**TÀI LIỆU KIỂM THỬ**

# TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

## 1.1 Định nghĩa và các thuật ngữ về kiểm thử

### 1.1.1 Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó (Theo Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi (Theo “The Art of Software Testing” – Nghệ thuật kiểm thử phần mềm).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau (Theo Bách khoa toàn thư mở Wikipedia).

Có thể định nghĩa một cách dễ hiểu như sau: Kiểm thử phần mềm là một tiến trình hay một tập hợp các tiến trình được thiết kế để đảm bảo mã hóa máy tính thực hiện theo cái mà chúng đã được thiết kế để làm, và không thực hiện bất cứ thứ gì không mong muốn. Đây là một pha quan trọng trong quá trình phát triển hệ thống, giúp cho người xây dựng hệ thống và khách hàng thấy được hệ thống mới đã đáp ứng yêu cầu đặt ra hay chưa.

### 1.1.2 Các thuật ngữ trong kiểm thử

Lỗi (Error): Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm.

Sai (Fault): Sai là kết quả của lỗi, hay nói khác đi, lỗi sẽ dẫn đến sai. Cũng có thể nói sai là một biểu diễn của lỗi dưới dạng một biểu thức, chẳng hạn chương trình, văn bản, sơ đồ dòng dữ liệu, biểu đồ lớp,.... Sai lầm có thể khó bị phát hiện. Khi nhà thiết kế mắc lỗi bỏ sót trong quá trình thiết kế, sai kết quả từ lỗi đó là thiếu mất cái gì đó mà lẽ ra cần phải có. Sai về nhiệm vụ xuất hiện khi vào sai thông tin, còn sai về bỏ quên xuất hiện khi không vào đủ thông tin. Loại sai thứ hai khó phát hiện và khó sửa hơn loại sai thứ

Thất bại (Failure): Thất bại xuất hiện khi một lỗi được thực thi. Có hai điều cần lưu ý ở đây. Một là thất bại chỉ xuất hiện dưới dạng có thể chạy được mà thông thường là mã nguồn. Hai là các thất bại chỉ liên kết với các lỗi về nhiệm vụ. Còn các thất bại tương ứng với các lỗi về bỏ quên thì xử lý thế nào? Những cái lỗi không bao giờ được tiến hành, hoặc không được tiến hành trong khoảng thời gian dài cần được xử lý thế nào? Virus Michaelangelo là một ví dụ về lỗi loại này.

Sự cố (Incident): Khi thất bại xuất hiện, nó có thể hiển thị hoặc không, tức là rõ ràng hoặc không rõ ràng đối với người dùng hoặc người kiểm thử. Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này.

Yêu cầu của khách hàng và đặc tả của phần mềm: Phần mềm được viết để thực hiện các nhu cầu của khách hàng. Các nhu cầu của khách hàng được thu thập, phân tích và khảo cứu và là cơ sở để quyết định chính xác các đặc trưng cần thiết mà sản phẩm phần mềm cần phải có. Dựa trên yêu cầu của khách hàng và các yêu cầu bắt buộc khác, đặc tả được xây dựng để mô tả chính xác các yêu cầu mà sản phẩm phần mềm cần đáp ứng, và có giao diện thế nào. Tài liệu đặc tả là cơ sở để đội ngũ phát triển phần mềm xây dựng sản phẩm phần mềm. Khi nói đến thất bại trên đây là nói đến việc sản phẩm phần mềm không hoạt động đúng như đặc tả. Lỗi một khi được tiến hành có thể dẫn đến thất bại. Do đó, lỗi về bỏ quên được coi là tương ứng với các lỗi khi xây dựng đặc tả.

Kiểm chứng và thẩm định: Kiểm chứng (verification) và thẩm định (validation) hay được dùng lẫn lộn, nhưng thực ra chúng có ý nghĩa khác nhau. Kiểm chứng là quá trình để đảm bảo rằng một sản phẩm phần mềm thỏa mãn đặc tả của nó. Còn thẩm định là quá trình để đảm bảo rằng sản phẩm đáp ứng được yêu cầu của người dùng (khách hàng). Trong thực tế, chúng ta cần thực hiện kiểm chứng trước khi thực hiện việc thẩm định sản phẩm phần mềm. Vì vậy, chúng ta có thuật ngữ V&V (Verification & Validation). Lý do của việc này là chúng ta cần đảm bảo sản phẩm đúng với đặc tả trước. Nếu thực hiện việc thẩm định trước, một khi phát hiện ra lỗi, chúng ta không thể xác định được lỗi này do đặc tả sai hay do lập trình sai so với đặc tả.

Vai trò của kiểm thử phần mềm: Kiểm thử phần mềm đóng vai trò quan trọng trong việc đánh giá và thu được chất lượng cao của sản phẩm phần mềm trong quá trình phát triển. Thông qua chu trình “ kiểm thử - tìm lỗi - sửa lỗi”, ta hy vọng chất lượng của sản phẩm phần mềm sẽ được cải tiến. Mặt khác, thông qua việc tiến hành kiểm thử mức hệ thống trước khi cho lưu hành sản phẩm, ta biết được sản phẩm của ta tốt ở mức nào. Vì thế, nhiều tác giả đã mô tả việc kiểm thử phần mềm là một quy trình kiểm chứng để đánh giá và tăng cường chất lượng của sản phẩm phần mềm. Quy trình này gồm hai công việc chính là phân tích tĩnh và phân tích động.

Phân tích tĩnh: Việc phân tích tĩnh được tiến hành dựa trên việc khảo sát các tài liệu được xây dựng trong quá trình phát triển sản phẩm như tài liệu đặc tả nhu cầu người dùng, mô hình phần mềm, hồ sơ thiết kế và mã nguồn phần mềm. Các phương pháp phân tích tĩnh truyền thống bao gồm việc khảo sát đặc tả và mã nguồn cùng các tài liệu thiết kế.

Phân tích động: Phân tích động liên quan đến việc thực thi chương trình để phát hiện những thất bại có thể có của chương trình, hoặc quan sát các tính chất nào đó về hành vi và hiệu quả (performance). Vì gần như không thể thực thi chương trình trên tất cả các dữ liệu đầu vào có thể, ta chỉ có thể chọn một tập con các dữ liệu đầu vào để thực thi, gọi là các “ca kiểm thử”.

### 1.1.3 Tiến trình kiểm thử

Kiểm thử thường bao gồm các bước:

Bước 1: Lập kế hoạch kiểm thử, xác định kịch bản kiểm thử.

Bước 2: Tạo dữ liệu thử (thiết kế các ca kiểm thử).

* + Kiểm thử với tất cả dữ liệu vào là cần thiết.
  + Chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào.

Bước 3: Thực thi chương trình trên dữ liệu thử.

* + Cung cấp dữ liệu thử.
  + Thực thi.
  + Ghi nhận kết quả.

Bước 4: quan sát kết quả kiểm thử:

* + Thực hiện trong khi hoặc sau khi thực hiện.
  + So sánh kết quả nhận được và kết quả mong đợi.

### 1.1.4 Nguyên tắc kiểm thử

Có 7 nguyên tắc trong kiểm thử phần mềm:

* + **Kiểm thử chỉ chứng tỏ được việc có lỗi.**
  + **Kiểm thử toàn bộ là không thể.**
  + **Kiểm thử càng sớm càng tốt.**
  + **Sự tập trung của lỗi.**
  + **Nghịch lí thuốc trừ sâu.**
  + **Kiểm thử phụ thuộc vào ngữ cảnh.**
  + **Sự sai lầm về việc không có lỗi.**

## 1.2 Phương pháp kiểm thử

*Kiểm thử tĩnh – Static Testing*: Là phươn pháp thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay, thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình. Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế người mà viết mã lệnh một mình.

*Kiểm thử động – Dynamic testing*: Kiểm thử động kiểm tra cách thức hoạt động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian. Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy. Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không.

## 1.3 Các chiến lược kiểm thử

### 1.3.1 Kiểm thử hộp đen

*Kiểm thử hộp đen – Black box testing:* Một trong những chiến lược kiểm thử quan trọng là kiểm thử hộp đen, hướng dữ liệu, hay hướng vào/ra. Kiểm thử hộp đen xem chương trình như là một “hộp đen”. Mục đích của bạn là hoàn toàn không quan tâm về cách cư xử và cấu trúc bên trong của chương trình. Thay vào đó, tập trung vào tìm các trường hợp mà chương trình không thực hiện theo các đặc tả của nó.

Theo hướng tiếp cận này, dữ liệu kiểm tra được lấy chỉ từ các đặc tả:

* Phân lớp tương đương *– Equivalence partitioning*
* Phân tích giá trị biên *– Boundary value analysis*
* Kiểm thử mọi cặp *– All-pairs testing*
* Kiểm thử fuzz *– Fuzz testing.*
* Kiểm thử dựa trên mô hình *– Model-based testing.*
* Ma trận dấu vết *– Traceability matrix.*
* Kiểm thử thăm dò *– Exploratory testing.*
* Kiểm thử dựa trên đặc tả *– Specification-base testing.*

Ưu điểm: Kiểm thử hộp đen không có mối liên quan nào tới mã lệnh, và kiểm thử viên chỉ rất đơn giản tâm niệm là: một mã lệnh phải có lỗi. Sử dụng nguyên tắc “ Hãy đòi hỏi và bạn sẽ được nhận”, những kiểm thử viên hộp đen tìm ra lỗi mà những lập trình viên đã không tìm ra

Nhược điểm: kiểm thử hộp đen “giống như là đi trong bóng tối mà không có đèn vậy”, bởi vì kiểm thử viên không biết các phần mềm được kiểm tra thực sự được xây dựng như thế nào. Đó là lý do mà có nhiều trường hợp mà một kiểm thử viên hộp đen viết rất nhiều ca kiểm thử để kiểm tra một thứ gì đó mà đáng lẽ có thể chỉ cần kiểm tra bằng 1 ca kiểm thử duy nhất, và/hoặc một số phần của chương trình không được kiểm tra chút nào.

Do vậy, kiểm thử hộp đen có ưu điểm của “một sự đánh giá khách quan”, mặt khác nó lại có nhược điểm của “thăm dò mù”.

### 1.3.2 Kiểm thử hộp trắng

Kiểm thử hộp trắng – White box testing: Là một chiến lược kiểm thử khác, trái ngược hoàn toàn với kiểm thử hộp đen, kiểm thử hộp trắng hay kiểm thử hướng logic cho phép bạn khảo sát cấu trúc bên trong của chương trình. Chiến lược này xuất phát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình. Kiểm thử viên sẽ truy cập vào cấu trúc dữ liệu và giải thuật bên trong chương trình (và cả mã lệnh thực hiện chúng).

Các phương pháp kiểm thử hộp trắng:

*+ Kiểm thử giao diện lập trình ứng dụng – API testing (application programming interface):* là phương pháp kiểm thử của ứng dụng sử dụng các API công khai và riêng tư

*+ Bao phủ mã lệnh – Code coverage:* tạo các kiểm tra để đáp ứng một số tiêu chuẩn về bao phủ mã lệnh.

*+ Các phương pháp gán lỗi – Fault injection.*

*+ Các phương pháp kiểm thử hoán chuyển – Mutation testing methods.*

*+ Kiểm thử tĩnh–Static testing:* kiểm thử hộp trắng bao gồm mọi kiểm thử tĩnh.

Phương pháp kiểm thử hộp trắng cũng có thể được sử dụng để đánh giá sự hoàn thành của một bộ kiểm thử mà được tạo cùng với các phương pháp kiểm thử hộp đen. Điều này cho phép các nhóm phần mềm khảo sát các phần của 1 hệ thống ít khi được kiểm tra và đảm bảo rằng những điểm chức năng quan trọng nhất đã được kiểm tra.

### 1.3.3 Kiểm thử hộp xám

Kiểm thử hộp xám đòi hỏi phải có sự truy cập tới cấu trúc dữ liệu và giải thuật bên trong cho những mục đích thiết kế các ca kiểm thử, nhưng là kiểm thử ở mức người sử dụng hay mức hộp đen.

Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, giống như một chiếc *“hộp xám”,* bởi vì đầu vào và đầu ra rõ ràng là ở bên ngoài *“hộp đen”* mà chúng ta vẫn gọi về hệ thống được kiểm tra. Sự khác biệt này đặc biệt quan trọng khi quản lý kiểm thử tích hợp.

*Intergartion testing* giữa 2 modun mã lệnh được viết bởi hai chuyên viên thiết kế khác nhau, trong đó chỉ giao diện là được đưa ra để kiểm thử. Kiểm thử hộp xám có thể cũng bao gồm cả thiết kế đối chiếu để quyết định, ví dụ, giá trị biên hay thông báo lỗi.

## 1.4 Các cấp độ kiểm thử

### 1.4.1 Kiểm thử đơn vị - unit test

Kiểm thử đơn vị là hoạt động kiểm thử nhỏ nhất.Kiểm thử thực hiện trên các hàm hay thành phần riêng lẻ.

Cần hiểu biết về thiết kế chương trình và code.

Thực hiện bởi Lập trình viên (không phải kiểm thử viên).

Đơn vị: Là thành phần nhỏ nhất của phần mềm có thể kiểm thử được. Ví dụ: Các hàm, lớp, thủ tục, phương thức.

Mục đích của việc kiểm tra unit test là cô lập từng thành phần của chương trình và chứng minh các bộ phận riêng lẻ chính xác về các yêu cầu chức năng.

### 1.4.2 Kiểm thử tích hợp **- Integration test**

Kiểm thử tích hợp nhằm phát hiện lỗi giao tiếp xảy ra giữa các thành phần cũng như lỗi của bản thân từng thành phần (nếu có).

Thành phần có thể là:

* Các module.
* Các ứng dụng riêng lẻ.
* Các ứng dụng client/server trên một mạng.

Integration Test có 2 mục tiêu chính:

* Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
* Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm tra ở mức hệ thống (System Test).

### 1.4.3 Kiểm thử hệ thống

Kiểm thử hệ thống là một mức của tiến trình kiểm thử phần mềm khi các module và tích hợp các module đã được test.

Mục tiêu của kiểm thử hệ thống là để đánh giá phần mềm có tuân thủ theo các yêu cầu đã đưa ra không.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

System Test thường được thực hiện bởi một nhóm kiểm tra viên hoàn toàn độc lập với nhóm phát triển dự án.

### 1.4.4 Kiểm thử chấp nhận - acceptance test

Kiểm thử chấp nhận là một cấp độ trong tiến trình kiểm thử phần mềm nhằm kiểm thử hệ thống về khả năng chấp nhận được.

•  Mục tiêu của kiểm thử này là để đánh giá sự tuân thủ của hệ thống với các yêu cầu nghiệp vụ và thẩm định xem đã có thể chấp nhận để bàn giao chưa.

•  Kiểm thử chấp nhận được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện).

Kiểm thử chấp nhận là hoạt động thẩm định (validation) và do người sử dụng, khách hàng kiểm tra. Họ xem hệ thống phần mềm có đáp ứng đúng như mong muốn của họ không, không cần tài liệu đặc tả. Chúng ta cần kiểm thử chấp nhận vì đặc tả có thể đã có khiếm khuyết hoặc người khách hàng và người phát triển cùng đọc một tài liệu nhưng hiểu không hoàn toàn như nhau.

### 1.4.5 Kiểm thử hồi quy – regression test

Có bất cứ sự thay đổi trong một ứng dụng phần mềm sẽ hoàn toàn có thể gây ảnh hưởng tới những thành phần khác. Regression testing được thực hiện để xác minh rằng các lỗi đã được sửa không làm thay đổi kết quả của chức năng khác hoặc ảnh hưởng đến các thành phần khác. Mục đích của Regression testing là để đảm bảo rằng thay đổi sẽ không dẫn đến lỗi khác được phát sinh trong ứng dụng phần mềm. Các kỹ thuật trong kiểm thử hồi quy giúp chúng ta xác định các ca kiểm thử cần thực hiện lại để chỉ kiểm tra hệ thống ở những phần có liên quan, thay vì phải thực hiện lại toàn bộ hai công việc này.

## 1.5 Lựa chọn phương pháp kiểm thử

Khi phát triển phần mềm, việc thực hiện kiểm thử là bắt buộc, cho dù người thực hiện kiểm thử có thể là developer hoặc là tester. Vì thế, có kiến thức về kiểm thử, lựa chọn loại hình kiểm thử phù hợp với sản phẩm là điều cần thiết cho bất cứ người nào tham gia vào quá trình làm sản phẩm. Có 2 phương pháp kiểm thử: kiểm thử thủ công và kiểm thử tự động:

Khi phát triển phần mềm, việc thực hiện kiểm thử là bắt buộc, cho dù người thực hiện kiểm thử có thể là developer hoặc là tester. Vì thế, có kiến thức về kiểm thử, lựa chọn loại hình kiểm thử phù hợp với sản phẩm là điều cần thiết cho bất cứ người nào tham gia vào quá trình làm sản phẩm. Có 2 phương pháp kiểm thử: kiểm thử thủ công và kiểm thử tự động.

### 1.5.1 Kiểm thử thủ công

Kiểm thử thủ công là Tester làm mọi thứ bằng tay, từ viết test case đến thực hiện test, mọi thao tác như nhập điều kiện đầu vào, thực hiện một số sự kiện khác như click nút và quan sát kết quả thực tế, sau đó so sánh kết quả thực tế với kết quả mong muốn trong test case, điền kết quả test. Mọi thứ hoàn toàn bằng tay. Hiện nay, phần lớn các tổ chức, các công ty phần mềm, hoặc các nhóm làm phần mềm đều thực hiện kiểm thử thủ công là chủ yếu.

### 1.5.2 Kiểm thử tự động

Kiểm thử tự động: nghĩa là áp dụng một số công việc kiểm thử bằng công cụ, phần mềm hỗ trợ test - test tool - không phải nhất thiết hoàn toàn mọi hoạt động bằng cách tự động hóa. Trong quá trình kiểm thử có rất ít hoặc không có sự tương tác của con người, giúp cho người thực hiện việc kiểm thử phần mềm (tester) không phải lặp đi lặp lại các bước nhàm chán.

Kiểm thử tự động giúp giảm chi phí kiểm thử bằng cách hỗ trợ quá trình kiểm thử thông qua các công cụ phần mềm.

Có thể thực hiện các kiểm thử một cách liên tục, lặp lại và giảm chi phí cho nhân lực kiểm thử.

Luôn đảm bảo hoạt động theo một kịch bản duy nhất – không bị ảnh hưởng như với kiểm thử viên.

### 1.5.3 So sánh ưu – nhược điểm của phương pháp kiểm thử thủ công và kiểm thử tự động

|  |  |  |
| --- | --- | --- |
| Phương pháp | Kiểm thử thủ công | Kiểm thử tự động |
| Ưu điểm | -Thích hợp kiểm tra sản phẩm lần đầu tiên.  -Thích hợp kiểm thử trong trường hợp các test case chỉ phải thực hiện một số ít lần.  -Giảm được chi phí ngắn hạn | -Thích hợp với trường hợp phải test nhiều lần cho một case.  -Có tính ổng định và tin cậy cao hơn.  -Giảm thời gian và công sức.   * -Giảm chi phí đầu tư dài hạn   -Công cụ kiểm thử tự động mang tính ổn định cao hơn so với con người. Đặc biệt, trong những trường hợp nhiều test case nên độ tin cậy cậy của kiểm thử sẽ đạt mức tối ưu hơn so với kiểm thử thủ công. |
| Nhược điểm | --Tốn thời gian. Đối với mỗi lần phát hành(relase) người kiểm thử vẫn phải thực hiện lại một tập hợp các test case đã chạy dẫn đến sự mệt mỏi và lãng phí. | -Mất thời gian tìm hiểu công cụ test tự động và cách tạo các dữ liệu chuẩn bị cho quá trình kiểm thử.  -Đối với những tester không hiểu biết về code sẽ khó khăn trong việc tạo dữ liệu test.  -Chi phí đầu tư ban đầu lớn. |

So sánh ưu – nhược điểm của các phương pháp nhằm đảm bảo giảm tối đa những lỗi không thể lường trước trong bất kỳ kịch bản nào.

=> Kết luận: Từ việc nhận thức được yêu cầu thực tế của ứng dụng và những ưu – nhược điểm của các phương pháp kiểm thử nên em quyết định sử dụng phương pháp kiểm thử tự động để kiểm thử cho ứng dụng lần này . Bởi tính ổn định cao và tốc độ nhanh mà nó mang lại