1. **Vài định nghĩa về kiểm thử phần mềm**

* Kiểm thử phần mềm là một quá trình mà hành vi được kiểm tra và quan sát đối với một hành vi mong đợi cụ thể
* Kiểm thử phần mềm là một cuộc điều tra tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của sản phẩm hoặc dịch vụ được thử nghiệm
* Testing là quá trình thực thi một chương trình với mục đích tìm ra lỗi.

1. **Levels of Test**

* **Unit Testing**:

Test ở mức độ thấp như các hàm, module.

* **Integration Testing:**

Tester sẽ tiến hành test giao diện, sự tương tác giữa các components, modules, window.

* **Functional Testing:**

Test ở bất kỳ mức độ nào(lớp, module, giao diện, hệ thống) để kiểm tra xem có đúng với đặc tả hay không?

* **System Testing:**

Test hệ thống một cách toàn diện 🡪 đáp ứng được yêu cầu?

* **System Integration Testing:**

Hệ thống tích hợp đúng với các phần mềm của hãng thứ ba hay các giao diện của hệ thống khác

* **Performance Testing:**

Hiệu năng chương trình có đáp ứng được mong đợi

* **Acceptance Testing:**

Test được thực hiện bởi khách hàng hay người dùng cuối.

1. **Mục đích của Testing**

* Cung cấp thông tin về chất lượng sản phẩm
* Tìm các bug quan trọng (nghiêm trọng) 🡪 fix
* Ước định chất lượng sản phẩm phầm mềm
* Giúp quản lý ra quyết định release hay không?
* Hạn chế release các sản phẩm không đủ chất lượng
* Hỗ trợ dự đoán, điều khiển giá sản phẩm
* Tìm các ngữ cảnh an toàn cho việc sử dụng
* Đảm bảo sản phẩm phù hợp với các chuẩn

1. **Tầm quan trọng trong kiểm thử phần mềm**

* Đảm bảo rằng các sản phẩm là có thể sử dụng
* Đảm bảo rằng các mục tiêu của khách hàng được đáp ứng
* Phát hiện sớm các lỗi để ngăn ngừa sự cố ở giai đoạn sau
* Đảm bảo rằng phần mềm là đáng tin cậy
* Xây dựng niềm tin trong phần mềm
* Tăng sự hài lòng của khách hàng
* Đảm bảo thực hiện hiệu quả trong môi trường nhất định
* Giảm chi phí tổng thể của phần mềm

1. **Một số điểm lưu ý khi kiểm thử**

* Chất lượng phần mềm do
  + Khâu thiết kế quyết định là chủ yếu
  + Chứ không phải khâu kiểm thử
* Tính dễ kiểm thử phụ thuộc vào cấu trúc chương trình
* Người kiểm thử và người phát triển nên khác nhau
* Dữ liệu thử cho kết quả bình thường
  + Không có ý nghĩa nhiều
  + Cần có những dữ liệu kiểm thử mà phát hiện ra lỗi
* Khi thiết kế *testcase*
  + Thiết kế dữ liệu kiểm thử nhập vào
  + Thiết kế trước cả dữ liệu kết quả sẽ có
* Khi phát sinh thêm *testcase*
  + Thử lại những *testcase* trước đó
  + Để tránh ảnh hưởng lan truyền sóng

1. **Làm nghề tester**

* Testing từng được xem là nhiệm vụ phụ trợ cho lập trình
* Hiện nay testing là 1 nghề thực sự, yêu cầu:
  + Kiến thức
  + Được huấn luyện
  + Kinh nghiệm chuyên biệt
* Chất lượng PM ngày càng được chú trọng

🡪 Lương ngày càng cao

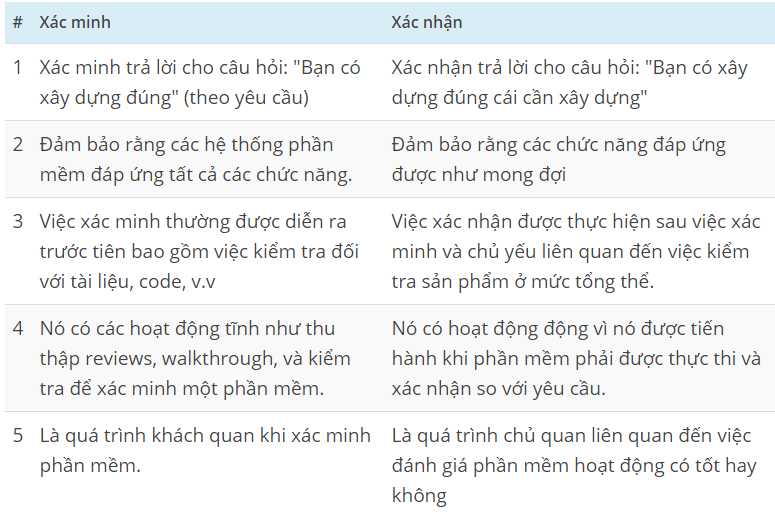
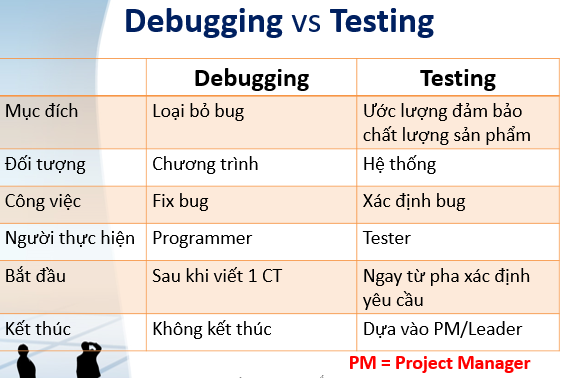
1. **Kỹ năng của Tester**

* Phân tích tốt, giao tiếp tốt, có nền tảng về QA
* Kinh nghiệm về:
  + Phát triển Test plan
  + Triển khai môi trường test
  + Hiểu các chuẩn test
  + Tài liệu test
* Kinh nghiệm trong các mức test
* Có kiến thức thực tiễn (ngân hàng, tài chính, truyền thông,…)
* Có kinh nghiệm về các công cụ test
* Kỹ năng về môi trường (phần cứng, hđh)
* Kỹ năng về ứng dụng (Office, Database, Programing)

1. **Phẩm chất của Tester**

* Có kế hoạch, hệ thống
* Kiên nhẫn, không từ bỏ dễ dàng
* Thực tế: cân bằng hiệu quả, thời gian, chi phí
* Đầu óc phân tích 🡪 tiếp cận đào sâu lỗi
* Đạo đức nghề nghiệp: đưa ra sản phẩm có chất lượng tốt
* Một số phẩm chất khác, ảnh hưởng trái chiều
  + Bảo thủ
  + Hoài nghi
  + Thiếu khéo léo
  + Đồng cảm với programmer

**Kiểm thử phần mềm là 1 thành phần trong lĩnh vực rộng hơn, ₫ó là Verification & Validation, ta tạm dịch là xác minh, xác nhận**

1. **Sự khác nhau giữa xác minh và xác nhận**
2. **Debugging vs Testing**
3. **Fault, Defect, Bug, Error**

* Đều là lỗi phần mềm cả
* Nhưng dùng ở ngữ cảnh khác nhau
* **Fault, Error**: kết quả không phù hợp theo đặc tả, giữa thực tế và mong đợi trên lý thuyết
* **Bug**: Lỗi phát sinh trong quá trình viết mã
* **Defect**: Lỗi phát sinh sau khi sản phẩm đã được chuyển giao cho khách hàng

1. **Code coverage**

* **Coverage** (độ bao phủ): phép đo xác định nỗ lực của tester trong việc phát các hiện lỗi tiềm tàng.
  + **100% line coverage**: đã test cho tất cả các bug mà có thể bị phát hiện bởi thực thi một dòng code đơn giản
  + **100% branch coverage**: tìm tất cả các lỗi mà có thể phát hiện trên từng nhánh
* **100% path coverage**: đã test cho tất cả các bug mà có thể xảy ra **(impossible)**

1. **Usercase**

* **Use case là** một kỹ thuật được dùng trong kỹ thuật phần mềm và hệ thống để nắm bắt yêu cầu chức năng của hệ thống. **Use case** mô tả sự tương tác đặc trưng giữa người dùng bên ngoài (actor) và hệ thống. Nó thể hiện ứng xử của hệ thống đối với bên ngoài, trong một hoàn cảnh nhất định, xét từ quan điểm của người sử dụng.

1. **Test Case**

* Test case mô tả một dữ liệu đầu vào (input), hành động (action) hoặc sự kiện (event) và một kết quả mong đợi (expected response), để xác định một chức năng của ứng dụng phần mềm hoạt động đúng hay không. Một test case có thể có các phần đặc thù khác nhau như mã test case, tên test case, mục tiêu test, các điều kiện test, các yêu cầu data input, các bước thực hiện và các kết quả mong đợi.
* .Testcase hiểu nôm na là một tình huống kiểm tra, được thiết kế để kiểm tra một đối tượng có thỏa mãn yêu cầu đặt ra hay không.  
  Nó bao gồm 3 bước cơ bản :  
  - Mô tả : đặc tả các điều kiện cần cố để tiến hành kiểm tra.  
  - Nhập : đặc tả đối tượng hoặc dữ liệu cần thiết, được sử dụng làm đầu vào để thực hiện kiểm tra.  
  - Kết quả mong chờ : kết quả trả về từ đối tượng kiểm tra.

1. **Test scripts**

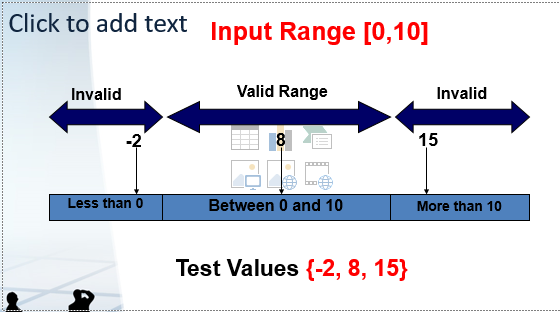
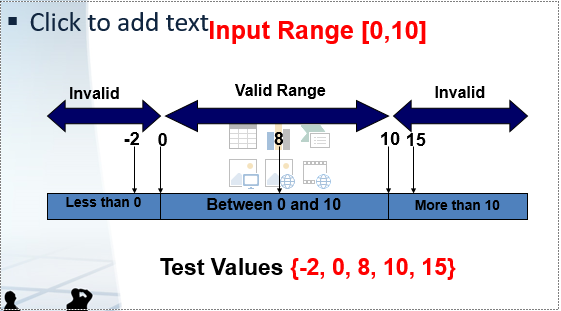
* Một Test Script là một nhóm mã lệnh dạng đặc tả kịch bản dùng để tự động hóa một trình tự kiểm tra, giúp cho việc kiểm tra nhanh hơn, hoặc cho những trường hợp mà kiểm tra bằng tay sẽ rất khó khăn hoặc không khả thi. Các Test Script có thể tạo thủ công hoặc tạo tự động dùng công cụ kiểm tra tự động.
* Chú ý: Một test script có thể gồm nhiều test case.

1. **Test Suite**

* Một tập hợp các test script hoặc test case
* Để đánh giá các bản vá lỗi
* Hoặc tìm lỗi mới
* một tập hợp các trường hợp thử nghiệm thiết kế để thực hiện và xác nhận một **business process**
* **Test requirement**: tài liệu mô tả các hạng mục, chức năng cần được test.
* **Test plan**: Một tài liệu phát thảo các thông tin tổng quan về test
* **Test strategy**: đề xuất cách tiếp cận test, cân bằng lợi ích và rủi ro.

-----------------------------------------------------------------------------------------------------------------------

1. **Các loại test:** 
   1. **Static Testing**: Tìm lỗi thông qua việc
      1. Đọc các đặc tả
      2. Đọc mã nguồn
      3. Tài liệu thiết kế
      4. Chưa biết hệ thống sẽ vận hành như mong đợi hay không.
   2. **Dynamic Testing**
      1. Tìm lỗi thông qua thực thi mã với dữ liệu test và môi trường giả lập
2. **Các kỹ thuật:** Black Box Testing, White Box Testing, Gray Box Testing
3. **Black Box Testing**

* Là kĩ thuật test chức năng của hệ thống/module/component mà **không quan tâm đến chi tiết cấu trúc thực thi bên trong**
* Black Box Testing dựa vào đặc tả yêu cầu hệ thống.
* Các tên gọi khác:
  + Specification-based Testing, Input-output Testing, Functional Testing
* Black Box Testing thường được thực thi gần các giai đoạn
  + Test tích hợp, Test giao diện, Test hệ thống, Test chấp nhận
* Một số dạng lỗi thường tìm được:
  + Lỗi giao diện
  + Hàm thực thi không đúng chức năng
  + Lỗi load
  + Quá trình khởi tạo/ kết thúc
* Trước khi tiến hành test
  + Phân rã yêu cầu thành các Test Objectives
  + Phân rã cho đến khi đạt được các Test Objectives tương đương với Test Case
* Các kỹ thuật phân rã
  + Equivalence Class Partitioning
  + Boundary Value Analysis
  + State Transition Testing
  + Cause-Effect Graphing
* **Equivalence Class Partitioning**
  + Phân hoạch lớp tương đương
  + Input/output có thể chia thành các lớp
  + Các giá trị trong cùng 1 lớp cho kết quả giống nhau khi đi qua “box”
  + Có thể lấy 1 giá trị trong lớp để đại diện cho lớp đó
* **Boundary Value Analysis**
  + Mở rộng Equivalence Class Partitioning bằng cách xét thêm giá trị biên của phân vùng
  + 

1. **White Box Testing**

* Có được source code
* Nhìn thấy được bên trong như thế nào
* Mục đích kiểm tra mọi path chạy qua chương trình
* Tập trung vào các hành vi bên trong
* Các kỹ thuật thường liên quan đến các đại lượng đo độ phủ (test coverage metric)
* Các bước thiết kế test cho coverage
  + Phân tích mã nguồn, vẽ **flow graph**
  + Đề xuất **test path** để có **coverage** từ **flow graph**.
  + Đánh giá điều kiện test để được **path**.
  + White Box Testing có thể sử dụng các công cụ tự động kiểm tra các giá trị coverage.

1. **Performance testing**

* **Performance testing là** việc thực hiện **test** để xác định một hệ thống có thể đáp ứng và ổn định với yêu cầu độ tải cao. Nó có thể phục vụ để điều tra, đo đạc, xác nhận hoặc xác minh chất lượng các thuộc tính của hệ thống như: khả năng thay đổi, tính tin cậy, và tài nguyên sử dụng.

1. **Stress Testing**

* **Stress Testing**: Stress testing là một hình thức kiểm thử được sử dụng để xác định tính ổn định của một hệ thống phần mềm. Là kiểu test kiểm tra thời gian đáp lại người dùng với số lượng người dùng bất kỳ trong nhiều ngữ cảnh khác nhau của cùng một ứng dụng tại cùng một thời điểm. Nó liên quan đến những kiểm thử vượt quá khả năng bình thường của hệ thống, thường để xác định các điểm phá vỡ của hệ thống, để quan sát các kết quả khi vượt qua ngưỡng giới hạn.Thực hiện bởi các kỹ sư hệ thống, Testers

1. **User Interface Testing**

* **User Interface Testing**: Là việc kiểm tra ứng dụng thông qua giao diện đồ họa (GUI) để kiểm tra giao diện của ứng dụng có đáp ứng được yêu cầu về thiết kế cũng như các hoạt động của từng thành phần trên giao diện đó (Click button, link...). Được thực hiện bởi các tester.

1. **Selenium**

* Selenium là một công cụ kiểm thử phần mềm tự động, được phát triển bởi ThoughtWorks từ năm 2004 với tên ban đầu là JavaScriptTestRunner
* Đến năm 2007, tác giả Jason Huggins rời ThoughtWorks và gia nhập Selenium team, một phần của Google và phát triển thành Selenium như hiện nay.
* SE là một công cụ kiểm tra tự động các ứng dụng web, nó có thể chạy trên hầu hết các trình duyệt và hệ điều hành.
* Selenium là một công cụ mã nguồn mở
* Có thể triển khai trên đa nền tảng: Windowns, Linux, Mac
* Hỗ trợ hầu hết các ngôn ngữ lập trình web như: HTML, Java, PHP, Python, Ruby, Perl,…
* SE bao gồm 4 phiên bản chính: Selenium IDE, Selenium Remote Control(Selenium RC) và Selenium Grid và Selenium WebDriver.
* **Selenium IDE (Integrated Development Environment)** là một công cụ cho phép chúng ta Record/Playback một test script. Đây là một add-on hỗ trợ cho FireFox. Chúng ta chỉ có thể Record trên trình duyệt FireFox, nhưng bù lại, chúng ta có thể Playback trên các trình duyện khác như là IE, Chrome….
* **Selenium Gird** là một hệ thống hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa test script.
* **Selenium RC, Selenium WebDriver** là một thư viện cho phép chúng ta lập trình (scripting) test script trên các ngôn ngữ lập trình khác nhau như Python, Java, C#, Ruby.

### Các đặc điểm của Selenium

1. Mã nguồn mở. Phải nói điểm này là điểm mạnh nhất của Selenium khi so sánh với các test tool khác. Vì là mã nguồn mở nên chúng ta có thể sử dụng mà không phải lo lắng về phí bản quyền hay thời hạn sử dụng.
2. Cộng đồng hỗ trợ. Vì là mã nguồn mở nên Selenium có một cộng đồng hỗ trợ khá mạnh mẽ. Bên cạnh đó, Google là nơi phát triển Selenium nên chúng ta hoàn toàn có thể yên tâm về sự hổ trợ miễn phí khi có vấn đề về Selenium. Tuy nhiên, đây cũng là một điểm yếu của Selenium. Cơ bản vì là hàng miễn phí, cộng đồng lại đông nên một vấn đề có thể nhiều giải pháp, và có thể một số giải pháp là không hữu ích. Mặc khác, chúng ta không thể hối thúc hay ra deadline cho sự hỗ trợ.
3. Selenium hỗ trợ nhiều ngôn ngữ lập trình.
4. Selenium hỗ trợ chạy trên nhiều OS khác nhau với mức độ chỉnh sửa script hầu như là không có. Thực sự thì điều này phụ thuộc phần lớn vào khả năng viết script của chúng ta.
5. Chạy test case ở backround. Khi chúng ta thực thi một test scrpit, chúng ta hoàn toàn có thể làm việc khác trên cùng một PC. Điều này hỗ trợ chúng ta không cần tốn quá nhiều tài nguyên máy móc khi chạy test script.
6. Không hỗ trợ Win app. Selenium thực sự chỉ hỗ trợ chúng ta tương tác với Browser mà không hỗ trợ chúng ta làm việc với các Win app, kể cả Win dialog như Download/Upload – ngoại trừ Browser Alarm. Vậy nên, để xử lý các trường hợp cần tương tác với hệ thống hay một app thứ ba, chúng ta cần một hay nhiều thư viện khác như AutoIt hay Coded UI.

**Ưu điểm**

* Selenium là một công cụ mã nguồn mở nên hoàn toàn miễn phí và không có vấn đề về bản quyền.
* Gọn nhẹ, đơn giản trong cài đặt. Dễ sử dụng, giao diện đơn giản, thân thiện.
* Hỗ trợ cho nhiều trình duyệt như IE, Firefox, Safari, Opera và một vài trình duyệt khác
* Selenium hỗ trợ Java, .NET và các ngôn ngữ lập trình tiêu chuẩn khác.
* Selenium có thể chạy trên nhiều hệ điều hành khác nhau như Windows, Unix, Mac,…
* Môi trường phát triển kiểm thử của Selenium đa dạng như Eclipse, Netbeans, Visual Studio,.. Tùy thuộc vào sự lựa chọn của bạn về ngôn ngữ phát triển.

**Nhược điểm**

* Selenium tích hợp với các hệ thông phát triển trên nền tảng web, nên không thích hợp với các phần mềm ứng dụng khác
* Không hỗ trợ win app
* Không có khả năng test GUI, vì không có các hàm hỗ trợ test giao diện như bắt cỡ chữ, cỡ tiêu đề, màu sắc,…
* Chưa có IDE cho các trình duyệt khác Firefox là một nhược điểm.
* Không tích hợp liền mạch với công cụ quản lý kiểm tra.

**Selenium driver:**

* Là công cụ kiểm thử phần mềm tự động mã nguồn mở với sự ủng hộ của các doanh nghiệp.
* Selinium script có thể chạy được trên hầu hết các trình duyệt như IE, Mozila FrieFox, Chrome, Safari, Opera.
* Hầu hết các hệ điều hành như Windows, Mac, Linux.
* Cho phép ghi âm, chụp ảnh, chỉnh sửa và kiểm tra gỡ lỗi.
* Hỗ trợ hầu hết các ngôn ngữ lập trình: html, java, .net, perl, ruby…
* Tính năng chính trong Selenium 2.0 là tích hợp Webdrive API, Webdrive được thiết kế để mang lại một giao diện lập trình đơn giản và giải quyết một số hạn chế của Selenium-RC API.
* Selenium Webdrive cho phép sử dụng một trong số các ngôn ngữ lập trình như HTML, Java, .net, php, ruby… để tạo kịch bản test kết hợp với các điều kiện vòng lặp khiến cho test script trở nên chính xác hơn.
* Selenium webdrie được phát triển tốt hơn để hỗ trọ cho các trang web động.
* Mục đích của webdrive là hỗ trợ cho các vấn đề kiểm thử web-app hiện nay.

**Selenium** là một tập hợp mạnh mẽ của các công cụ hỗ trợ phát triển nhanh chóng của các thử nghiệm tự động hóa cho các ứng dụng dựa trên web.Selenium cung cấp một tập phong phú của các thử nghiệm chức năng đặc biệt hướng đến các nhu cầu của các thử nghiệm của một ứng dụng web. Các hoạt động này là rất linh hoạt, cho phép nhiều tùy chọn cho vị trí các thành phần UI và so sánh kết quả thử nghiệm dự kiến sẽ chống lại hành vi ứng dụng thực tế.  
  
**Selenium** - nói 1 cách đơn giản là công cụ automator tất cả các thao tác của con người trên browser. Bạn có thể giả lập 1 set các action, VD như: <User bật browser lên, đăng nhập hệ thống bằng tài khoản username, password, user đi theo link route link1 - link2 - link3 - link4.

Selenium hỗ trợ kiểm tra hầu hết trên các trình duyệt phổ biến hiện nay như Firefox , Internet Explorer , Safari ,.. cũng như các hệ điều hành chủ yếu như Windows , Linux , Mac ,.. Selenium hỗ trợ một số lớn các ngôn ngữ lập trình như C# , Java , Perl , PHP , Python, Ruby,..

Selenium có thể kết hợp thêm với một số công cụ khác như Bromien , Junit nhưng với người dùng thông thường chỉ cần chạy tự động mà không cần cài thêm các công cụ bổ trợ.

Selenium bao gồm một bộ công cụ hỗ trợ kiểm tra tự động tính năng của ứng dụng web bao gồm: Selenium IDE, Selenium Remote control(RC), Selenium Core và Selenium Grid.

* Selenium IDE được phát triển dưới dạng một ứng dụng mở rộng (add-on) cho Mozilla Firefox phiên bản 2.0 trở lên. Công cụ cung cấp chức năng Record and Playback.
* Selenium Remote Control với bộ công cụ này cho phép nhận các Test script được thu bởi Selenium IDE, cho phép chỉnh sửa cải tiến linh động bằng ngôn ngữ lập trình khác nhau . Sau đó khởi động một trong các trình duyệt web được chỉ định để thực thi kiểm tra trực tiếp trên trình duyệt đó. Selenium RC còn cũng cấp khả năng lưu lại kết quả kiểm tra.
* Selenium Grid thực hiện phương pháp kiểm tra phân bố, phối hợp nhiều kết quả Selenium RC để có thể thực thi trên nhiều trình duyệt web khác nhau trong cùng một lúc. Selenium Grid cũng cho phép lưu lại kết quả kiếm tra.
* Selenium Core được tích hợp trong Selenium IDE, là một công cụ chạy các testscript viết bằng Selenese

**SoapUI là** một dịch vụ web ứng dụng thử nghiệm mã nguồn mở, với tính năng dễ sử dụng, **soap ui** cho phép bạn kiểm tra dễ dàng và nhanh chóng cho phép thực hiện các thử nghiệm chức năng trên web

1. **SOAP là gì?**

* SoapUI là một sản phẩm của hãng SmartBear,
* SOAP UI là công cụ kiểm thử API mã nguồn mở, cho phép thực hiện việc kiểm thử chức năng một cách tự động trên các Web API khác nhau.
* SOAP UI hỗ trợ tất cả các chuẩn giao thức và công nghệ để test tất cả các loại API. Ngoài ra SOAP UI còn cho phép thực hiện thử nghiệm phi chức năng như kiểm thử hiệu suất và kiểm thử bảo mật.
* Giao diện SOAP UI đơn giản, thân thiện, dễ sử dụng.

1. **Các tính năng của SOAP UI**

* **Kiểm thử chức năng:** 
  + Một công cụ mạnh mẽ cho phép tester viết Functional API Tests trong SoapUI
  + Hỗ trợ tính năng kéo-thả mà làm tăng tốc độ phát triển scriptHỗ trợ gỡ lỗi và cho phép tester phát triển data driven tests.
* **Kiểm thử bảo mật:** 
  + Ngăn chặn SQL Injection để bảo đảm cơ sở dữ liệu
  + Thực hiện Fuzzing scan và Boundary scan để tránh những hành vi thất thường của các dịch vụ.
* **Kiểm thử tải:** 
  + Kiểm thử khả năng chịu tải của một ứng dụng web sử dụng loadUI. Sau khi thực hiện kiểm tra tải, LoadUI sẽ tạo ra một bản báo cáo, giúp xác định liệu các ứng dụng có thể chịu tải nặng hay không.
  + Kiểm thử khả năng chịu tải của một ứng dụng web sử dụng loadUI
  + Mô phỏng mức độ cao và kiểm thử tải thực tế một cách dễ dàng.
  + Cho phép tùy chỉnh báo cáo chi tiết để nắm bắt các thông số hiệu suất.
* **Tích hợp với các công cụ tự động khác:**
  + Maven
  + HUDSON
  + Junit
  + Apache – Ant

## **Maven là gì?**

* Maven là một công cụ quản lý và thiết lập tự động một dự án phần mềm. Chủ yếu dùng cho các lập trình viên Java, nhưng nó cũng có thể được dùng để xây dựng và quản lý các dự án dùng C#, Ruby, Scala hay ngôn ngữ khác
* Maven phục vụ mục đích tương tự như Apache Ant, nhưng nó dựa trên khái niệm khác và cách hoạt động khác.
* Maven hỗ trợ việc tự động hóa các quá trình tạo dự án ban đầu, thực hiện biên dịch, kiểm thử, đóng gói và triển khai sản phẩm
* Được phát triển bằng ngôn ngữ Java cho phép Maven chạy trên nhiều nền tảng khác nhau: Window, Linux và Mac OS….
* Là một công cụ, hỗ trợ cho việc tự động hóa tạo các project, thực hiện biên dịch, kiểm thử đóng gói và triển khai sản phẩm

## **Tại sao nên dùng Maven?**

* Trong trường hợp nhiều team làm việc với nhau, Maven có thể thiết lập một chuẩn làm việc trong thời gian rất ngắn. Các project sẽ được thiết lập cho đơn giản và dễ sử dụng, Maven sẽ làm cho cuộc sống của lập trình viên sẽ dễ dàng hơn khi tạo các report, kiểm tra, xây dựng và thiết lập kiểm thử tự động
* Maven cung cấp cho lập trình viên cách để quản lý:
  + Build, Documentation , Reporting, Dependencies, SCMs, Releases, Distribution
  + Mailing list
* Tóm lại, Maven đơn giản hóa và chuẩn hóa quá trình build dự án. Nó xử lý compilation, distribution, documentation, team collaboration và các task khác. Maven tăng tính tái sử dụng, và xử lý hầu hết các task liên quan để build dự án.

## **Mục tiêu của Maven**

* Mục đích chính của maven là cung cấp cho lập trình viên:
* Một mô hình hoàn chỉnh bao quát cho các project như là tái sử dụng, dễ bảo trì, và dễ dàng để hiểu
* Các plugin và các công cụ dùng dể tương tác với các khai báo trong mô hình này.
* Cấu trúc và nội dung của project Maven được mô tả trong một file xml, pom.xml viết tắt của Project Object Model (POM), nó là một đơn vị căn bản của toàn hệ thống Maven.

**Lợi ích của Maven**  
  
\* Quản lý các dependencies (các thư viện) trong project một các ưu việt. Tự động cập nhật, mở rộng dễ dàng, đóng gói các thư viện mà project sử dụng. Maven có một repository lớn chứa toàn bộ các thư viện cần thiết ([http://mvnrepository.com](http://mvnrepository.com/)), nó không chỉ lưu trữ gói mà còn lưu cả version, các version này được cập nhật liên tực bởi người phát triển ra nó. Chính vì thế khi cần thay đổi version thì rất nhanh chóng.  
\* Tự động hóa toàn bộ quá trình release project. Một dự án bao gồm các quá trình như khởi tạo, cập nhật thư viện, build & test, release và tạo tài liệu mô tả. Hãy thử tưởng tượng về một dự án lớn, trong đó các hàng chục các module làm các công việc khác nhau. Mỗi team phụ trách một module và quá trình phát triển các module: từ phân tích yêu cầu, đặc tả, đến phát triển, kiểm thử và release là đồng thời và độc lập. Nhưng các module này phụ thuộc vào nhau, cần thiết cho nhau để phát triển. Chỉ cần một sự chậm trễ, sai lầm ở một công đoạn có thể làm đổ vỡ cả quá trình release của project. Cho nên việc tự động hóa toàn bộ là ưu tiên hàng đầu, nhờ đó là việc lên kế hoạch hiệu quả và chính xác hơn, việc tích hợp và triển khai cũng đơn giản hơn rất nhiều.  
\* Tự động hóa quá trình test với các plugin. Toàn bộ quá trình test được tự động, từ đó các lỗi được phát hiện sớm hơn. Thậm chí lỗi có thể phát hiện ở giai đoạn phát triển, do các developer đều có thể chạy các test case một cách dễ dàng bằng Maven.  
\* Phân chia 1 project lớn thành các module nhỏ. Từ đó cho phép làm việc đồng thời trên các module khác nhau, đồng thời vẫn tạo được tính thống nhất.  
  
\* Không phụ thuộc vào IDE. Cho phép chuyển đổi tử Eclipse sang NetBeans,.. một cách dễ dàng. Maven hỗ trợ một số command để tạo ra project tương thích với IDE muốn sử dụng (ví dụ: mvn eclipse:eclipse – để tạo project với eclipse).

-------------------------------------------------------------------------------------------------------------------------------------

1. **Giới thiệu về Maven**

* Maven là một công cụ giúp build và đóng gói các project java một cách tự động và đơn giản. Maven cũng được định nghĩa như là một công cụ quản lý dự án toàn diện. Nó nhằm mục đích cung cấp cho các lập trình viên một framework giúp xây dựng và quản lý chi tiết và đầy đủ một ứng dụng java.
* Maven cũng dễ dàng lên các nhiệm vụ của các lập trình viên trong việc kiểm tra tình trạng xây dựng, tạo báo cáo (về cơ bản javadocs) và thiết lập các quá trình tự động xây dựng và giám sát việc cùng.
* Sự dễ dàng của biên dịch mã nguồn, phân phối, tài liệu hướng dẫn, làm việc nhóm và nhiệm vụ quan trọng khác khi sử dụng maven.

2. Maven nhằm 2 mục đích quan trọng:  
\* Làm thế nào một phần mềm được xây dựng.

\* Các phụ thuộc dependencies (thư viện), plug-ins và hồ sơ dự án được liên kết trong một môi trường độc lập hoặc một phân tán.

\* Maven cũng có thể được sử dụng trong việc xây dựng và quản lý dự án được viết bằng C #, ruby và các ngôn ngữ lập trình khác.​

3. Ưu điểm của việc sử dụng Maven so với Ant

\* Quản lý sự phụ thuộc dependencies như thư viện.

\* Cấu hình đơn giản, dễ sử dụng theo tiêu chuẩn chung.

\* Xay dựng một hoặc nhiều dự án đơn giản.

\* Tập trung vào tự động hóa.

\* Quản lý Plugin, hỗ trợ build

\* Thử nghiệm - khả năng chạy JUnit và các tool kiểm thử tích hợp khác.

\* Làm cho quá trình phát triển thông suốt.

\* Cung cấp tình hình từng giai đoạn của quá trình xây dựng dự án.

\* Tránh các thiết lập xung khắc trong dự án.

\* Tiêu chuẩn và thống nhất giữa các dự án.

1. **Giới Thiệu Về Gradle**

* Là một công cụ xây dựng tự động mở nguồn mở , dùng để quản lí việc build hệ thống .
* Sử dụng ngôn ngữ DST (domain specific language) dựa trên Groovy (ngôn ngữ động của JVM) thay vì sử dụng XML như Apache Maven để khai báo cấu hình dự án
* Được thiết kế để xây dựng những dự án lớn
* Tập trung chủ yếu xung quanh Java và đang phát triển hỗ trợ các ngôn ngữ khác
* Là một công cụ tự động build project dựa trên các khái niệm của Apache Ant và Apache Maven, đồng thời giới thiệu một “domain specific language” dựa trên Groovy thay cho hình thức khai báo cấu hình truyền thống (XML) của project.
* Bên cạnh việc cấu hình project được thực hiện thông thường trong file xml (AndroidMainifest.xml). Gradle là một công cụ được tích hợp sẵn tự động trong  Android Studio để giúp những nhà phát triển tổ chức, quản lý và từng bước xây dựng ứng dụng của họ.

--

* Gradle là một công cụ tự động build project.
* Gradle là sự kết hợp của Maven và Ant
* Gradle có thể khởi tạo project, biên dịch, đóng gói, chạy test… một cách tự động và nhanh chóng.
* Gradle sinh ra để giải quyết bài toán tự động hóa quá trình xây dựng và quản lý các thư viện tham khảo

**Tính năng nổi bật của Gradle**

* **Tasks**
  + Mỗi project gradle thực hiện tự động hóa build được tạo thành nhiều task khác nhau.
  + Mỗi Task là 1 phần công việc mà build project thực hiện.
  + Mỗi task có thể biên dịch thành nhiều lớp, files, tạo thư viện jar, tạo java doc,…
* **Groovy**
  + Groovy là ngôn ngữ kịch bản mà Gradle dùng để quản lý mọi thành phần như build library, quản lý depedency.
  + Groovy là 1 ngôn ngữ hướng đối tượng trên Java
  + Cấu trúc Groovy đơn giản giúp giảm bớt sự khó khăn khi tạo 1 script tự động hóa biên dịch
* **Testing**
  + **Test Detection:** Test task phát hiện các lớp bằng cách quét tất cả các file .class. Tùy thuộc vào framework test sử dụng như (Junit, TestNG)
  + **Test Grouping:** Junit và TestNG cho phép nhóm các test methods
* **Quản lý phụ thuộc**
  + Mỗi project khi phát triển sẽ có 1 vài dependencies và 1 vài publications.
  + Dependencies là 1 thứ hỗ trợ cho việc build project
  + Publications là kết quả của project
  + Gradle xây dựng script cho 1 quá trình build project
* **Cấu hình build cho nhiều project**