**Chapter 4: Software Testing**

Nhóm 4: Phạm Văn Thiện

Ngô Thế Hải Anh

Nguyễn Anh Tuấn

**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.

**Chi tiết chủ đề kiểm thử phần mềm**

1. **Kiểm thử phần mềm cơ bản**
   1. ***Thuật ngữ liên quan đến kiểm thử***
      1. Định nghĩa về kiểm thử và thuật ngữ liên quan
      2. Lỗi và chịu lỗi
   2. ***Key Issues (Các vấn đề chính)***
      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. 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. Kiểm thử phát hiện khiếm khuyết

Phát hiện lỗi hoặc những khiếm khuyết của phần mềm để thấy được ứng xử của nó có chính xác hoặc phù hợp với tài liệu đặc tả của nó hay không. Về mặt lý thuyết, chúng ta phải kiểm thử hệ thống một cách cặn kẽ thì mới khẳng định được chương trình không còn khiếm khuyết. Tuy nhiên, trong thực tế không thể kiểm thử một cách cặn kẽ được.

* + 1. **Các vấn đề về Oracle**

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 (các nguyên tắc hay cơ chế phát hiện vấn đề). Kiểm thử không thể xác định hoàn toàn được tất cả các lỗi bên trong phần mềm.Thay vào đó, nó so sánh trạng thái và hành vi của sản phẩm với các oracle. Các oracle này có thể bao gồm (nhưng không giới hạn ở) các đặc tả phần mềm, hợp đồng, sản phẩm tương đương, các phiên bản trước của cùng một sản phẩm, phù hợp với mục đích dự kiến nhằm đáp ứng sự kỳ vọng của người dùng, khách hàng, quy định của pháp luật hiện hành và các tiêu chuẩn liên quan khác.

* + 1. **Giới hạn lý thuyết và thực tiễn của kiểm thử**
    2. **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. **b**

Khả năng kiểm thử là mức độ mà một thành phần phần mềm (hệ thống phần mềm, module, tài liệu thiết kế) hỗ trợ kiểm thử trong một bối cảnh kiểm thử nhất định. Nếu khả năng kiểm thử của thành phần phần mềm cao, thì việc tìm kiếm lỗi trong hệ thống (nếu có) bằng việc kiểm thử được dễ dàng hơn.

* 1. **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

1. **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).

* 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.

* + 1. **Kiểm thử đơn vị**

Unit testing đề cập đến các kiểm thử để chứng thực (xác minh - verify) chức năng của một phần riêng biệt của code, thường ở mức hàm (function level). Trong một môi trường hướng đối tượng (object-oriented environment), kiểm thử đơn vị thường được sử dụng ở mức lớp (class) và kiểm thử các đơn vị nhỏ nhất bao gồm các hàm constructor và destructor.

Loại kiểm thử này thường được viết bởi các DEV như công việc của họ trong việc code (loại test white-box), để bảo đảm rằng từng hàm riêng biệt hoạt động đúng theo mong muốn. Một hàm có thể có nhiều kiểm thử, để bắt được các trường hợp hoặc các nhánh trong code. Unit testing một mình không thể bảo đảm chức năng của một bộ phận của phần mềm mà là sử dụng để bảo đảm rằng các khối kiến trúc của phần mềm làm việc độc lập với nhau.

Unit testing (kiểm thử đơn vị) cũng được gọi là component testing (kiểm thử thành phần).

* + 1. **Kiểm thử tích hợp**

Integration testing là một loại kiểm thử phần mềm mà tìm kiếm để kiểm tra các giao diện giữa các thành phần dựa vào thiết kế của phần mềm. Các thành phần phần mềm có thể được tích hợp lại với nhau theo cách lặp đi lặp lại (từng phần nhỏ ghép lại với nhau, rồi ghép tiếp phần nhỏ khác vào nữa, hành động này lặp lại cho đến khi kết hợp toàn bộ phần mềm) hoặc tất cả các thành phần cùng tích hợp một lần (gọi là “big bang”). Thông thường trước đây được xem là một cách làm tốt hơn từ khi nó cho phép các vấn đề về giao diện được xác định vị trí nhanh hơn và cố định. Integration testing làm việc để tìm ra lỗi (defect) trong các giao diện và giao tiếp giữa các thành phần (mô-đun). Các nhóm thành phần phần mềm đã được kiểm thử lớn dần từng bước tương ứng với các yếu tố của thiết kế kiến trúc đã được tích hợp và kiểm thử cho đến khi phần mềm hoạt động như một hệ thống.

* + 1. **Kiểm thử hệ thống**

System testing kiểm thử một hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đáp ứng được yêu cầu.

Kiểm thử tích hợp hệ thống chứng thực rằng hệ thống đã được tích hợp với các hệ thống bên ngoài hoặc hệ thống thứ ba đã được xác định trong các yêu cầu hệ thống.

* 1. ***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...

* + 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.

* + 1. **Kiểm thử cài đặt**

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.

* + 1. **Kiểm thử Alpha và Beta**

- Alpha testing là việc kiểm thử hoạt động chức năng thực tế hoặc giả lập do người dùng/khách hàng tiềm năng hoặc một nhóm test độc lập thực hiện tại nơi sản xuất phần mềm. Alpha testing thường dùng cho phần mềm đóng gói sẵn để bán (ví dụ như MS office, window, chương trình diệt virus) là một hình thức kiểm thử chấp nhận nội bộ, trước khi phần mềm được tiến hành kiểm thử beta.

- Beta testing được thực hiện sau alpha testing. Các phiên bản của phần mềm - được biết như là các phiên bản beta – chúng được phát hành tới một số lượng giới hạn khán giả bên ngoài nhóm sản xuất phần mềm. Sản phẩm được phát hành đến một số nhóm người để test nhiều hơn nữa có thể chắc chắn rằng sản phẩm có một số bug. Thỉnh thoảng, các phiên bản beta được phát hành rộng rãi để tăng phạm vi phản hồi từ một lượng người sử dụng tương lai lớn nhất.

* + 1. **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.

* + 1. **Kiểm thử hồi quy**

Tập trung vào việc tìm kiếm lỗi sau khi xảy ra việc thay đổi code. Đặc biệt, nó tìm kiếm theo cách hồi qui (đệ qui) hoặc kiểm tra các bug cũ có bị lại hay không. Hồi qui như vậy xảy bất cứ khi nào mà chức năng phần mềm trước đây làm việc đúng đã ngưng làm việc theo mong đợi. Điển hình, hồi qui xảy ra như là một kết quả không mong muốn của việc thay đổi chương trình, khi phần code của phần mềm mới được phát triển xung độ với code cũ đang có. Phương pháp thông thường của kiểm tra hồi quy là bao gồm việc chạy lại các kiểm thử trước đây và kiểm tra xem có lỗi đã được fixed trước đây bị lỗi lại (bị lại các lỗi cũ đã fixed rồi). Độ sâu của việc kiểm thử phụ thuộc vào các giai đoạn trong quá trình phát hành và rủi ro của các tính năng được thêm vào. Chúng có thể được hoàn thành – vì việc thay đổi đã thêm vào sau bản phát hành hoặc coi nó là mạo hiểm, rất hời hợt, bao gồm các kiểm thử trường hợp đúng (positive) trên từng chức năng – nếu các thay đổi được thêm vào trước khi phát hành hoặc coi nó ít rủi ro.

* + 1. **Kiểm thử hiệu năng**

Kiểm thử hiệu năng được thực hiện để xác định hệ thống hoặc hệ thống con thực hiện một khối lượng công việc cụ thể nhanh thế nào. Nó cũng có thể dùng để xác nhận và xác minh những thuộc tính chất lượng khác của hệ thống như là khả năng mở rộng, độ tin cậy và sử dụng tài nguyên. Load testing là khái niệm chủ yếu của việc kiểm thử mà có thể tiếp tục hoạt động ở một mức tải cụ thể, cho dù đó là một lượng lớn dữ liệu hoặc lượng lớn người sử dụng. Cái này gọi chung là khả năng mở rộng của phần mềm. Hoạt động load testing liên quan khi thực hiện hoạt động phi chức năng thường được gọi là kiểm thử sức chịu đựng (độ bền).

* + 1. **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.

* + 1. **Kiểm thử Stress**

Stress testing là một hình thức kiểm thử được sử dụng để xác định tính ổn định của một hệ thống hoặc một thực thể được đưa ra. Nó liên quan đến những kiểm thử vượt quá khả năng bình thường của hệ thống, thường để xác định điểm phá vỡ, để quan sát các kết quả. Stress testing có thể có nghĩa cụ thể hơn trong một số ngành công nghiệp nhất định, như là kiểm thử tính mỏi của vật liệu.

* + 1. **Kiểm thử Back-to-Back (so sánh)**

Khi triển khai nhiều phiên bản phần mềm từ cùng một đặc tả. Kiểm thử hộp đen cho các sản phẩm này được thực hiện với cùng ca kiểm thử và cùng các dữ liệu vào. So sánh các kết quả thu được, nếu có khác biệt nghĩa là có sai trong một sản phẩm nào đó.

* + 1. **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.

* + 1. **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.

* + 1. **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.

* + 1. **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

1. **Kỹ thuật kiểm thử**

Một trong những trọng tâm của việc kiểm thử đó là phát hiện nhiều lỗi nhất có thể. Nhiều kỹ thuật đã được phát triển để làm việc này. Những kỹ thuật này để thử phá một chương trình trình một cách hệ thống bằng việc chỉ ra đầu và mà sẽ tạo ra các trạng thái của chương trình. Ví dụ, xem xét các lớn con của miền đầu vào, các kịch bản, trạng thái và luồng dữ liệu.

Việc phân loại các kỹ thuật kiểm thử được trình bày ở đây là dựa trên cách kiểm thử được thực hiện: từ trực giác và kinh nghiệm của kỹ sư phần mềm, các chi tiết kỹ thuật, cấu trúc mã nguồn, lỗi thực hoặc lỗi tưởng tượng được tìm ra, việc sự dụng được dự đoán trước, mô hình hoặc bản chất của ứng dụng. Một loại tiến hành với 2 hoặc nhiều kỹ thuật kết hợp lại.

Một vài kỹ thuật này được xếp loại như white-box hoặc glass-box nếu kiểm thử dựa trên thông tin về cách phần mềm được thiết kế hoặc viết mã nguồn. Hoặc được xem như là black-box nếu nó chỉ dựa trên trạng thái đầu vào đầu ra của phần mềm. Dưới đây bao gồm vài kỹ thuật thông dụng, nhưng một vài người kiểm thử phụ thuộc vào một số kỹ thuật hơn các kỹ thuật khác

* 1. **Dựa trên trực giác và kinh nghiệm của kỹ sư phần mềm**
     1. **Tùy biến**

Có lẽ kỹ thuật được thực hiện nhiều nhất là kiểm thử tùy biến: các kiểm thử có được dựa trên kỹ năng, trực quan và kinh nghiệm của kỹ sư phần mềm với cá chương trình quen thuộc. Kiểm thử tùy biến có thể hữu ích cho việc định danh các trường hợp kiểm thử mà không dễ thực hiện bằng các kỹ thuật bình thường.

* + 1. **Kiểm thử khám phá**

Kiểm thử theo kiểu khám –phá được định nghĩa là tiến hành đồng thời việc học tập, thiết kế và thực hiện kiểm thử, nó không được thực hiện theo kế hoạch kiểm thử đã được lên trước và linh động về việc thiết kế, thực hiện và sửa đổi. Hiệu quả của kiểm thử khám phá phụ thuộc và kinh nghiệm của kỹ sư phần mềm, điều này có được từ nhiều nguồn như quan sán hành vi của sản phẩm trong quá trình kiểm thử, sự quen thuộc với các ứng dụng, môi trường, quá trình lỗi, các dạng lôi có thể có, nguy cơ khi tích hợp với các sản phẩm riêng biệt….

* 1. **Kỹ thuật dựa trên miền đầu vào**
     1. **Phân vùng tương đương**

Phân vùng tương đương liên quan đến việc phân chia miền đầu vào thành các vùng dựa trên tiêu chuẩn hoặc quan hệ. Tiêu chuẩn hoặc quan hệ này có thể là những kết quả tính toán khác nhau, quan hệ dựa trên luồng điều khiển hoặc luồng dữ liệu, hoặc một sự phân biệt được tạo ra giữa các đầu vào hợp được chấp nhận và đầu vào không hợp lệ sẽ bị từ chối và đưa ra lỗi.

* + 1. **Kiểm thử theo cặp**

Các trường hợp kiểm thử có được bằng việc kết hợp các giá trị tiêu biểu cho mỗi cặp của một tập hợp biến đầu vào thay vì xét đến toàn bộ các kết hợp có thế có. Kiểm thử theo cặp thuộc về kiểm thử tổ hợp, nhìn chung bao gồm mức tổ hợp cao hơn so với cặp: những kỹ thuật này được xem như t-wise, cái mà toàn bộ tổ hợp đầu vào t được xem xét.

* + 1. **Phân tích giá trị biên**

Các trường hợp kiểm thử được chọn ra sẽ nằm trên hoặc gần với điên của miền đầu vào, nơi tỉ lệ lỗi thường tập chung cao nhất. Trường hợp ngoại lệ của kỹ thuật này là kiểm thử sự bền vững, nơi mà trường hợp kiểm thử được chọn bên ngoài miền biến đầu vào để kiểm thử sự bền vững của sản phầm trong việc xử lý đầu vào lỗi.

* + 1. **Kiểm thử ngẫu nhiên**

Các kiểm thử được thực hiện đơn thuẩn một cách ngẫu nhiên. Dạng kiểm thử này đặt dưới đầu của miền đầu vào bởi vì miền đầu vào phải được biến để để có thể lấy ra được những điểm ngẫu nhiên. Kiểm thử ngẫu nhiên cung cấp một tiếp cận đơn giản để kiểm thử tự động, gần đây các dạng nâng cao của kiểm thử ngẫu nhiên được đưa ra, mẫu đầu vào ngẫu nhiên được hướng bởi tiêu chuẩn lựa chọn đầu vào khác. Kiểm thửu Fuzz hay Fuzzing là dạng đặc biệt của kiểm thử ngẫu nhiên tập chung vào việc phá vỡ phần mềm, nó chủ yếu được dùng để kiểm thử an ninh của phần mềm.

* 1. **Kỹ thuật dựa trên mã nguồn**
     1. **Tiêu chuẩn dựa trên luồng điều khiển**

Tiêu chuẩn phủ dựa trên luồng điều khiển được tập chung vào việc phủ toàn bộ các câu lệnh, khối lệnh hoặc kết hợp đặc biệt của các câu lệnh trong chương trình. Điểm mạnh nhất của tiêu chuẩn này là kiểm thử đường đi, cacsi mà tập chung vào để thực thi toàn bộ đường đi lường điều khiển từ khi vào đến khi kết thúc trong biểu đồ luồng điều khiển của chương trình. Vì kiểm thử vét kiệt đường đi là không khả thi do vòng lặp, tiêu chuẩn ít chính xác tập chung vào đọ phủ của đường đi các mà giới hạn các bước lặp như phủ câu lệnh, phủ nhánh và kiểm thử quyết định. Sự đầy đủ của các kiểm thử như thế được đo đạc theo phần trăm, ví dụ, khi tất cả các nhánh được thự thi ít nhất một lần khi kiểm thử thì độ phủ nhánh là 100%.

* + 1. **Tiêu chuẩn dựa trên luồng dữ liệu**

Trong kiểm thử dựa trên luồng dữ liệu, biểu đồ luồng điều khiển được diễn giải với thông tin về việc bằng cách nào các biến chương trình được định nghĩa, dử dụng và kết thúc. Tiêu chuẩn mạnh nhất yêu cầu từ định nghĩa của biến đến việc sử dụng định nghĩa đó được thực thi. Để giảm số đường đi yêu cầu, những chiến lược yếu hơn như định nghĩa tất cả và sử dụng tất cả được dùng.

* + 1. **Các mô hình tham khảo cỏ kiểm thử dựa trên mã nguồn**

Mặc dù bản thân không phải là một kỹ thuật, cấu trúc điều khiển của một chương trình có thể được hình ảnh hoác để biểu diễn việc sử dụng một biểu đồ như kỹ thuật dựa trên mã nguồn. Một biểu đồ là đồ thị có hứng, các nút và cung biểu diễn cho các yếu tố của chương trình. Ví dụ, các nút có thể biểu diễn các lệnh hoặc tần xuất không bị can thiệp của lệnh và cung có thể biểu diễn sử trao đổi giữa các nút.

* 1. **Kỹ thuật dựa trên lỗi**
     1. **Dự đoán lỗi**

Trong việc dự đoán lỗi, các trường hợp kiểm thử được theiets kế đặc biệt bởi kỹ sư phần mềm người mà cố gắng dự đoán trước các lỗi có thể xảy ra trong chương trình cho trước. Một nguồn thông tin tốt là lịch sử lỗi được tìm ra trong những dự án có trước cũng như kỹ năng của kỹ sư phần mềm.

* + 1. **Kiểm thử đột biến**

Một đột biến là một phiên bản chỉnh sửa nhẹ của chương trình đang kiểm thử, với một sử thay đổi nhẹ để khác biệt với chương trình cũ. Mỗi lần kiểm thử là kiểm thử phiên bản gốc và toàn bộ các đột biến: nếu trường hợp kiểm thử thành công trong việc định danh sự khác biệt giữa chương trình và đột biến thì đột biến sẽ bị tiêu diệt. Nói cách khác, khi một đột biến được kiểm thử mà trả kết quả khác so với bản gốc thì đột biến bị tiêu diệt và xem như bản gốc đạt tiêu chuẩn. Khi bản đột biến mà cho kết quả giống bản gốc thì xem như kiểm thửu thất bại. Hiệu của của kỹ thuật này được đánh giá bằng số lượng lớn của các đột biến bị tiêu diệt. Các đột biến thường được tạo ra một cách tự động và kiểm thử thực hiện theo một cách có hệ thống.

* 1. **Kỹ thuật dựa trên việc sử dụng**
     1. **Hồ sơ vận hành**

Trong việc kiểm thử sự đánh giá độ tin cậy, mội trường kiểm thử được tạo ra gần với môi trường vận hành của phần mềm hoặc gọi là hồ sơ vận hành. Mục tiêu là để suy ra từ các kết quả đã được theo dõi độ tin cậy tương lai của phần mềm khi được sử dụng thật. để làm điều này, đầu vào được gán các xác suất hoặc hồ sơ theo tần suất xảy ra trong thực tế. Hồ sơ vận hafnhc so thể được dùng trong kiểm thử hệ thống để hướng dẫn bắt nguồn của các trường hợp kiểm thử sẽ đánh giá sự tin cậy và thử nghiệm liên quan đến việc sử dụng và các chức năng khác nhau giống với những gì xảy ra trong môi trường kiểm thử vận hành.

* + 1. **Quan sát người dùng**

Các nguyên tắc sử dụng có thể cung cấp hướng dẫn cho việc tìm ra vấn đề trong thiết kế giao diện. Nguồn thông tin cho vấn đề này có thể được lấy qua các nguồn như phỏng vấn và làm khảo sát người dùng.

* 1. **Kỹ thuật dựa trên mô hình**
     1. **Bảng quyết đinh**

Những bảng quyết định biểu diễ các mối quan hệ logic giữa các điều kiện và hành động. Các trường hợp kiểm thử có được một cách hệ thống bằng việc xem xét mọi tổ hợp của điều kiện và các hành động phản ứng. Một kỹ thauatj liên quan là đồ thị nguyên nhân - ảnh hưởng.

* + 1. **Máy trạng thái hữu hạn**

Bằng việc mô hình hóa một chương trình như một máy trạng thái hữu hạn, việc kiểm thử được lựa chọn để bao phủ các trạng thái và sự chuyển tiếp.

* + 1. **Chi tiết kỹ thuật thông thường**

Trạng thái hóa cả chi tiết kỹ thuật trong ngôn ngữ hình thức cho phép sự dẫn xuất tự động các trường hợp kiểm thử chức năng và cùng lúc cung cấp một dự đoán cho vieeck kiểm tra kết quả kiểm thử. TTCN3 là một ngôn ngữ được phát triển để viết ra các trường hợp kiểm thử. Định nghĩa được hiểu với những yêu cầu riêng biệt của kiểm thử các hệ thống truyền thông, vì thế nó đặc biệt thích hợp cho kiểm thử các giao thức truyền thông phức tạp.

* + 1. **Mô hình luồng công việc**

Mô hình luồng công việc chỉ ra tần xuất của các hoạt động được thực hiện bởi con người hay ứng dụng phần mềm, thường được biểu diễn thông qua các định nghiwax hình ảnh. Mỗi tần suất của hành động cấu thành một luồng công việc.

* 1. **Kỹ thuật dựa trên tính chất của phần mềm**

Các kỹ thuật bên trên áp dụng cho toàn bộ các loại phần mềm. các kỹ thuật phụ cho kiểm thử dựa trên tính chấn của phần mềm như:

* Phần mềm hướng đối tượng
* Phần mềm dựa trên bộ phận
* Phần mềm nền web
* Các chương trình tương tranh
* Phần mềm dựa trên giao thức
* Hệ thống thời gian thực
* Hệ thống an toàn nguy cấp
* Phần mềm hướng dịch vụ
* Phần mềm mã nguồn mở
* Phần mềm nhúng
  1. **Chọn và kết hợp các kỹ thuật**
     1. **Kết hợp chức năng và cấu trúc**

Các kỹ thuật kiểm thử dựa trên mô hình và mã nguồn thường tương phản như kiểm thử chức năng và cấu trúc. Hai cách tiếp cận lựa chọn kiểm thử này không được xem như thay thế cho nhau mà được xem là bổ sung cho nhau; trong thực tế, chúng sử dụng các nguồn thông tin khác nhau và đã chỉ ra các loại vấn đề tiêu biểu khác nhau. Chúng có thể được dùng kết hợp và phụ thuộc vào xem xét đầu tư.

* + 1. **Tất định và ngẫu nhiên**

Các trường hợp kiểm thử được chọn theo cách tất định, theo một trong số các kỹ thuật hoặc thực hiện ngẫu nhiên từ sự phân phối đầu vào như trong kiểm thử độ tin cậy. Một vài so sánh theo kiểu phân tích và kinh nghiệm được chọn để phân tích các điều kiện mà tạo cách tiếp cận có hiệu quả hơn.

1. **Các phép đo liên quan đến việc kiểm thử**
   1. **Đánh giá chương trình đang kiểm thử**
      1. **Các phép đo chương trình hỗ trợ lên kế hoạc và thiết kế kiểm thử**

Các phép đó dựa trên kích cỡ của phần mềm (ví dụ: các dòng mã nguồn hoặc kích thước chức năng) hoặc cấu trúc phần mềm có thể được dùng để hướng dẫn kiểm thử. Các phép đo cấu trúc bao gồm các phép đo mà quyết định tần suất của mô đun này gọi thực hiện mô đun khác.

* + 1. **Các loại lỗi, phân loại và thống kê**

Văn học kiểm thử đa dạng trong phân loại và nguyên tắc phân loại lỗi. Để tạo ra kiêm thử hiệu quả hơn thì quan trọng là biết loại lỗi nào có thể được phát hiện trong phần mềm đang kiểm thử và tần xuất tương đối những lỗi đó tồn tại trong quá khứ. Thông tin này có thể hữu ích trong việc dự đoán chất lượng cũng như trong việc cải tiến quá trình.

* + 1. **Mật độ lỗi**

Một chương trình đang kiểm thử có thể được đánh giá bằng việc tính toán các lỗi đã được tìm ra như tỉ lệ giữa số lỗi được tìm thấy và độ lớn cửa chương trình.

* + 1. **Đánh giá độ tin cậy**

Một ước lượng thống kê của độ tin cậy phần mềm có thể được dùng để đánh giá một sản phẩm phần mềm và quết định kiểm thử có thể dừng lại hay không bằng việc quan sát độ tin cậy được tạo ra.

* + 1. **Những mô hình phát triển độ tin cậy**

Những mô hình phát triển độ tin cậy cung cấp dự đoán dộ tin cậy dựa trên lỗi. Chúng giả định rằng các lỗi gây ra các thất bại được quan sát đều được sửa, độ tin cậy của sản phầm đữa ra một xu thế tăng dần. Rất nhiều mô hình kiểu này được đưa ra. Đáng chú ý, những mô hình này được chi ra là 2 loại là mô hình đếm lỗi và mô hình thời gian giữa các lỗi.

* 1. **Đánh giá các kiểm thử được thực hiện**
     1. **Các phép đo phủ**

Một vài tiêu chuẩn kiểm thử yêu cầu rằng các trường hợp kiểm thử thực hiện một tập yếu tố được định danh trong chương trình hoặc trong chi tiết kỹ thuật một các hệ thống. Để đánh giá độ chi tiết của các kiểm thử được thực hiện, các kỹ sư phần mềm cso thể theo dõi các yếu tố được bao phủ để họ có thể đo đạc tỉ lệ giữa yếu tố được phủ và tổng số một các linh động. Ví dụ, có thể đó tỉ lệ phần trăm của các nhánh được phủ trong biểu đồ dòng chảy chương trình hoặc phần trăm yêu cầu chức năng được thực hiện và danh sách chức năng trong tài liệu chi tiết kỹ thuật. Các tiêu chí phù hợp dứa trên mã nguồn yêu cầu sự đo đạc thích hợp của chương trình đang kiểm thử.

* + 1. **Gieo lỗi**

Trong việc gieo lỗi, vài lỗi được đưa nhân tạo vào chương trình trước khi kiểm thử. Khi kiểm thử được tiến hành, và trong số này sẽ được bộc lộ cũng như và lỗi có sẵn. Theo lý thuyết, phụ thuộc vào việc cái nào và bao nhiêu lỗi nhân tạo được tìm ra, hiệu quả kiểm thử có thể được đánh giá và số lỗi thực có thể được đánh giá. Trong thực tế, các nhà thống kê đặt ra câu hỏi thử phân phối và biểu diễn của các lỗi được gieo trước đó quan hệ vơi lỗi thật và một cỡ bé mẫu thử lên bất kỳ phéo ngoại suy nào được dựa trên. Một số khác tranh cãi rang kỹ thauatj này nên dùng thận trọng vì việc chèn lỗi vào phần mềm liên quan đến mạo hiểm rõ rằng rằng có thể quên các lỗi đó trong chương trình.

* + 1. **Điểm đột biến**

Trong kiểm thử đột biến, tỉ lệ các đột biến bị tiêu diệt với tổng số đột biến có thể là phép đo cho sự hiệu của của tập kiểm thử được thực hiện.

* + 1. **Sự so sánh và hiệu quả tương đối của các kỹ thuật khác nhau**

Và nghiên cứu được tiến hành đê so sánh hiệu quả tương đối của việc sử dụng các kỹ thuật khác nhau. Quan trọng là để được chính xác như đối với thuộc tính dựa vào đó các kỹ thuật đang được đánh giá, những gì là ý nghĩa của thuật ngữ “hiệu quả”? Cách diễn dải có thể có báo gồm số các kiểm thử cần để tìm ra thất bại đầu tiên, tỉ lệ của số lỗi được tìm ra thông qua kiểm thử và toàn bộ số lỗi được tìm ra trong và sau khi kiểm thử, và cải thiện độ tin cậy được bao nhiêu. Các so sánh phân tích và thực nghiệm giữa các kỹ thuật khác nhau được tiến hành theo từng ý niệm về hiệu quả quy định ở trên.

**5. Quy trình kiểm thử**

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. Nhìn nhận thực tế**

**5.1.1. Thái độ / Lập trình giảm thiểu “cái tôi”**

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. Hướng dẫn kiểm thử**

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. Quản lý quá trình kiểm thử**

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. Tài liệu kiểm thử và sản phẩm công việc**

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. Phát triển hướng kiểm thử (TDD)**

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. Đội kiểm thử nội bộ và độc lập**

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.  Chi phí / dự toán và đo lường quá trình kiểm thử**

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.  Kết thúc kiểm thử**

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.  Tái sử dụng kiểm thử và Mẫu kiểm thử**

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. Hoạt động kiểm thử**

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. Lập kế hoạch**

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. Tạo ra các trường hợp kiểm thử**

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. Phát triển môi trường kiểm thử**

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. Thực hiện kiểm thử**

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. Đánh giá kết quả kiểm thử**

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. Báo cáo / nhật ký kiểm thử**

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.  Theo dõi khiếm khuyết**

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. Công cụ kiểm thử phần mềm**

**6.1. Công cụ kiểm thử hỗ trợ**

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. Lựa chọn công cụ**

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. Danh mục các công cụ kiểm thử**

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

**• Test harnesses (drivers, stubs):**

- Trong kiểm thử phần mềm, test harness (khai thác kiểm thử) đề cập đến một bộ sưu tập các phần mềm và bộ dữ liệu kiểm thử được cấu hình sẵn để kiểm thử một đơn vị chương trình bằng cách chạy nó trong nhiều điều kiện khác nhau và chúng ta sẽ theo dõi hành vi và kết quả đầu ra của nó.

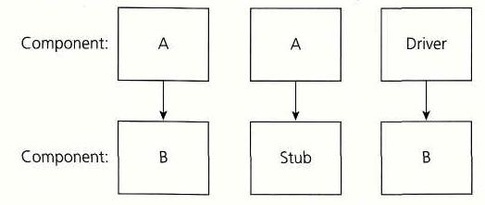
- Cung cấp một môi trường điều khiển, trong đó việc kiểm thử có thể được thực hiện và kết quả đầu ra có thể được ghi lại. Để thực hiện các phần của một chương trình, các drivers và stubs được cung cấp để mô phỏng một cách tương ứng.

- Sự khác nhau giữa Drivers và Stubs:

Drivers: là loại giả mô-đun, cái được gọi là "các chương trình gọi", được sử dụng trong kiểm thử tích hợp từ dưới lên, được sử dụng khi chương trình chính đang được xây dựng.

Stub: có nghĩa là một mô hình giả của một module cụ thể.

- Ví dụ:



Giả sử chúng ta cần kiểm thử sự tích hợp giữa mô đun A và B và chúng ta đã xây dựng được mô đun A trong khi mô đun B chưa có trong giai đoạn phát triển.

Như vậy, khi cần kiểm thử mô đun A mà chưa có mô đun B, chúng ta cần tạo ra một giả mô đun để thay thế cho mô đun B, có tính năng tương tự như mô đun B sau đó có thể sử dụng. Trường hợp mô đun giả này được gọi là Stub.

Trường hợp ngược lại, chẳng hạn chúng ta đã xây dựng xong mô đun B nhưng mô đun A chưa xây dựng xong, để kiểm thử mô đun B, chúng ta sẽ sử dụng một giải mô đun khác truy xuất dữ liệu đến mô đun B được gọi là Driver.

• **Test generators:** cung cấp các hỗ trợ trong trường hợp tự tạo ra các kiểm thử. Các dữ liệu kiểm thử được tạo ra có thể là ngẫu nhiên hoặc dựa trên một yếu tố nào đó như: 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 trong việc quyết định một kết quả kiểm thử có thành công hay không.

• **Coverage analyzers and instrumenters:** phân tích mức độ bao phủ trong tất cả những yêu cầu của các tiêu chuẩn 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.