# **CHƯƠNG II: NGHIÊN CỨU CÔNG CỤ KIỂM THỬ SELENIUM WEBDIRVER**

## **2.1 Khái quát về kiểm thử tự động (Automation testing)**

### ***2.1.1 Khái quát kiểm thử tự động***

Trong kiểm thử phần mềm, kiểm thử tự động (automation testing) là việc sử dụng phần mềm đặc biệt (tách biệt với phần mềm đang được kiểm thử) để kiểm soát việc thực hiện các bài kiểm tra và so kết quả thực tế với kết quả dự đoán. Tự động kiểm thử có thể tự động hóa một số nhiệm vụ lặp đi lặp lại nhưng cần thiết trong một quá trình thử nghiệm đã được chính thức hóa, hay là các kiểm thử bổ sung nhưng sẽ khó thực hiện thủ công. Kiểm thử tự động là rất quan trọng cho phân phối liên tục và kiểm thử liên tục.

### ***2.1.2 Quy trình kiểm thử tự động***

Một quy trình kiểm thử tự động thường trải qua những bước cơ bản như sau:

Bước 1: Lựa chọn công cụ kiểm thử

Việc lựa chọn công cụ kiểm thử tự động phù hợp thường phụ thuộc vào Loại của Ứng dụng đang được kiểm tra (Application Under Test) và Môi trường mà ứng dụng đó sẽ chạy.

Hiện tại trên thị trường có đa dạng các loại công cụ kiểm thử tự động, từ miễn phí (Open Source, Freeware), đến có phí (License). Chẳng hạn Opensource có thể kể đến như: Selenium IDE, Selenium WebDriver, Appium, Cucumber, hay có License như: Postman, Katalon, akaAT, Ranorex, Tosca Tricentis, Testcomplete, HP-UFT.

Bước 2: Xác định phạm vi Kiểm thử tự động

Trong bước này, bạn sẽ xem xét khoanh vùng các luồng nghiệp vụ test phù hợp cho việc tự động hóa nó, chuẩn bị dữ liệu và môi trường diễn ra kiểm thử.

Dưới đây là một số yếu tố cần xem xét khi xác định phạm vi thử nghiệm tự động hóa của bạn:

* Các tính năng chính, quan trọng của ứng dụng
* Các trường hợp kiểm thử có nhiều dữ liệu
* Các tính năng dùng chung (common) trên ứng dụng
* Những vùng khả thi về kỹ thuật mà công cụ đáp ứng được
* Những nghiệp vụ hay được tái sử dụng
* Mức độ phức tạp của các test cases
* Khả năng sử dụng các trường hợp kiểm thử giống nhau để test trên nhiều trình duyệt (với trường hợp kiểm thử ứng dụng web)

Bước 3: Lập kế hoạch

Trong giai đoạn này trước hết bạn sẽ xây dựng bản Kế hoạch bao gồm:

* Thông tin lộ trình thực hiện (Master Schedule)
* Effort và thời gian cho việc thiết kế Framework
* Effort và thời gian cho việc viết kịch bản, Effort cho việc thực thi và báo cáo.
* Effort cho bảo trì sau phát triển
* Các rủi ro phát sinh

Bước 4 + Bước 5: Thiết kế và Phát triển

Ở giai đoạn này, các thành viên của nhóm sẽ bắt tay vào thiết kế các kịch bản tự động và dùng công cụ được chọn để tạo ra các script tự động hóa. Các công việc điển hình như:

* Thiết kế Framework và các tính năng của nó
* Thiết kế các kịch bản tự động
* Viết script và kiểm tra tính ổn định của script
* Review tính đúng đắn của script so với thiết kế

Bước 6: Thực thi kiểm thử (Execution)

Đây là giai đoạn áp chót trong quy trình kiểm thử tự động hóa. Khi bạn đã tạo xong các script tự động của mình, đã đến lúc chạy các script này để thực hiện kiểm thử ngay trên ứng dụng. Kết quả của việc thực thi các script thường sẽ được tổng hợp vào một báo cáo cho biết số lượng test cases PASS/FAIL và kèm theo các bằng chứng (Evidence) ghi lại hình ảnh/trạng thái tại thời điểm xảy ra các lỗi được tìm thấy.

Bước 7: Bảo trì

Khi phần mềm cần test (Application Under Test) được cập nhật thêm tính năng mới hoặc chỉnh sửa, thì bộ script sẽ được chạy để kiểm tra xem mức độ ảnh hưởng của các tính năng mới tới các tính năng hiện tại, liệu chúng có còn hoạt động đúng hay không. Và khi phần mềm có cập nhật, những thay đổi trên phần mềm có thể làm cho bộ script không còn đúng. Đó là lúc bạn cần có hoạt động xem xét chỉnh sửa, điều chỉnh lại script cho phù hợp với những thay đổi , cũng như thường xuyên nâng cấp nhằm nâng cao hiệu quả của các script tự động ở mỗi chu kỳ phát hành tiếp theo

### ***2.1.3 Ưu nhược điểm của kiểm thử tự động***

#### *2.1.3.1 Ưu điểm*

Tính hiệu quả trong công việc

Ưu điểm lớn nhất của kiểm thử tự động là thay thế con người lặp đi lặp lại đúng quy tắc các bước kiểm thử nhàm chán, không biết mệt mỏi, không cần xin phép đột xuất khi cần, không làm những việc ảnh hưởng đến quá trình test => tránh được hao phí về mặt thời gian.

Độ tin cậy

Với những script chuyên nghiệp, một khi đã Execute thành công và nhận về kết quả thì chúng ta có thể yên tâm nó chạy đúng với những gì đã yêu cầu. Dù lặp đi lặp lại nhiều lần vẫn cho ra kết quả giống nhau do vậy độ ổn định cao, tránh được rủi ro có thể phát sinh. Ngoài ra, kiểm thử tự động còn giúp tái hiện bug một cách hiệu quả, những bug thỉnh thoảng mới xảy ra hay xảy ra một cách ngẫu nhiên sẽ được repdure một cách dễ dàng nhờ việc lưu lại các bước một cách tự động.

Cải thiện chất lượng

Kiểm thử tự động làm giảm rủi ro về mặt chất lượng sản phẩm, việc kiểm thử được thực hiện một cách nhanh chóng, không gây mệt mỏi, tránh trường hợp chủ quan cho những trường hợp test hồi quy, retest, test toàn bộ hệ thống. Có thể tái sử dụng các trường hợp kiểm thử.

Nếu bạn mất 5 phút để kiểm thử thủ công thì chỉ cần mất 30s nếu sử dụng kiểm thử tự động, công việc của bạn là "Run" và ngồi chờ kết quả, nó giúp bạn chạy liên tiếp được nhiều case.

Chi phí thấp

Việc rút ngắn thời gian và tiết kiệm nhân lực giúp cho việc kiểm thử tự động trở nên hiệu quả. Hỗ trợ export ra nhiều ngôn ngữ phổ biến như Ruby, Java, Python…

#### *2.1.3.2 Nhược điểm*

Ban đầu thì chi phí cho kiểm thử tự động sẽ cao hơn kiểm thử thủ công.

Để kiểm thử tự động thực hiện được thì vẫn cần con người phải bỏ thời gian, công sức và tiền bạc,...

Mất chi phí cho các công cụ tự động hóa như bản quyền, bảo trì, tìm hiểu, training.

Khó mở rộng hơn nhiều so với kiểm thử thủ công.

Yêu cầu những người có trình độ chuyên môn cao mới thực hiện được.

Số lượng công việc phải làm để mở rộng cho kiểm thử tự động sẽ nhiều và khó hơn so với kiểm thử thủ công. Để kiểm thử thủ công chỉ cần mở, gõ,... rất đơn giản. Nhưng kiểm thử tự động cập nhật, chỉnh sửa yêu cầu rất nhiều công việc như debug, thay dữ liệu đầu vào, cập nhật lại code mới.

### ***2.1.4 Một số công cụ kiểm thử tự động phổ biến***

#### *2.1.4.1 Selenium*

Khái niệm: Nó là một công cụ kiểm tra phần mềm được sử dụng để kiểm tra hồi quy( Regression Testing). Đây là một công cụ kiểm tra mã nguồn mở cung cấp chức năng phát lại và thu âm để kiểm tra hồi quy. Các Selenium IDE chỉ hỗ trợ trình duyệt web Mozilla Firefox.

Đặc điểm của Selenium:

* Cung cấp các điều khoản để xuất khẩu ghi lại kịch bản trong các ngôn ngữ khác như Java, Ruby, RSpec, Python, C #, JUnit và TestNG
* Nó có thể thực hiện nhiều bộ kiểm thử cùng một lúc
* Xác định phần tử sử dụng id, tên, đường dẫn X, v.v.
* Lưu trữ các bộ kiểm thử như Ruby Script, HTML và bất kỳ định dạng nào khác + Hỗ trợ tệp tin người dùng selenium-extensions.js
* Cho phép để chèn ý kiến ở giữa của kịch bản để hiểu rõ hơn nội dung và mục đích của kịch bản kiểm thử

#### *2.1.4.2 QTP (HP UFT)*

Khái niệm: QTP được sử dụng rộng rãi để kiểm tra chức năng( Functional Testing) và hồi quy( Regression Testing), giải quyết các ứng dụng phần mềm và môi trường. Để đơn giản hóa việc tạo và bảo trì thử nghiệm, nó sử dụng khái niệm kiểm tra từ khóa.

Đặc điểm:

* Được sử dụng dễ dàng hơn dành cho người kiểm thử viên không theo ngành kỹ thuật để thích ứng và tạo ra các trường hợp thử nghiệm làm việc
* Sửa lỗi nhanh hơn bằng cách ghi lại và sao chép các lỗi cho nhà phát triển
* Thu gọn tài liệu thử nghiệm tại một trang web
* QTP hỗ trợ môi trường phát triển .NET
* Có cơ chế xác định đối tượng kiểm thử tốt

#### *2.1.4.3 Rational Function Tester*

Khái niệm: Là 1 công cụ kiểm tra tự động hướng đối tượng có khả năng tự động kiểm tra dữ liệu, kiểm tra giao diện, và kiểm thử hồi quy( Regression Testing)

Đặc điểm:

* Hỗ trợ một loạt các giao thức và ứng dụng như Java, HTML, NET, Windows, SAP, Visual Basic ...
* Có thể ghi lại và phát lại các hành động theo yêu cầu
* Tích hợp tốt với các công cụ quản lý kiểm soát nguồn như Rational Clear Case và tích hợp Rational Team Concert
* Cho phép các nhà phát triển tạo ra các kịch bản liên quan đến từ khóa để có thể được tái sử dụng
* Bộ biên tập Công cụ Java Developer Toolkit của Eclipse tạo điều kiện cho nhóm tạo mã thử nghiệm các đoạn mã trong Java với Eclipse
* Hỗ trợ điều khiển tùy chỉnh thông qua proxy SDK (Java / .Net)
* Hỗ trợ kiểm soát phiên bản để cho phép phát triển song song các kịch bản thử nghiệm

#### *2.1.4.4 WATIR*

Khái niệm: Là một phần mềm kiểm tra mã nguồn mở để kiểm thử hồi quy( Regression Testing). Watir chỉ hỗ trợ khám phá Internet trên các cửa sổ trong khi Watir webdriver hỗ trợ Chrome, Firefox, IE, Opera,

Đặc điểm:

* Hỗ trợ nhiều trình duyệt trên các nền tảng khác nhau
* Sử dụng một ngôn ngữ kịch bản hiện đại có đầy đủ tính năng
* Hỗ trợ ứng dụng web được viết bởi bất kỳ ngôn ngữ nào
* Cho phép bạn viết các test case dễ đọc và bảo trì

#### *2.1.4.5 SilkTest*

Khái niệm: SilkTest được thiết kế để thực hiện kiểm tra chức năng( Functional Testing) và hồi quy( Regression Testing). Nó là một ngôn ngữ hướng đối tượng giống như C ++. Nó sử dụng các khái niệm về đối tượng, các class và sự kế thừa.

Đặc điểm:

* Nó bao gồm tất cả các tập tin mã nguồn
* Chuyển đổi các lệnh script thành các lệnh GUI. Trên cùng một máy, các lệnh có thể được chạy trên một máy từ xa hoặc máy chủ
* Để xác định chuyển động của con chuột cùng với các bấm phím, Silktest có thể được thực hiện. Nó có thể sử dụng cả phương pháp phát lại và ghi hoặc các phương pháp lập trình mô tả.

### ***2.1.5 Các trường hợp nên áp dụng kiểm thử tự động***

Việc cân nhắc liệu "kiểm thử tự động" có phải là hướng đi phù hợp cho nhóm dự án của bạn hay không là rất quan trọng. Vì không phải tất cả các nhóm QA đều cần tự động hóa để đẩy nhanh quá trình thử nghiệm của họ. Kiểm thử thủ công vẫn đóng 1 vai trò rất quan trọng trong quy trình sản xuất phần mềm/sản phẩm. Vì vậy, chỉ thực hiện kiểm thử tự động khi:

* Dự án có nhiều trường hợp kiểm thử lặp đi lặp lại cần thực hiện
* Khi có các bài kiểm tra hồi quy thường xuyên
* Khi nhóm dự án phải mô phỏng một số lượng lớn người dùng để kiểm tra hiệu suất
* Khi giao diện người dùng ổn định
* Khi các chức năng quan trọng không thể chỉ dựa vào thử nghiệm thủ công