình xác định và kiểm soát. Quá trình kiểm thử hỗ trợ các hoạt động kiểm tra và cung cấp hướng dẫn cho các tester và đội tester, từ việc lập kế hoạch để đánh giá đầu ra, theo cách như vậy để đảm bảo cung cấp rằng các mục tiêu kiểm thử sẽ được đáp ứng trong một chi phí hiệu quả kịp thời.

**5.1. Practical Considerations**

**5.1.1. Attitudes / Egoless Programming**

Một yếu tố quan trọng của kiểm thử thành công là một thái độ hợp tác hướng tới thử nghiệm và hoạt động đảm bảo chất lượng. Những người quản lý có vai trò quan trọng trong việc thúc đẩy một tiếp nhận thuận lợi nói chung để hướng tới phát hiện lỗi và sửa chữa trong quá trình phát triển và bảo trì phần mềm.

Ví dụ: bằng cách vượt qua những suy nghĩ của quyền sở hữu cá nhân giữa các lập trình và bằng cách thúc đẩy một môi trường cộng tác với đội ngũ chịu trách nhiệm về bất thường trong các mã lập trình.

**5.1.2. Test Guides**

Các giai đoạn kiểm thử có thể được hướng dẫn bởi các mục tiêu khác nhau.

Ví dụ: Kiểm thử dựa trên rủi ro sử dụng các rủi ro sản phẩm ưu tiên và tập trung các chiến lược kiểm thử, và kiểm thử dựa trên kịch bản xác định các trường hợp kiểm thử dựa trên kịch bản phần mềm quy định.

**5.1.3. Test Process Management**

Các hoạt động kiểm thử được tiến hành ở các cấp độ khác nhau (xem thêm ở level 2: Test levels) phải được tổ chức lại với nhau, cùng với con người, công cụ, chính sách và các biện pháp thành quá trình xác định, đó là một phần không thể thiếu của vòng đời.

**5.1.4. Test Documentation and Work Products**

Tài liệu kiểm thử là một phần không thể thiếu khi chính thức quá trình kiểm thử. Tài liệu kiểm thử có thể bao gồm các kế hoạch kiểm tra, đặc điểm kỹ thuật thiết kế kiểm tra, đặc điểm kỹ thuật quy trình kiểm tra, kiểm tra các trường hợp đặc điểm kỹ thuật, kiểm tra log và báo cáo sự việc kiểm tra.

Các phần mềm được kiểm thử được ghi nhận như một tác phẩm kiểm thử. Tài liệu kiểm thử nên được tạo ra và liên tục cập nhật tới cùng một mức độ chất lượng như các loại tài liệu khác trong công nghệ phần mềm. Tài liệu kiểm thử cũng nên chịu sự giám sát của quản lý cấu hình phần mềm. Hơn nữa, tài liệu kiểm thử bao gồm các sản phẩm công việc mà có thể cung cấp các tài liệu hướng dẫn cho người sử dụng và đào tạo.

**5.1.5. Test-Driven Development**

Phát triển hướng Test (TDD) ban đầu là một trong những thực hành cốt lõi XP (extreme programming) và bao gồm các kiểm thử đơn vị bằng văn bản trước khi các mã lập trình được kiểm tra (xem phương pháp Agile trong mô hình công nghệ phần mềm và phương pháp KA).

Bằng cách này, TDD phát triển các trường hợp kiểm thử như là một thay thế cho tài liệu đặc tả yêu cầu phần mềm hơn là một kiểm tra độc lập mà phần mềm đã thực hiện đúng các yêu cầu. Thay vì một chiến lược kiểm thử, TDD là một thực tế đòi hỏi các nhà phát triển phần mềm để xác định và duy trì kiểm tra đơn vị (unit tests). Điều này có thể có một động thái tích cực vào việc xây dựng nhu cầu người dùng và các đặc tả yêu cầu phần mềm.

**5.1.6. Internal vs. Independent Test Team**

Quá trình kiểm thử cũng có thể liên quan đến việc tổ chức các nhóm kiểm thử. Các nhóm kiểm thử có thể bao gồm các thành viên nội bộ (có nghĩa là, trong nhóm dự án, tham gia hay không trong xây dựng phần mềm), các thành viên bên ngoài (với hy vọng mang lại một quan điểm độc lập, không thiên vị) , hoặc của cả hai thành viên nội bộ hay bên ngoài. Cân nhắc về chi phí, thời gian, mức độ trưởng thành của tổ chức có liên quan và quan trọng của các ứng dụng có thể hướng dẫn các quyết định.

**5.1.7. Cost/Effort Estimation and Test Process Measures**

Một số các biện pháp liên quan đến các nguồn tài nguyên dành cho việc kiểm tra, cũng như hiệu quả tìm kiếm lỗi tương ứng của các giai đoạn kiểm thử khác nhau, được sử dụng bởi các nhà quản lý để kiểm soát và cải thiện quá trình kiểm thử. Những biện pháp kiểm tra có thể bao gồm các khía cạnh như số trường hợp kiểm thử được chỉ định, số trường hợp kiểm tra được thực hiện, số lượng trường hợp kiểm thử đạt yêu cầu, số lượng trường hợp kiểm thử không đạt yêu cầu trong số những cái khác.

Thẩm định báo cáo giai đoạn kiểm thử có thể được kết hợp với phân tích nguyên nhân gốc để đánh giá hiệu quả quá trình kiểm thử trong việc tìm lỗi càng sớm càng tốt. Việc đánh giá đó có thể kết hợp với việc phân tích rủi ro. Hơn nữa, các tài nguyên dành cho việc kiểm thử phải phù hợp với việc sử dụng/quan trọng của các ứng dụng: các kỹ thuật khác nhau có các chi phí khác nhau và mang lại mức độ khác nhau của độ tin tưởng trong mức độ tin cậy của sản phẩm.

**5.1.8. Termination**

Một quyết định cần phải đưa ra là việc kiểm thử như thế nào là đủ và khi nào giai đoạn kiểm thử được chấm dứt. Thông qua đo đạc, xác định các chức năng đảm bảo, cũng như ước tính mật độ lỗi hoặc độ tin cậy của hoạt động.

Các quyết định cũng bao gồm việc xem xét về chi phí và rủi ro phát sinh bởi sự thất bại còn có thể xảy ra, như trái ngược với chi phí phát sinh bằng cách tiếp tục kiểm thử (xem tiêu chuẩn lựa chọn kiểm thử / tiêu chuẩn thử nghiệm Adequacy ở mục 1.2)

**5.1.9. Test Reuse and Test Patterns**

Để thực hiện kiểm thử, bảo trì một cách có tổ chức và hiệu quả chi phí, các kiểm thử từng phần của phần mềm nên được tái sử dụng một cách hệ thống. Một kho lưu trữ kiểm thử nên chịu sự kiểm soát của quản lý cấu hình phần mềm để mà thay đổi yêu cầu phần mềm hoặc thiết kế có thể được phản ảnh trong thay đổi cho quản lý các kiểm thử.

Các giải pháp kiểm thử được áp dụng để kiểm thử một số loại ứng dụng trong những hoàn cảnh nhất định, với những động cơ đằng sau các quyết định đưa ra, tạo thành một mô hình kiểm thử mà chính nó có thể được ghi lại để tái sử dụng sau này trong các dự án tương tự.

**5.2. Test Activities**

Như thể hiện trong các mô tả dưới đây, quản lý thành công của các hoạt động kiểm thử phụ thuộc mạnh mẽ vào quá trình quản lý cấu hình phần mềm (xem quản lý cấu hình phần mềm KA)

**5.2.1. Planning**

Giống như tất cả các khía cạnh khác nhau của quản lý dự án, các hoạt động kiểm thử phải được quy hoạch. Khía cạnh quan trọng của việc lập kế hoạch kiểm thử bao gồm sự phối hợp của các nhân viên, tính sẵn có của các cơ sở và trang thiết bị kiểm thử, tạo ra và bảo trì tất cả các tài liệu liên quan đến kiểm thử, và lên kế hoạch cho những kết quả không mong muốn có thể. Nếu có nhiều hơn cơ sở của các phần mềm đang duy trì, sau đó xem xét một kế hoạch chính là thời gian và nỗ lực cần thiết để đảm bảo rằng các môi trường kiểm thử được thiết lập để cấu hình đúng.

**5.2.2. Test-Case Generation**

Việc tạo ra các trường hợp kiểm thử dựa trên mức độ kiểm thử được thực hiện và kỹ thuật kiểm thử cụ thể. Các trường hợp kiểm thử nên được dưới sự kiểm soát của quản lý cấu hình phần mềm và bao gồm các kết quả dự kiến cho mỗi kiểm thử.

**5.2.3. Test Environment Development**

Môi trường được sử dụng để kiểm thử cần có sự tương thích với các công cụ phần mềm được đề nghị. Nó tạo điều kiện phát triển và kiểm soát các trường hợp kiểm thử, cũng giống như ghi lại và phục hồi các kết quả mong đợi, kịch bản và các tài liệu kiểm thử khác.

**5.2.4. Execution**

Thực hiện các kiểm thử nên thể hiện một nguyên tắc cơ bản của thí nghiệm khoa học: tất cả mọi thứ được thực hiện trong quá trình kiểm thử nên được thực hiện và ghi chép rõ ràng đủ để người khác có thể nhân rộng kết quả. Do đó, việc kiểm thử nên được thực hiện theo thủ tục bằng cách sử dụng một phiên bản xác định rõ ràng của phần mềm bên dưới các kiểm thử.

**5.2.5. Test Results Evaluation**

Các kết quả của kiểm thử cần được đánh giá để xác định việc kiểm thử có thành công hay không. Trong hầu hết các trường hợp, thành công nghĩa là phần mềm thực hiện như mong đợi và không có các kết quả bất ngờ nào. Không phải tất cả các kết quả bất ngờ là nhất thiết phải lỗi.

Trước khi một lỗi có thể được xóa đi, một phân tích và nỗ lực gỡ lỗi là cần thiết để cô lập, xác định và mô tả nó. Khi kết quả kiểm thử là đặc biệt quan trọng, một hội đồng xét duyệt chính thức có thể được triệu tập để đánh giá chúng.

**5.2.6. Problem Reporting / Test Log**

Hoạt động kiểm thử có thể sử dụng các nhật ký kiểm thử để xác định một kiểm thử đã được tiến hành, ai là người thực hiện các kiểm tra, những cấu hình phần mềm nào được sử dụng và các thông tin nhận dạng khác có liên quan. Kết quả kiểm thử đột xuất hoặc không chính xác có thể được ghi lại trong một hệ thống báo cáo sự cố, các dữ liệu mà tạo cơ sở cho việc gỡ lỗi sau đó và sửa chữa các vấn đề đã được quan sát như những thất bại trong quá trình kiểm thử. Ngoài ra, bất thường không được phân loại là lỗi có thể được ghi lại trong trường hợp chúng lần lượt xuất hiện là nghiêm trọng hơn nhiều so với suy nghĩ đầu tiên. Báo cáo kiểm thử cũng là đầu vào để thay đổi quy trình quản lý theo yêu cầu (xem điều khiển cấu hình phần mềm trong quản lý cấu hình phần mềm KA)

**5.2.7. Defect Tracking**

Các khiếm khuyết có thể được theo dõi và phân tích để xác định khi chúng được đưa vào phần mềm, tại sao chúng được tạo ra, (ví dụ, yêu cầu kém được xác định, khai báo biến không đúng, rò rỉ bộ nhớ, lập trình lỗi cú pháp), và khi họ có thể quan sát thấy lần đầu tiên trong phần mềm. Khiếm khuyết về thông tin được sử dụng để xác định những khía cạnh của kiểm thử phần mềm và các quá trình khác cần cải thiện và hiệu quả của phương pháp tiếp cận trước đây diễn ra như thế nào.

**6. Software Testing Tools**

**6.1. Testing Tool Support**

Kiểm thử đòi hỏi nhiều công việc cần thực hiện, chạy nhiều chương trình và xử lý một lượng lớn thông tin. Các công cụ thích hợp có thể làm giảm bớt gánh nặng của việc lặp lại, hoạt động tẻ nhạt và làm cho việc kiểm thử ít dễ bị lỗi. Công cụ tinh vi có thể hỗ trợ thiết kế kiểm thử và tạo ra các trường hợp kiểm thử, làm cho nó hiệu quả hơn.

**6.1.1. Selecting Tools**

Hướng dẫn cho các nhà quản lý và các người kiểm thử về cách chọn công cụ kiểm tra rằng sẽ hữu ích nhất để tổ chức và các quy trình là một chủ đề rất quan trọng, như là một công cụ tạo ảnh hưởng lớn để hiệu quả kiểm thử và thật sự hiệu quả. Việc lựa chọn công cụ phụ thuộc vào các dấu hiệu khác nhau, chẳng hạn như lựa chọn phát triển, mục tiêu đánh giá, các cơ sở thực hiện,…Nói chung, có thể không có một công cụ duy nhất để đáp ứng các nhu cầu cụ thể, do đó một bộ công cụ có thể là một lựa chọn thích hợp.

**6.2. Categories of Tools**

Chúng ta phân loại các công cụ có sẵn theo chức năng của chúng:

• Test harnesses (drivers, stubs): cung cấp một môi trường điều khiển, trong đó việc kiểm thử có thể được thực hiện và kết quả đầu ra có thể được ghi lại. Để thực hiện các phần của một chương trình, các drivers và stubs được cung cấp để mô phỏng cuộc gọi và gọi là mô đun, một cách tương ứng.

• Test generators: cung cấp các hỗ trợ trong trường hợp kiểm thử thế hệ. Các thế hệ có thể là ngẫu nhiên, dựa trên tìm đường đi, dựa trên mô hình, hoặc hỗn hợp của chúng.

• Capture/replay tools: tự động thực thi lại, hoặc phát lại kiểm thử thực hiện trước đó đã ghi lại các đầu vào và đầu ra (ví dụ: màn hình).

• Oracle/file comparators/assertion checking tools: hỗ trợ trog việc quyết định một kết quả kiểm thử có thành công hay không.

• Coverage analyzers and instrumenters: làm việc cùng nhau, phân tích các truy cập theo vùng và có bao nhiêu thực thể của đồ thị lưu lượng chương trình đã được thực hiện trong tất cả những yêu cầu của các tiêu chuẩn vùng đã được lựa chọn kiểm thử. Các phân tích có thể được thực hiện nhờ vào các công cụ chương trình.

• Tracers: ghi lại lịch sử của đường dẫn thực thi một chương trình.

• Regression testing tools: hỗ trợ các thực thi lại của một bộ kiểm thử sau khi một phần của phần mềm đã được sửa đổi. Công cụ này cũng giúp đỡ để chọn một tập hợp kiểm thử theo các thay đổi được thực hiện.

• Reliability evaluation  tools: hỗ trợ phân tích kết quả kiểm thử và hiển thị đồ họa để đánh giá độ tin cậy liên quan đến các biện pháp theo mô hình lựa chọn.