**Tìm hiểu các khái niệm về Test**

# Unit Test

## Khái niệm

UT là kỹ thuật kiểm nghiệm các hoạt động của mọi chi tiết mã (code) với một quy trình tách biệt với quy trình phát triển PM, giúp phát hiện sai sót kịp thời. UT còn có thể giúp phát hiện các vấn đề tiềm ẩn và các lỗi thời gian thực ngay cả trước khi chuyên viên kiểm định chất lượng (QA - Quality Assurance) tìm ra, thậm chí có thể sửa lỗi ngay từ ý tưởng thiết kế.

UT là các đoạn mã có cấu trúc giống như các đối tượng được xây dựng để kiểm tra từng bộ phận trong hệ thống. Mỗi UT sẽ gửi đi một thông điệp và kiểm tra câu trả lời nhận được đúng hay không, bao gồm:

* Các kết quả trả về mong muốn
* Các lỗi ngoại lệ mong muốn

Các đoạn mã UT hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó UT còn được gọi là kỹ thuật kiểm nghiệm tự động.

UT có các đặc điểm sau:

* Đóng vai trò như những người sử dụng đầu tiên của hệ thống.
* Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

## Vòng đời của Unit Test

UT có 3 trạng thái cơ bản:

* Fail (trạng thái lỗi)
* Ignore (tạm ngừng thực hiện)
* Pass (trạng thái làm việc)

Toàn bộ UT được vận hành trong một hệ thống tách biệt. Có rất nhiều PM hỗ trợ thực thi UT với giao diện trực quan. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail).

UT chỉ thực sự đem lại hiệu quả khi:

* Được vận hành lặp lại nhiều lần
* Tự động hoàn toàn
* Độc lập với các UT khác.

## Thiết kế

Mỗi UT đều được thiết kế theo trình tự sau:

* Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…
* Triệu gọi các phương thức cần kiểm tra.
* Kiểm tra sự hoạt động đúng đắn của các phương thức.
* Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

## Ứng dụng

* Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.
* Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.
* Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc(workflow – tập hợp của nhiều quy trình)

## Lợi ích

Thời gian đầu, người ta thường do dự khi phải viết UT thay vì tập trung vào code cho các chức năng nghiệp vụ. Công việc viết Unit Test có thể mất nhiều thời gian hơn code rất nhiều nhưng lại có lợi ích sau:

* Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn code nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.
* Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
* Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
* Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
* Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.

Trong môi trường làm việc Unit Test còn có tác dụng rất lớn đến năng suất làm việc:

* Giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.
* Tăng sự tự tin khi hoàn thành một công việc. Chúng ta thường có cảm giác không chắc chắn về các đoạn mã của mình như liệu các lỗi có quay lại không, hoạt động của module hiện hành có bị tác động không, hoặc liệu công việc hiệu chỉnh mã có gây hư hỏng đâu đó…
* Là công cụ đánh giá năng lực của bạn. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái “pass” sẽ thể hiện tốc độ làm việc, năng suất của bạn.

# Acceptance Test

## Khái niệm

* Đây là một kiểm thử liên quan đến nhu cầu của người sử dụng, yêu cầu và quy trình kinh doanh được tiến hành để xác định có hay không một hệ thống đáp ứng các tiêu chí chấp nhận và kiểm tra hệ thống đáp ứng yêu cầu của khách hàng.
* Kiểm thử chấp nhận kiểm thử các chức năng để kiểm tra hành vi của hệ thống bằng cách sử dụng dữ liệu thực tế. Nó cũng được gọi là thử nghiệm người dùng doanh nghiệp.
* Kiểm thử chấp nhận được thực hiện bởi người dùng cuối để kiểm tra hệ thống được xây dựng để phù hợp với yêu cầu kinh doanh của tổ chức.
* Trong kiểm thử này, tất cả các giao diện đã được kết hợp và hệ thống đã hoàn thành và đã được kiểm tra. Người dùng cuối cũng thực hiện các kiểm thử để kiểm tra khả năng sử dụng của hệ thống.
* Nhiều kỹ thuật kiểm thử chức năng sử dụng cho loại này là phân tích giá trị biên giới, phân vùng tương đương, bảng quyết định. Đây là loại kiểm tra tập trung chủ yếu vào các kiểm thử hợp lệ của hệ thống.

## Phân loại

Trong kiểm thử chấp nhận có hai mức độ kiểm thử là Alpha testing và Beta testing:

* Alpha Testing: Kiểm thử Alpha cũng được gọi là kiểm thử trang web off. Trong kiểm thử này quá trình kiểm thử sẽ kiểm tra các ứng dụng với sự hiện diện của người dùng cuối trong môi trường tổ chức.
* Beta Testing: Kiểm thử Beta nên được thực hiện bởi người dùng cuối trong môi trường riêng của họ với sự hiện diện của đội phát hành.

## Thuộc tính của Acceptance Test

* Kiểm thử chấp nhận là kiểm thử chức năng, được thực hiện để kiểm tra xem phần mềm đáp ứng các yêu cầu của khách hàng
* Kiểm thử chấp nhận được thực hiện bởi khách hàng, người dùng và các bên liên quan.
* Kiểm thử chấp nhận kiểm tra các yêu cầu chức năng
* Trong kiểm thử chấp nhận sẽ kiểm tra hệ thống đáp ứng các nhu cầu kinh doanh của tổ chức, khả năng sử dụng của sản phẩm
* Được thực hiện với các dữ liệu thời gian thực tế.
* Kiểm thử phần mềm cho các nhu cầu sử dụng và nhu cầu của người sử dụng được đáp ứng trong phát triển phần mềm.
* Kiểm thử chấp nhận bao gồm kiểm thử alpha và kiểm thử beta.
* Kiểm thử chấp nhận được thực hiện sau kiểm thử hệ thống. 9. Kiểm thử chấp nhận liên quan đến kiểm thử chức năng đó là phân tích giá trị biên, phân vùng tương đương và bảng quyết định.
* Kiểm thử chấp nhận chứa nhiều các trường hợp kiểm thử thông thường (normal test cases) .
* Các lỗi tìm thấy trong kiểm thử chấp nhận được xem như là sự thất bại của sản phẩm.
* Kiểm thử với các dữ liệu ngẫu nhiên

# Automation Test

## Khái niệm

Trong lĩnh vực kiểm thử phần mềm, thì kiểm thử tự động hay còn gọi là Automation testing đóng một vai trò quan trọng góp phần nâng cao năng suất kiểm thử, giảm thiểu lỗi cũng như sự nhàm chán với việc kiểm thử bằng tay trong một thời gian dài hoặc lặp đi lặp lại.

Kiểm thử tự động là một quá trình xử lý tự động các bước thực hiện một test case. Kiểm thử tự động được thực hiện bởi phần mềm kiểm thử tự động - hay còn gọi là Automation Testing Tool. Một số phần mềm kiểm thử tự động nổi tiếng hiện nay như:

* Quick Test Profressional - (HP)
* Selenium
* Test Architect - (LogiGear)
* Ranorex
* Visual Studio CodedUI Testing
* TestComplete (SmartBear)
* SOAPUI - Web Services Testing (SmartBear)

## Ưu điểm và nhược điểm

* Ưu điểm
* Độ tin cậy cao: công cụ kiểm thử tự động có sự ổn định cao hơn so với con người, đặc biệt trong trường hợp nhiều test cases, nên độ tin cậy cao hơn so với kiểm thử thủ công.
* Khả năng lặp: công cụ kiểm thử tự động ra đời là để giúp cho các tester không phải lặp đi lặp lại các thao tác (ví dụ: nhập dữ liệu, click, check kết quả…) một cách nhàm chán với độ tin cậy và ổn định cao.
* Khả năng tái sử dụng: với một bộ kiểm thử tự động, người ta có thể sử dụng cho nhiều phiên bản ứng dụng khác nhau, đây được gọi là tính tái sử dụng.
* Tốc độ cao: do thực thi bởi máy nên tốc độ của kiểm thử tự động nhanh hơn nhiều so với tốc độ của con người. Nếu cần 5 phú để thực thi một test case một cách thủ công thì có thể người ta chỉ cần khoảng 30s để thực thi một cách tự động.
* Chi phí thấp: nếu áp dụng kiểm thử tự động đúng cách, người ta có thể tiết kiệm được nhiều chi phí, thời gian và nhân lực, do kiểm thử tự động nhanh hơn nhiều so với kiểm thử thủ công, đồng thời nhân lực cần để thực thi và bảo trì scripts không nhiều.
* Nhược điểm:
* Khó mở rộng, khó bảo trì: trong cùng một dự án, để mở rộng phạm vi cho kiểm thử tự động khó hơn nhiều so với kiểm thử thủ công vì cập nhật hay chỉnh sửa yêu cầu nhiều công việc như debug, thay đổi dữ liệu đầu vào và cập nhật code mới.
* Khả năng bao phủ thấp: do khó mở rộng và đòi hỏi nhiều kỹ năng lập trình nên độ bao phủ của kiểm thử tự động thấp xét trên góc nhìn toàn dự án.
* Vấn đề công cụ và nhân lực: hiện nay cũng có nhiều công cụ hỗ trợ kiểm thử tự động khá tốt nhưng chúng vẫn còn nhiều hạn chế. Ngoài ra nhân lực đạt yêu cầu (có thể sử dụng thành thạo các công cụ này) cũng không nhiều.

## Kiến thức cần có

Để trở thành một Automation Tester, trước tiên, bạn cần có những kiến thức nằm lòng về testing:

* Types testing: Unit/Intergration/System/Sanity/Regression test/.... là gì?
* Testing Techniques: Phân tích giá trị biên/Phân vùng tương đương/Biểu đồ kết quả/Đoán lỗi/... là gì?
* Nắm rõ về HTML và XPath để nhận dạng đúng test objects/elements mà mình cần thao tác.
* Thành thạo ít nhất một ngôn ngữ lập trình thông dụng, hỗ trợ cho việc scripting trên test tools: Java/C#/Python/Ruby/JS/...
* Sử dụng thành thạo thư viện của Selenium WebDriver API bởi Selenium open source, dễ sử dụng, cộng đồng lớn.
* Sử dụng thành thạo ít nhất 1 framework testing: Junit/TestNG/NUnit/... Từ đây sẽ giúp bạn rất nhiều trong việc build framework, hỗ trợ trong việc phân nhóm, quản lý testscript, report, prepare data/environment/browsers.
* Tìm hiểu về software design pattern để build framework/common library
* Sử dụng/build framework thành thạo từ Page Object Model pattern.
* Kỹ năng về coding/IDE: Debug, coding convension, source version control (GIT, SVN,...), cách sử dụng IDE: Visual Studio, Eclipse, IntelliJ..., làm việc với database...
* Học hỏi công nghệ mới trong mảng automation testing: build tools: Maven, ANT..., CI/CD: Jenkins, TeamCity, CircleCI, TFS, Docker.., Clould: AWS, Saucelab, Browserstack, Testingbot..., big data: Hadoop, HBase, Hive..., mobile: Appinum...

## Tầm quan trọng của Automation Test

Cách mạng công nghiệp lần thứ tư đánh dấu kỷ nguyên vạn vật kết nối Internet. Nó xảy ra dựa trên sự hội tụ của nhiều công nghệ trong đó có công nghệ cốt lõi có công nghệ thông tin với sự phát triển không ngừng của công nghệ Internet từ thời kỳ kết nối nội dung như email đến mạng xã hội, Internet vạn vật, Internet kết nối thiết bị máy móc kết nối quá trình vận hành của các nhà máy. Ngoài công nghệ cốt lõi còn có sự hội tụ của công nghệ in 3D, công nghệ vật liệu tiên tiến, công nghệ lưu trữ…

Hiện nay, nền công nghiệp 4.0 đang phát triển mạnh mẽ, kéo theo sự phát triển của rất nhiều ngành nghề, đặc biệt là ngành công nghệ thông tin. Do đó, yêu cầu về nhân lực trong mảng này cũng đòi hỏi cac ứng viên cần phải có kỹ năng tốt hơn nữa về lập trình, technical, các kỹ năng về automation....

Nếu search trên các trang mạng tuyển dụng lớn như ITviec, Vietnamworks, LinkIn, Indeed, Dice, Monster, CareerBuilder … thì bạn sẽ thấy có vô vàn kết quả với các từ khóa tìm kiếm như:

* Test Automation Engineer
* Automation Developer
* Automation Testing/Automation Tester
* QA Automation Engineer
* Software Development Engineer in Test (SDET)

Từ đây, bạn có thể thấy được nhu cầu tuyển một QA có kiến thức và kinh nghiệm về automation test là rất lớn. Do đó, đủ để hiểu automation test sẽ là tiềm năng lớn cho tương lai của mảng Test nói riêng và công nghệ thông tin nói chung.

# Stability Test

## Khái niệm

Stability testing là khả năng duy trì hoạt động của sản phẩm xuyên suốt và vượt quá thời hạn sử dụng của nó, mà không hỏng hoặc xảy ra lỗi. Đây là 1 kỹ thuật non-functional, với mục đích đòi hỏi khả năng chịu tải của phần mềm tới mức tối đa. Trong quá trình xác định nó hoạt động tốt thế nào dưới tải ở mức chấp nhận được, mức đỉnh, các tải được tạo ra đột ngột, với số lượng dữ liệu lớn được xử lý… Stability testing được thực hiện để kiểm tra hiệu quả của 1 sản phẩm được phát triển vượt qua mức hoạt động bình thường, hay tới 1 điểm dừng. Có ý nghĩa quan trọng hơn là trong việc xử lý lỗi, độ tin cậy của phần mềm, khả năng chịu tải và khả năng mở rộng của 1 sản phẩm dưới tải lớn chứ không phải là kiểm tra cách hoạt động của hệ thống trong các hoàn cảnh bình thường.

## Tại sao phải thực hiện Stability Test

Loại testing này giúp user hiểu được cách thức hệ thống sẽ làm việc trong các tình huống thực tế.

Do đó, Stability testing cho phép bạn kiểm tra:

* Ước lượng độ tin cậy trong sự ổn định của hệ thống của bạn đang thử nghiệm
* Đảm bảo rằng hệ thống của bạn có thể xử lý các chương trình lớn
* Theo dõi hiệu quả của hệ thống của bạn
* Kiểm tra hệ thống ổn định theo yêu cầu

Nó đóng 1 vai trò quan trọng trong phát triển sản phẩm vì nó được sử dụng để xác định những hạn chế của 1 sản phẩm phần mềm đang thử nghiệm trước khi nó được released hoặc những phạm vi cải tiến hơn trước khi sản phẩm goes live hay ở Production. 1 ví dụ rất phổ biến của kỹ thuật Stability testing là

Các cổng mua sắm trực tuyến: Stability testing sẽ kiểm tra cách các website sẽ hoạt động khi:

* Lượng dữ liệu nhập vào lớn ở giờ cao điểm
* Số lượt truy cập ở 1 thời điểm nhất định
* Vấn đề page load ở cùng 1 thời điểm
* Cách vận hành của hệ thống
* Phản ứng của hệ thống và nhiều hơn nữa buộc phải trải nghiệm dưới Stability testing.

## Phương pháp tiếp cận cơ bản cho chiến lược test

* Để xác định phạm vi và mục tiêu của việc test, chúng ta buộc phải đảm bảo rằng các Application Server(s) không bị crash trong quá trình thực hiện Load test.
* Để xác định các Business issue, kiểm tra hiệu năng và tải của hệ thống đối với end user.
* Để gán các trách nhiệm và các vai trò khác nhau như – tạo các test plan, thiết kế và review các test case, thực hiện test …
* Để đảm bảo sự phân phối test trong thời gian quy định
* Để đảm bảo các tool load testing và experience team thích hợp hiện diện cùng lúc
* Để ước tính các rủi ro và chi phí liên quan trong việc testing. Điều này sẽ xác định chi phí cho mỗi lần thực hiện trong việc sử dụng CPU và bộ nhớ
* Xác định việc theo dõi và báo cáo lỗi và có phác đồ thích hợp với các yêu cầu

## Các test case cho Stability Testing đối với CPU performance

* Để xác minh giới hạn trên của hệ thống
* Cách hệ thống crash hay recover
* Tổng số transaction hoàn thành mỗi yêu cầu
* Các giao dịch phản hồi có ổn định hay không hay tăng theo thời gian
* Làm thế nào hệ thống hoạt động dưới tải nặng
* Phản ứng và cách vận hành của nó dưới sức tải nặng

## Các test report cho Stability Test

Một số số liệu thống kê được thu thập và đánh giá trong quá trình thực hiện test; những con số này được phân tích để tạo ra 1 report và để xác định các vấn đề về hiệu năng.

Các ví dụ về số liệu thống kê được thu thập dưới việc test là:

* Transaction response time: Thời gian trung bình thực hiện các giao dịch trong quá trình test. Thống kê này sẽ đánh giá liệu hiệu năng của server nằm trong khoảng thời gian thực hiện các giao dịch tối thiểu và tối đa có thể chấp nhận được định nghĩa cho hệ thống hay không. Thông tin này sẽ đánh giá thời gian xử lý yêu cầu của web server và gửi tới application server, cái mà trong hầu hết các trường hợp sẽ tạo 1 yêu cầu tới database server.
* Hits per second: Số lượt truy cập thực hiện trên server bởi các user. Thống kê này có lợi cho việc xác định lượng tải các user tạo ra, liên quan đến 1 số lần truy cập.
* Throughput: Thông lượng trên web server trong quá trình test được đo bằng byte. Thông lượng là lượng dữ liệu mà các user nhận được từ server tại bất kỳ thời điểm nào. Thống kê này giúp đánh giá lượng tải mà các user tạo ra.
* Transaction per second: Đây là tổng số các giao dịch hoàn thành (cả thành công và thất bại) thực hiện trong quá trình test. Thống kê này giúp kiểm tra tải thực sự của giao dịch trên hệ thống.
* CPU: % CPU sử dụng dành ra trong quá trình test.
* Memory: Bộ nhớ sử dụng trong quá trình test
* Disk: Không gian sử dụng của ổ đĩa dành ra trong quá trình test

Lúc này, bạn có thể dễ dàng nhận ra rằng:

Stability testing đi kèm theo performance testing – một kỹ thuật mà được thực hiện để kiểm tra 1 số thuộc tính chất lượng của phần mềm như tính ổn định, độ tin cậy và tính khả dụng.

Việc test này được sử dụng để xác định mức độ nhanh chóng của 1 hệ thống hoặc 1 hệ thống phụ thực thi dưới 1 khối lượng công việc cụ thể.

Performance testing có nhiều loại và Stability testing là 1 trong số chúng.

* Stress testing: nó là 1 loại test mà kiểm tra sức chịu tải của hệ thống vượt qua khả năng của hệ thống.
* Spike testing: nó được sử dụng để kiểm tra hành vi của 1 hệ thống bằng cách tăng tải của 1 hệ thống ngay lập tức. Mục đích là để kiểm tra tại thời điểm nào hệ thống sẽ có vấn đề về hiệu năng, hay nó sẽ pass.
* Scalability testing: Nó được sử dụng để kiểm tra khả năng của 1 hệ thống. Hệ thống sẽ có hiệu lực hành xử thế nào trong như cầu ngày càng tăng, thay đổi về quy mô và khối lượng.
* Volume testing: Nó là 1 kỹ thuật test non-functional nơi mà phần mềm đang thử nghiệm phải chịu khối lượng công việc khổng lồ, và hành vi của 1 hệ thống được kiểm tra và xác minh phù hợp
* Load or stability testing: đã được mô tả bên trên.

1 vài công cụ cho performance testing như:

* WebLOAD
* LoadRunner
* Apache JMeter
* NeoLoad
* CloudTest
* Loadstorm
* LoadUI
* WAPT
* LoadImpact
* Loadster
* Httperf
* OpenSTA
* Performance Tester
* Testing Anywhere