**BÁO CÁO MÔN HỌC**

# Tổng quan về kiểm thử phần mềm

* 1. **Cơ sở lý thuyết tổng quan**

Kiểm thử nhằm đánh giá chất lượng hoặc tính chấp nhận được của sản phẩm. Kiểm thử cũng nhằm phát hiện lỗi hoặc bất cứ vấn đề gì về sản phẩm. Chúng ta cần kiểm thử vì biết rằng con người luôn có thể mắc sai lầm. Điều này đặc biệt đúng trong lĩnh vực phát triển phần mềm và các hệ thống điều khiển bởi phần mềm. Chương này nhằm phác họa một bức tranh tổng thể về kiểm thử phần mềm. Các chương còn lại sẽ nằm trong khuôn khổ của bức tranh này và ở mức chi tiết hơn

* **Lỗi (Error):** Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm. Trong thực tế, con người luôn có thể phạm lỗi. Khi lập trình viên phạm lỗi trong lập trình, ta gọi các lỗi đó là bug (con bọ). Lỗi có thể phát tán. Chẳng hạn, một lỗi về xác định yêu cầu có thể dẫn đến sai lầm về thiết kế và càng sai khi lập trình theo thiết kế này. Lỗi là nguyên nhân dẫn đến sai.
* **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 nhất.
* **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. Nó chỉ được tiến hành vào ngày sinh của Michaelangelo, tức ngày 6/3 mà thôi. Việc khảo sát có thể ngăn chặn nhiều thất bại bằng cách tìm ra các lỗi thuộc cả hai loại.
* **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.
* **Chất lượng và độ tin cậy của phần mềm:** Theo từ điển, chất lượng của một sản phẩm được thể hiện bằng các đặc trưng phù hợp với đặc tả của nó. Theo cách hiểu này, chất lượng của một sản phẩm phần mềm là sự đáp ứng các yêu cầu về chức năng (tức là các hàm cần được tính toán), sự hoàn thiện và các chuẩn đã được đặc tả, cùng các đặc trưng mong chờ từ mọi sản phẩm phần mềm chuyên nghiệp. Chất lượng phần mềm đặc trưng cho “độ tốt, độ tuyệt hảo” của phần mềm, và gồm có các yếu tố về chất lượng như: tính đúng đắn (hành vi đúng như đặc tả), tính hiệu quả (tiết kiệm thời gian và tiền bạc), độ tin cậy, tính khả kiểm thử (kiểm thử được và dễ), dễ học, dễ sử dụng, dễ bảo trì ... Như vậy, độ tin cậy chỉ là một yếu tố để đánh giá chất lượng phầm mềm. Người kiểm thử hay nhầm lẫn độ tin cậy với chất lượng. Khi kiểm thử đạt tới mức phần mềm chạy ổn định, có thể phụ thuộc vào nó, người kiểm thử thường cho rằng phần mềm đã đạt chất lượng cao. Các yếu tố về mặt chất lượng mà liên quan trực tiếp đến việc phát triển phần mềm được gọi là các tiêu chuẩn chất lượng như tính có cấu trúc, tính đơn thể, tính khả kiểm thử, ... Độ tin cậy của phần mềm là xác suất để phần mềm chạy không có thất bại trong một khoảng thời gian nhất định. Nó được xem là một yếu tố quan trọng của chất lượng phần mềm. Ngoài ra, thời gian trung bình cho việc khắc phục một sự cố cũng là một thông số quan trọng trong việc đánh giá độ tin cậy của sản phẩm phần mềm.
* **Kiểm thử:** Rõ ràng việc kiểm thử liên quan đến các khái niệm trên: lỗi, sai, thất bại và sự cố. Có hai mục đích chính của một phép thử: tìm thất bại hoặc chứng tỏ việc tiến hành của phần mềm là đúng đắn.
* **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. Các kỹ thuật khảo sát này sẽ được giới thiệu trong chương 4. Người ta cũng có thể dùng các kỹ thuật phân tích hình thức như kiểm chứng mô hình (model checking) và chứng minh định lý (theorem proving) để chứng minh tính đúng đắn của thiết kế và mã nguồn. Các kỹ thuật này tương đối phức tạp và nằm ngoài khuôn khổ của cuốn giáo trình này. Công việc này không động đến việc thực thi chương trình mà chỉ duyệt, lý giải về tất cả các hành vi có thể của chương trình khi được thực thi. Tối ưu hóa các chương trình dịch là các ví dụ về phân tích tĩnh.
  + ***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”. Chọn như thế nào để được các bộ dữ liệu đầu vào hiệu quả (tức là các bộ dữ liệu có xác suất phát hiện thất bại (nếu có) cao hơn là công việc cần suy nghĩ và là nội dung chính của các giáo trình này.
  1. **Lý thuyết chi tiết về kỹ thuật / phương pháp kiểm thử sẽ sử dụng**

Mục tiêu của kiểm thử là phải thiết kế các trường hợp kiểm thử có khả năng cao nhất trong việc phát hiện nhiều lỗi với thời gian và công sức tối thiểu. Do đó có thể chia các kỹ thuật kiểm thử thành hai loại:

* Kỹ thuật kiểm thử hộp đen (Black – box Testing) hay còn gọi là kỹ thuật kiểm thử chức năng (Functional Testing).
* Kỹ thuật kiểm thử hộp trắng (White – box Testing) hay còn gọi là kỹ thuật kiểm thử cấu trúc (Structural Testing).
  + 1. **Kỹ thuật kiểm thử hộp đen (Black – box Testing)**

Kiểm thử hộp đen còn được gọi là kiểm thử hướng dữ liệu (data -

driven) hay là kiểm thử hướng vào/ra (input/output driven).

Trong kỹ thuật này, người kiểm thử xem phần mềm như là một hộp đen.

Người kiểm thử hoàn toàn không quan tâm đến cấu trúc và hành vi bên trong của chương trình. Người kiểm thử chỉ cần quan tâm đến việc tìm các hiện tượng mà phần mềm không hành xử theo đúng đặc tả của nó. Do đó, dữ liệu kiểm thử sẽ xuất phát từ đặc tả.

* + 1. **Kỹ thuật kiểm thử hộp trắng (White – box Testing)**

Kiểm thử hộp trắng hay còn gọi là kiểm thử hướng logic, cho phép kiểm tra cấu trúc bên trong của phần mềm với mục đích bảo đảm rằng tất cả các câu lệnh và điều kiện sẽ được thực hiện ít nhất một lần. Người kiểm thử truy nhập vào mã nguồn chương trình và có thể kiểm tra nó, lấy đó làm cơ sở để hỗ trợ việc kiểm thử.

* + 1. **Kết luận**

Trong chương 1 đã nêu tổng quan về các cấp độ và loại kiểm thử phần mềm cơ bản. Kiểm thử hộp trắng xem xét chương trình ở mức độ chi tiết và phù hợp khi kiểm tra các môđun nhỏ. Tuy nhiên, kiểm thử hộp trắng có thể không đầy đủ vì kiểm thử hết các lệnh không chứng tỏ là chúng ta đã kiểm thử hết các trường hợp có thể. Ngoài ra chúng ta không thể kiểm thử hết các đường đi đối với các vòng lặp lớn.

Kiểm thử hộp đen chú trọng vào việc kiểm tra các quan hệ vào ra và những chức năng giao diêṇ bên ngoà i , nó thích hợp hơn cho các hê ̣t hống phần mềm lớn hay các thành phần quan troṇ g của chúng . Nhưng chỉ sử dụng kiểm thử hộp đen là chưa đủ. Bởi vì, kiểm thử chức năng chỉ dựa trên đặc tả của môđun nên không thể kiểm thử được các trường hợp không được khai báo trong đặc tả. Ngoài ra, do không phân tích mã nguồn nên không thể biết được môđun nào của chương trình đã hay chưa được kiểm thử, khi đó phải kiểm thử lại hay bỏ qua những lỗi tiềm ẩn trong gói phần mềm.

Phương pháp kiểm thử hộp trắng và kiểm thử hộp đen là hai phương pháp cơ bản có vai trò rất quan trọng trong quá trình phát triển phần mềm, nếu chúng ta biết kết hợp chúng để bổ sung khiếm khuyết lẫn nhau.

# Giới thiệu về công cụ kiểm thử tự động (Selenium)

* 1. **Tổng quan**

**Selenium là gì?**

Selenium là bộ kiểm thử tự động miễn phí (mã nguồn mở) dành cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau. Nó khá là giống với HP Quick Test Pro (QTP bây giờ là UFT) chỉ khác là Selenium thì tập trung vào việc tự động hoá các ứng dụng dựa trên nền tảng web. Kiểm thử được thực hiện bằng cách sử dụng công cụ Selenium thường được gọi là Kiểm thử Selenium. Selenium không chỉ là 1 công cụ độc lập mà là 1 bộ công cụ của phần mềm, mỗi bộ đều đáp ứng được nhu cầu kiểm thử khác nhau của 1 tổ chức. Nó có 4 thành phần.

* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid

Hiện tại, Selenium RC và WebDriver được hợp nhất thành một framework duy nhất để tạo ra Selenium 2. Còn Selenium 1 thì tham chiếu đến Selenium RC.

* 1. **Lịch sử ra đời**

**Sự ra đời của Selenium Remote Control (Selenium RC)**

Không may là người kiểm thử sử dụng Selenium Core phải cài đặt toàn bộ ứng dụng được thử nghiệm và máy chủ web trên các máy tính cục bộ của họ do những hạn chế của\*\* chính sách nguồn gốc\*\* giống nhau áp đặt. Vì vậy, một kỹ sư khác của ThoughtWork, **Paul Hammant**, đã quyết định tạo ra một máy chủ sẽ hoạt động như một proxy HTTP để "lừa" trình duyệt tin rằng Selenium Core và ứng dụng web được thử nghiệm đến từ cùng một tên miền. Hệ thống này được gọi là **Selenium Remote Control** hay **Selenium 1.**

**Sự ra đời của Selenium Grid**

Selenium Grid được phát triển bởi **Patrick Lightbody** để giải quyết nhu cầu giảm thiểu thời gian thực hiện thử nghiệm càng nhiều càng tốt. Lúc đầu ông gọi hệ thống là "**Hosted QA**.". Nó có khả năng chụp ảnh màn hình của trình duyệt trong các giai đoạn và cũng có thể **gửi ra các lệnh Selenium cho các máy khác nhau cùng một lúc**.

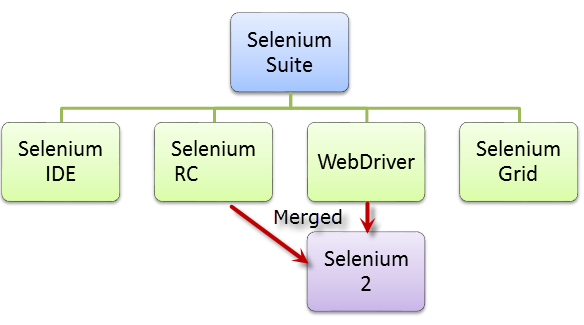
**Sự ra đời của Selenium IDE**

**Shinya Kasatani** của Nhật Bản đã tạo ra\*\* Selenium IDE\*\*, một phần mở rộng của Firefox có thể tự động hoá trình duyệt thông qua tính năng ghi lại và phát lại. Ông đã đưa ra ý tưởng này để tăng tốc độ trong việc tạo ra các trường hợp thử nghiệm. Ông đã tặng Selenium IDE cho dự án Selenium vào năm **2006**.

**Sự ra đời của Selenium 2**

Năm\*\* 2008\*\*, toàn bộ nhóm Selenium quyết định hợp nhất WebDriver và Selenium RC để tạo thành một công cụ mạnh hơn mang tên **Selenium 2**, với **WebDriver là nhân tố cốt lõi**. Hiện tại, Selenium RC vẫn đang được phát triển nhưng chỉ ở chế độ bảo trì. Hầu hết các nỗ lực của dự án Selenium hiện đang tập trung vào Selenium 2.

* 1. **Các phiên bản**



Selenium đã được tạo bởi Jason Huggins vào năm 2004. Trải qua lịch sử phát triển khá dài, cho đến hiện tại Selenium WebDriver 2 là bản mới nhất. Ban đầu Selenium Suite gồm Selenium IDE và Selenium RC, sau này 2 bản WebDriver và Selenium Grid mới được phát triển thêm.

* 1. **Tính năng chính**
* Có thể kết hợp với nhiều ngôn ngữ lập trình: Java, .Net, Ruby, Python, Perl
* Selenium giả lập thao tác người dùng trên web page và các web element.
* Có thể verify và so sánh thông tin trên web page
* Script đc base theo HTML, học dễ và ứng dụng cũng rất nhanh
* Có thể tạo một bộ test bao gồm nhiều test case
* Có thể run các test suite thông qua Selenium IDE hoặc Selenium command line
* Selenium API hỗ trợ nhiều loại trình duyệt, do đó sẽ rất dễ dàng khi test các ứng dụng web với các trình duyệt khác nhau.

Selenium Integrated Development Environment (IDE), là framework đơn giản nhất và dễ học nhất trong bộ Selenium. Nó là một plug-in chỉ dành cho trình duyệt FireFox – bạn chỉ có thể sử dụng Selenium IDE với trình duyệt FireFox mà thôi. Bạn có thể kết hợp Selenium IDE với các plug-in khác để tận dụng được nhiều tính năng hơn với IDE.

* 1. **Ưu – nhược điểm**
* **Ưu điểm**

1. Communicate trực tiếp với trình duyệt
2. Tương tác với trình duyệt giống như thao tác của một người dùng thật
3. Tốc độ nhanh hơn so với Selenium IDE
4. Thao tác dễ dàng hơn với các phép tính toán logic hay các điều kiện phức tạp

* **Nhược điểm**

1. Cài đặt phức tạp hơn so với Selenium IDE
2. Đòi hỏi người dùng phải có kĩ năng lập trình
   1. **So sánh với các công cụ khác**

# **Selenium và QTP**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tiêu chí** | **Selenium** | **QTP** |
| 1 | **Chi phí bản quyền** | Đây là công cụ mã nguồn mở cho nên không có chi phí bản quyền, hoàn toàn miễn phí | Rất mắc…khoảng 5000USD cho 1 máy |
| 2 | **Thực sự mô phỏng hành động của người dùng cuối** | Selenium thực hiện hành động trong nền của trình duyệt  tức là người sử dụng, có thể thực hiện các kiểm thử tự động với trình duyệt được thu nhỏ tối thiểu (minimized). | QTP thực hiện kiểm thử tự động tương đương với một người thực hiện các thao tác vào ứng dụng. Vì vậy, các trình duyệt không có thể được giảm thiểu (người sử dụng không thể thực hiện bất kỳ hành động nào khác trên máy tính trong khi kiểm thử đang chạy). |
| 3 | **Plug-ins** | Có và hoàn toàn miễn phí | Có. Người sử dụng phải trả tiền bản quyền |
| 4 | **Nỗ lực và kỹ năng để sử dụng công cụ** | Người dùng cần phải có kỹ năng tốt về lập trình và coding vì đây là điều kiện cần. | Đòi hỏi nỗ lực ít h ơn để tạo ra một kịch bản kiểm thử, vì nó có một môi trường phát triển rất tốt, thân thiện với người sử dụng. |
| 5 | **Dễ dàng tạo ra các kịch bản kiểm thử?** | Tốt. Selenium IDE Recorder không mạnh mẽ như QTP nhưng là một công cụ miễn phí, nhiều hành động không được ghi lại bởi các IDE và có thể tự nhập vào. | Hoàn hảo. |
| 6 | **Hỗ trợ trên các HĐH** | Windows, Linux, Unix và Macintosh. | Windows. |
| 7 | **Loại ứng dụng** | Web based applications. | Web-based và desktop applications. |
| 8 | **Trình duyệt** | IE, Firefox, Safari, Opera và một vài trình duyệt khác | IE, Chrome và Firefox. |
| 9 | **Ngôn ngữ phát triển kịch bản kiểm thử** | Java, Ruby, Python, C#, Perl, Groovy và nhiều ngôn ngữ khác | VBScript và/hoặc JavaScript. |
| 10 | **Tích hợp với các công cụ quản lý thử nghiệm** | Khi phát triển kiểm thử (test case) sử dụng C# trong VS thì có thể sử dụng MTM để quản lý test cases | HP Quality Center và TestDirector |
| 11 | **Môi trường phát triển kiểm thử** | Kịch bản thử nghiệm có thể được phát triển trong nhiều trình soạn thảo như Eclipse, Visual Studio, Netbeans… | Kịch bản thử nghiệm chỉ có thể được phát triển trong QTP. |
| 12 | **Hỗ trợ kỹ thuật** | Vì đây là công cụ m ã nguồn mở nên không có hỗ trợ kỹ thuật chính thức. | QTP cung cấp hỗ trợ kỹ thuật rất tốt qua điện thoại, mail, web, forum. |
| 13 | **Quản lý và lưu trữ đối tượng giao diện (UI-Object)** | Không. Đối tượng có thể được quản lý bằng cách sử dụng giao diện người dùng mở rộn g (UI-Element user extension) hoặc properties files. Tất cả các phương pháp trên đểu phải sử dụng code cứng (hard-code) | Có. QTP được xây dựng trong với Object Repository. Quản lý Object repository là khá dễ dàng trong QTP |
| 14 | **Hỗ trợ cho các hộp thoại** | Chỉ hỗ trợ 1 phần. Không thể thực hiện một số hành động như lấy tiêu đề của hộp thoại. | Có. QTP hỗ trợ tất cả các loại hộp thoại. |
| 15 | **Hỗ trợ cho các kịch bản tải lên tập tin** | Không. Selenium không hỗ trợ kịch bản này vì hạn chế JavaScript. | QTP không có vấn đề gì trong việc mô phỏng |

**Selenium vs Protractor**

* 1. **Hướng dẫn cài đặt, hướng dẫn sử dụng công cụ**

# Giới thiệu về phần mềm sẽ test