1. **Các khái niệm cơ bản:**
2. Test bao gồm:

+ Nội dung: Kiểm lỗi, Kiểm lỗi phân hệ, Kiểm lỗi hệ thống.

+ Kết quả: Roadmap, Test plan, Test case, Bug, Test report

1. Khái niệm kiểm thử:

Là hoạt động kiểm tra xem phần mềm có chạy chính xác hay không (Verification) và có thoả mãn yêu cầu của khách hàng hay không (Validation) nhằm hướng tới mục tiêu Chất lượng cho phần mềm.

1. Mục đích của kiểm thử:

* Giảm chi phí trong quá trình phát triển, nâng cấp, bảo trì phần mềm.
* Chất lượng phần mềm được nâng cao.
* Chúng ta có thể đánh giá chất lượng phần mềm dựa vào số lượng lỗi tìm thấy và các đặc tính như: tính đúng đắn, tính dễ sử dụng, tính dễ bảo trì.

1. Thời điểm tiến hành kiểm thử: Tiến hành ở mọi công đoạn phát triển phần mềm
2. Yêu cầu kiểm thử:

+ Tính lặp lại:

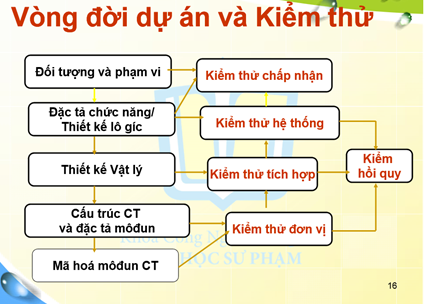
* Kiểm thử phải được lặp lại(kiểm tra xem lỗi đã được sửa hay chưa)
* Dữ liệu/trạng thái phải được mô tả

+ Tính hệ thống:

* Đảm bảo kiểm tra hết các trường hợp

+ Được lập tài liệu: Kiểm soát tiến trình/kết quả.

1. Vòng đời dự án kiểm thử:



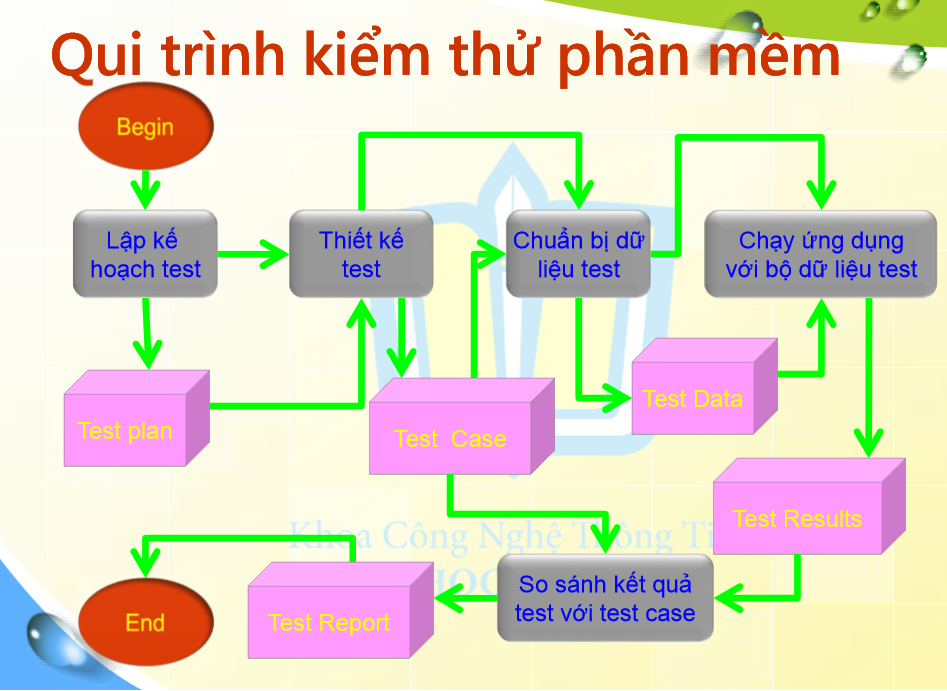
1. Quy trình kiểm thử:

+ Kiểm thử thành phần:

* Kiểm thử từng thành phần của chương trình
* Thường là trách nhiệm của lập trình viên tạo ra thành phần đó
* Các test được tạo ra từ kinh nghiệm của lập trình viên

+ Kiểm thử hệ thống:

* Kiểm thử một nhóm các thành phần được kết hợp lại để tạo ra hệ thống hay hệ thống con.
* Trách nhiệm của một đội test độc lập
* Các test được tạo ra dựa trên bản đặc tả hệ thống.



1. Nguyên tắc kiểm thử:

+ Chọn các input làm cho hệ thống tạo ra tất cả các thông báo lỗi

+ Thiết kế input làm tràn bộ đệm

+ Lặp lại cùng một input hay một dãy các input một vài lần

+ Ép các output không hợp lệ phải xuất hiện

+ Ép các kết quả tính toán phải hoặc là quá lớn hoặc là quá nhỏ

1. Chính sách kiểm thử(Testing policy):

+ Kiểm tra tất cả các chức năng trong hệ thống menu

+ Kiểm tra tất cả các mục khác có cùng chức năng trong hệ thống menu (Toolbar, Listbar, Dialog bar, Context Menu,…)

+ Kiểm tra cùng một chức năng với nhiều vai trò khác nhau (đối với hệ thống có nhiều người dùng)

+ Kiểm tra tất cả các dữ liệu bắt buộc nhập trong các màn hình (hợp lệ/không hợp lệ)

1. Các phương pháp kiểm thử:

* Theo khái niệm thông thường được nhắc đến Testing thường gồm 2 phương pháp là: Kiểm thử hộp đen (Black Box Testing) và Kiểm thử hộp trắng (White Box Testing).
* Theo ISTQB thì có 2 kiểu kiểm thử là: Kiểm thử động (Dynamic Testing) và Kiểm thử tĩnh (Static Testing).

1. White-Box

* Khái niệm kiểm thử hộp trắng: Kiểm thử hộp trắng dựa vào thuật toán, cấu trúc code bên trong của chương trình với mục đích đảm bảo rằng tất cả các câu lệnh và điều kiện sẽ được thực hiện ít nhất một lần.
* Đặc điểm của kiểm thử hộp trắng:
* Người kiểm thử truy cập vào mã nguồn chương trình và có thể kiểm tra nó, lấy đó làm cơ sở để hỗ trợ việc kiểm thử.
* Việc kiểm thử bằng kỹ thuật hộp trắng không thể đảm bảo rằng chương trình đã tuân theo đặc tả.
* Một chương trình sai do thiếu đường dẫn. Việc kiểm thử hộp trắng không thể biết được sự thiếu sót này.
* Việc kiểm thử bằng kỹ thuật hộp trắng không thể phát hiện được lỗi do dữ liệu. Như vậy, việc kiểm thử chỉ dùng kỹ thuật hộp trắng là không đủ để phát hiện lỗi.
* Kiểm thử hộp trắng bao gồm:
* Kiểm thử đường dẫn (Path test) : Kiểm thử bao quát các dòng source code, nhánh và đường dẫn.
* Kiểm thử luồng điều khiển(Control flow test) : Xác nhận truy cứu các lịch sử thực hiện source code bằng cách sử dụng trình gỡ lỗi.
* Kiểm thử nội bộ: Xác nhận các tham số, counter, vòng lặp
* Kiểm thử tính năng: Đo thời gian xử lý của module, đường dẫn, dữ liệu cụ thể.
* Các kỹ thuật trong kiểm thử hộp trắng:
* Kiểm thử đường cơ bản(Basic Path Testing)
* Kiểm thử dữ liệu luồng dữ liệu(Data Flow Testing)
* Kiểm thử vòng lặp(Loop Testing)
* Kiểm thử điều kiện(Condition Testing)
* Kiểm thử luồng dữ liệu:
* Phương pháp tạo các kiểm thử dựa vào vị trí khai báo và quá trình sử dụng biến trong chương trình, gọi là chuỗi khai báo sử dụng của biến.
* Mỗi chuỗi cần được duyệt qua và kiểm thử ít nhất 1 lần. Phương pháp này không đảm bảo mọi nhánh thực hiện chương trình nhưng thích hợp để kiểm tra độ chính xác của dữ liệu khi thực hiện.

1. Black-box:

* Khái niệm kiểm thử hộp đen: Là phương pháp test dựa trên đầu vào và đầu ra của chương trình để test mà không quan tâm tới code bên trong được viết ra sao. Tester xem phần mềm như là một hộp đen
* Đặc điểm kiểm thử hộp đen:
* Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out
* Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình.
* Kiểm thử hộp đen bao gồm:
* Kiểm thử chức năng và kiểm thử hệ thống
* Kiểm thử quá tải và kiểm thử hỏng hóc
* Kiểm tra hiệu năng
* Các kỹ thuật trong kiểm thử hộp đen:
* Phân vùng tương đương (Equivalence partitioning)
* Phân tích giá trị biên (Boundary value analysis)
* Đồ thị nguyên nhân - kết quả(Cause & Effect Graphing)
* Đoán lỗi (Error Guessing)
* Phân tích giá trị biên(Boundary value analysis):
* Chia (partition) đầu vào thành những nhóm tương đương nhau (equivalence). Nếu một giá trị trong nhóm hoạt động đúng thì tất cả các giá trị trong nhóm đó cũng hoạt động đúng và ngược lại.
* Mục đích : Giảm đáng kể số lượng test case cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.
* Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

(1). Xác định các lớp tương đương

(2). Xác định các ca kiểm thử.

1. So sánh kiểm thử hộp trắng và kiểm thử hộp đen:

* Giống nhau:

Hai loại hình kiểm thử đều nhằm mục đích là kiểm định lại phần mềm nhằm loại bỏ những lỗi và những gì thiếu trong quá trình làm phần mềm nhằm mục đích là đem lại một sản phẩm tốt đến khách hàng.

* Khác nhau:
* Kiểm thử hộp trắng là phương pháp kiểm thử dựa vào cấu trúc/mã lệnh chương trình. Phương pháp kiểm thử hộp trắng kiểm thử một chương trình (một phần chương trình, hay một hệ thống, một phần của hệ thống) đáp ứng tốt tất cả các giá trị input bao gồm các giá trị không đúng hay không theo dự định của chương trình. Phương pháp dựa trên:
* Các câu lệnh(Statement)
* Đường dẫn(Path)
* Các điều kiện(Condition)
* Vòng lặp(loop)
* Ngã rẽ(branch)
* Kiểm thử hộp đen hay còn gọi là kiểm thử chức năng, việc kiểm nghiệm này được thực hiện mà không cần quan tâm đến các thiết kế và viết mã của chương trình. Kiểm thử theo cách này chỉ quan tâm đến chức năng đã đề ra của chương trình. Vì vậy kiểm thử loại này chỉ dựa vào bản mô tả chức năng của chương trình, xem chương trình có thực sự cung cấp đúng chức năng đã mô tả trong bản chức năng hay không mà thôi.Phương pháp dựa trên:
* Chức năng thiếu hoặc không đúng đắn.
* Sai về giao diện.
* Sai trong cấu trúc hoặc hoặc trong truy cập dữ liệu ngoài.
* Sai thực thi chức năng.

1. Khái niệm loại kiểm thử? Có mấy loại?

* Khái niệm: Loại kiểm thử được hiểu là các loại hình kiểm thử (như Kiểm thử chức năng, Giao diện, Tải, Bảo mật, ...). Khái niệm này dễ bị nhầm với các Mức (còn được gọi là Giai đoạn) kiểm thử (như: Unit Test, Integration Test, System Test, ....).
* Các loại kiểm thử:
* Kiểm thử chức năng (Functional Testing)
* Kiểm thử giao diện (Interface Testing)
* Kiểm thử khả năng chịu tải (Performance Testing)
* Kiểm thử khả năng tới hạn (Stress Testing)
* Kiểm thử bảo mật (Security Testing)

1. Mức kiểm thử là gì? Có mấy mức?

* Khái niệm: Mức kiểm thử được hiểu là các giai đoạn kiểm thử gắn liền với hoạt động của Quy trình kiểm thử (và liên quan chặt chẽ tới Quy trình sản xuất của một phần mềm).
* Các mức kiểm thử: Có 4 mức là: Unit Test, Integration test, System test và Acceptance Test.
* Unit Test:
* Là giai đoạn kiểm thử mức đơn vị, thường được thực hiện bởi các Lập trình viên nhằm kiểm tra sự đúng đắn của từng Procedure, Function hay các đoạn mã lập trình.
* Unit Test tương đối mất thời gian nhưng đem lại hiệu quả cao về mặt chất lượng do các lỗi lặt vặt và nhỏ lẻ của function được loại bỏ từ sớm trong giai đoạn Unit Test và khi bàn giao cho Tester thì Tester sẽ tập trung tìm kiếm được nhiều lỗi nguy hiểm hơn.
* Integration Test:
* Là giai đoạn Test tích hợp với hình thức Black Box.
* Ở giai đoạn này Tester tiếp nhận các chức năng, module đã lập trình xong, thực hiện kiểm thử ở mức đơn vị thấp nhất có thể (ví dụ: 1 chức năng Thêm mới ở 1 module Gửi tiền, ....) và tích hợp dần lên các mức đơn vị lớn hơn.
* Kỹ năng Phân tích và Phân loại mức đơn vị nhỏ nhất cần Test là điều quan trọng cho Tester ở giai đoạn này.
* System Test:
* Là giai đoạn Test tổng thể cả hệ thống sau khi đã test từng bộ phận.
* Stress Test hay Performance Test và Security Test thường được thực hiện ở giai đoạn này.
* Acceptance Test:
* Là giai đoạn kiểm thử mức chấp nhận từ phía khách hàng (các bạn nhớ Acceptance Test là do khách hàng thực hiện đấy nhé).

1. **White box và Black box**

**White Box Testing** là việc thực hiện chạy ứng dụng khi biết cách hoạt động bên trong của nó dựa vào source code. White box testing được sử dụng trong 3 trên 6 mức test cơ bản: unit, integration, và regression testing.

test cases của white box sẽ được test theo các đường dẫn, các điểm quyết định (cả quyết định đúng và quyết định sai), các vòng lặp khác nhau và kiểm tra cấu trúc data bên trong của ứng dụng. Kiểm thử đường cơ bản (basis path testing), Phân vùng tương đương (equivalence partitioning), và phân tích giá trị biên (boundary value analysis) tất cả các phương pháp này đều được sử dụng để tạo test cases white box.

Mục tiêu của white box testing là test bao phủ nhiều nhất các câu lệnh, điểm quyết định và các rẽ nhánh trong code nếu có thể.

**Data flow**: Tập trung vào kiểm tra giá trị của các biến trong chương trình. Biến sẽ xuất hiện theo 2 dạng: khai báo và gán giá trị. Biến sẽ được sử dụng theo 2 cách: predicate (kiểm tra điều kiện) và computational (tính toán). Data Flow Testing tập trung vào những điểm mà tại đó các biến nhận giá trị và các điểm mà tại đó các giá trị này được sử dụng.

Data flow giúp xác định bất kỳ các vấn đề sau đây:

* Một biến được khai báo, nhưng không bao giờ được sử dụng trong chương trình.
* Một biến được sử dụng nhưng không bao giờ được khai báo.
* Một biến được định nghĩa nhiều lần trước khi nó được sử dụng.
* Phân bổ một biến trước khi nó được sử dụng. (Deallocating a variable before it is used.)

**Phương pháp kiểm thử Black box** là nghiên cứu xem phần mềm như là một “hộp đen” – không biết gì về hoạt động bên trong của nó. Các phương pháp kiểm thử Black box bao gồm Equivalence partitioning (phân vùng tương đương), boundary value analysis (phân tích giá trị biên), all-pairs testing (kiểm thử tất cả các cặp), fuzz testing (cách test: nhập vào các điều kiện sai hoặc data một cách ngẫu nhiên), model-based testing (Kiểm thử dựa trên model), traceability matrix (các chức năng của chương trình được tạo thành một ma trận, các trường hợp test là sự kết hợp các dòng hoặc các cột có liên quan), exploratory testing (kiểm thử chủ yếu dựa vào kinh nghiệm và khả năng focus vào việc test các chức năng của tester) và specification-based testing (kiểm thử dựa vào chức năng).

Ưu điểm và nhược điểm: Các tester kiểm thử theo phương pháp black box không có “mối ràng buộc” nào với code, và nhận thức của một tester rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bug ở nơi mà các DEV không tìm thấy. Mặt khác, việc kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin” bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào. Như là một kết quả, ở đây có nhiều trường hợp khi một tester viết rất nhiều trường hợp test để kiểm tra một số thứ có thể chỉ được test bằng một trường hợp test (1), và/hoặc một vài phần cuối cùng không được test hết.  
Vì vậy, black box testing có ưu điểm là "an unaffiliated opinion" (một quan điểm độc lập), mặt khác, nó có nhược điểm là "blind exploring" (khám phá mù).

**Equivalence class partitioning** – Phân lớp tương đương

Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy chúng ta có thể test một giá trị đại diện trong vùng tương đương.

- Phân các test cases theo nhóm các TEST CASE cùng loại, gọi là class hay lớp các TEST CASE.

- Trong mỗi class chọn test chỉ một vài test case.

- Nên test nhiều class thay cho test nhiều test cases trong cùng một class.

Một lợi thế của phương pháp này là nó làm giảm thời gian cần thiết để kiểm thử hiệu năng của một phần mềm do số lượng test case ít hơn.

**Boundary value analysis**, Hầu hết các lỗi được tìm thấy khi kiểm tra ở các giá trị biên. Vì vậy phương pháp này tập trung vào việc kiểm thử các giá trị biên này.

Phân tích giá trị biên là trường hợp đặc biệt của phân vùng tương đương(**Equivalence class partitioning**), dựa trên những phân vùng tương đương tester sẽ xác định giá trị biên giữa những phân vùng này và lựa chọn test case phù hợp.

Kiểm thử các dữ liệu vào bao gồm:

* Giá trị nhỏ nhất.
* Giá trị gần kề lớn hơn giá trị nhỏ nhất.
* Giá trị bình thường.
* Giá trị gần kề bé hơn giá trị lớn nhất.
* Giá trị lớn nhất.

**Ưu và nhược điểm**

* Ưu điểm
  + Hoạt động tốt đối với các chương trình có hàm mà biến đầu vào độc lập với nhau
  + Tiết kiệm thời gian thiết kế test case và thực hiện test.
* Nhược điểm
  + Chỉ phù hợp với các chương trình có biến đầu vào độc lập với nhau và biểu diễn đại lượng vật lý bị chặn (có biên rõ ràng).
  + BVA lấy các test case mà không quan tâm đến chức năng của hàm hay ý nghĩa của biến.
  + Chỉ phù hợp với việc kiểm thử trên “free-form” languages (weak typing language) such as: Cobol, fortran

1. **Tìm hiểu các công cụ Testing:**
   1. **Selenium**

* Khái niệm:
  + 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.
  + Năm 2008 kết hợp cùng với WebDriver và phát triển thành Selenium như ngày 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.
  + 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.
* Tính năng:
  + Selenium IDE (Selenium Integrated Development Environment):
    - Là framework đơn giản nhất trong bộ Selenium.
    - Cho phép Record/Playback một test script.
    - Là một add-on hỗ trợ trên Firefox phiên bản 2.0 trở lên.
    - Cho phép lưu test case đã Record dưới nhiều ngôn ngữ lập trình khác nhau như Java, PHP, C#, Python, Ryby hay Perl.
  + Selenium RC (Selenium Remote Control):
    - Là công cụ kiểm tra cho phép bạn sử dụng một ngôn ngữ lập trình để chỉnh sửa, cải tiến linh động và mở rộng trong việc phát triển logic thử nghiệm.
    - Có thể nhận các test script được thu bởi Selenium IDE.
    - Cung cấp một API và thư viện cho mỗi ngôn ngữ được hỗ trợ: HTML, Java, C#, Perl, PHP, Python và Ruby.
    - Selen-RC sử dụng toàn bộ sức mạnh của ngôn ngữ lập trình để tạo thêm các xét nghiệm phức tạp như đọc và viết các tập tin, truy vấn một cơ sở dữ liệu, và gửi email các kết quả thử nghiệm.
    - Làm việc với Browser thông qua một hệ thống Server ảo
  + Selenium WebDriver:
    - Là một bản nâng cấp của Selenium RC cho phép gửi các câu lệnh trực tiếp đến trình duyệt và truy xuất kết quả.
    - Được thiết kế để cung cấp một giao diện lập trình ngắn gọn đơn giản hơn ngoài việc giải quyết một số hạn chế trong các API Selenium-RC.
    - Được phát triển tốt hơn để hỗ trợ cho các trang web động. Mục đích của WebDriver là hỗ trợ cho các vấn đề về kiểm thử web-app hiện nay.
    - Cho phép chúng ta sử dụng các ngôn ngữ lập trình – Java; .Net; PHP; Python; Perl; Ruby – để thiết kế test case.
    - Làm việc trực tiếp với Browser nên tốc độ được cải tiến rất nhiều.
  + Selenium Grid:
    - Hỗ trợ cho Selenium RC và Selenium WebDriver.
    - Selenium Grid cho phép người dùng thực thi kiểm thử song song trên nhiều máy khác nhau với nhiều trình duyệt khác nhau.
    - Có 2 lý do để sử dụng Selenium Grid**:**
      * Để kiểm thử trên nhiều trình duyệt, nhiều phiên bản của trình duyệt, và các trình duyệt chạy trên các hệ điều hành khác nhau.
      * Để giảm thời gian hoàn thành của việc kiểm thử
  1. **SOAP UI**
* Khái niệm:
  + SoapUI là một sản phẩm của hãng SmartBear
  + SOAP UI là công cụ kiểm thử mã nguồn mở Web API hàng đầu
* Tính năng:
  + Kiểm thử chức năng: Functional API Testing, Kéo-thả mà hỗ trợ dễ dàng phát triển script, Hỗ 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, Mô phỏng mức độ cao và kiểm thử tải thực tế một cách dễ dàng.
  + Mô phỏng dịch vụ (Service Simulation): Soap UI MockServices cho phép bạn bắt chước, tạo ra các bài kiểm tra mạnh mẽ đối với các dịch vụ Web trước khi chúng được thực hiện
  + Tự động (Automation): SoapUI tạo Continuous Integration với Hudson một các dễ dàng, Sử dụng TestRunner
  + Hỗ trợ nhiều giao thức và công nghệ như: SOAP, HTTP, WSDL, REST, AMF, JDBC, …
  1. **Sahi**
* Khái niệm:
  + Sahi là một công cụ tự động và kiểm thử các ứng dụng web.
  + Sahi là một sản phẩm mã nguồn mở miễn phí.
  + Chạy lại trên bất kỳ trình duyệt máy tính và trên cả các trình duyệt di động.
  + Sahi đặc biệt thích hợp cho quá trình duyệt, thử nghiệm web.
  + Sahi hoạt động tốt trong môi trường Agile
* Tính năng:
  + Record on all browsers
  + Playback on all browsers
  + HTML playback reports
  + JUnit Style playback reports
  + Suites and batch run
  + Parallel playback of tests
  + Hỗ trợ nhiều proxy ngoại, HTTPS, 401 & NTLM authentications
  + Hỗ trợ AJAX và ứng dụng web rất cao
  + Có thể sao lưu trực tiếp Java code từ Script. Điều này được sử dụng cho việc truy cập database, đọc file pdf, kết nối với máy tính khác thông qua SSH
  + Hỗ trợ kết nối data. Có thể kết nối với database, Excel hoặc CSV file.
  + Sahi tự động chờ tải trang và hoạt động của Ajax.Điều này làm giảm kích thước cơ sở mã bằng 50% so với các công cụ khác và làm cho các kịch bản mạnh mẽ nhưng dễ duy trì.
  + Sahi hỗ trợ tùy chỉnh thẻ HTML5 và Shadow DOM.
  1. **Watir**
* Khái niệm:
  + Watir - viết tắt của Web Application Testing In Ruby, là một công cụ dùng để kiểm thử các ứng dụng web bằng ngôn ngữ Ruby
  + Watir là một bộ thư viện được viết bằng ngôn ngữ Ruby hỗ trợ lập trình việc tự hành của trình duyệt web.
  + Watir là một công cụ OPEN SOURCE
* Tính năng:
  + Cho phép người dùng viết các test case tự hành một cách đơn giản, linh hoạt, dễ đọc và dễ bảo trì.
  + Cho phép kiểm thử nhiều ứng dụng khác nhau không quan trọng ứng dụng đó viết bằng công nghệ nào miễn nó được chạy trên web
  + Hỗ trợ các chức năng của một ngôn ngữ lập trình, chẳng hạn như kết nối và đọc dữ liệu từ cơ sở dữ liệu
  + Watir-classic hoạt động trực tiếp trên trình duyệt IE thông qua cổng OLE (Object Linking and Embedding)
  + Watir-WebDriver có thể chạy trên hầu hết các trình duyệt thông dụng hiện nay
  + WatirSpec - Những những thư viện cơ bản được lập trình sẵn để sử dụng Watir API.
  1. **Maven**
* Khái niệm:
  + Maven là một công cụ để quản lý và bao quát dự án. Maven cung cấp cho các nhà phát triển một khuôn khổ để xây dựng một vòng đời hoàn chỉnh (a complete build lifecycle framework).
  + Maven đã được phát triển như là một cái chuẩn để xây dựng các dự án. Các hệ thống phát triển cùng một chuẩn sẽ dễ dàng thay đổi cũng như chia sẻ một cách đồng bộ.
  + **Apache Maven** là chương trình quản lý dự án được sử dụng nhiều bởi Java developer, nó thực hiện các công việc như khởi tạo project, biên dịch, đóng gói, chạy test… một cách tự động và nhanh chóng. Nó đặc biệt hữu dụng với các dự án tầm trung và lớn.
* Tính năng:
  + Giúp các nhà phát triển Java dễ dàng hơn trong việc quản lý các dự án Java.
  + Cung cấp các tính năng như Unit Test, Integration Test
  + Cung cấp các cách thiết lập dự án nhanh chóng chỉ trong vài giây
  + Tạo một chuẩn thống nhất cho tất cả các dự án
  + Chạy các dự án song song (Maven 3)
  1. **Gradle**
  + 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.
  + Đặc điểm:
    - Rất linh hoạt
    - Mạnh trong việc xây dựng các đa dự án
    - Quản lí các phụ thuộc (dependency) rất tốt (dựa trên Apache Ivy)
    - Sử dụng DST,Groovy là build script
    - Là một mô hình phong phú để miêu tả dự án
  1. **Junit**
  + JUnit là một framework đơn giản dùng cho việc tạo các unit testing tự động, và chạy các test có thể lặp đi lặp lại. Nó chỉ là một phần của họ kiến trúc xUnit cho việc tạo các unit testing. JUnit là một chuẩn trên thực tế cho unit testing trong Java.
  + Các test case của JUnit là các lớp của java, các lớp này bao gồm một hay nhiều các phương thức unit testing, và những test này được nhóm thành các Test Suite.
  + Các test trong JUnit có thể là các test được chấp nhận hay thất bại, các test này được thiết kế để khi chạy mà không có sự can thiệp của con người. Từ những thiết kế như thế, bạn có thể thêm các bộ test vào quá trình tích hợp và xây dựng phần mềm một cách liên tục và để cho các test chạy một cách tự động.
  + JUnit có những đặc điểm đáng lưu tâm như sau:
    - Xác nhận (assert) việc kiểm tra kết quả được mong đợi
    - Các Test Suite cho phép chúng ta dễ dàng tổ chức và chạy các test
    - Hỗ trợ giao diện đồ họa và giao diện dòng lệnh
    - Junit test framework cung cấp tính năng quan trọng sau đây:
      * Fixtures
      * Test suites
      * Test runners
      * JUnit classes
  1. **Jenkins**

Là một web application mã nguồn mở đóng vai trò máy chủ build & test của hệ thống tích hợp liên tục.

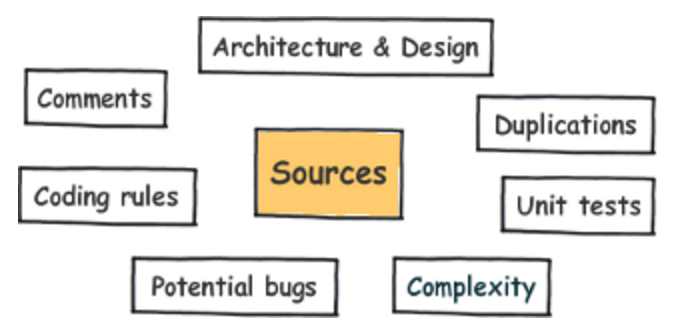
Jenkins được viết bằng Java nên Jenkins có thể kết hợp được với hầu hết các công cụ khác của hệ thống tích hợp liên tục với nhiều nền tảng khác nhau và đặc biệt là tảng mobi application (androi**d**).

Jenkins là ứng dụng dùng để theo dõi các dự án phần mềm rất hiệu quả, nó giúp cho người dùng có thể liên tục cập nhật về tình hình của dự án thông qua procmail. Với giao diện đơn giản, dễ dàng sử dụng và tích hợp với mọi Server lưu trữ.

Dùng để kiểm tra các lỗi và fixbug kịp thời khi mà người dùng chỉnh sửa và upload source code lên server dẫn đến việc trùng lặp hay thiếu các thư viện của project.

**Đặc điểm của Jenkins:**

* + Dễ dàng cài đặt và sử dụng.
  + Đa nền tảng.
  + Hỗ trợ cho nhiều công nghệ phát triển phần mềm.
  + Quá trình build và test hoàn toàn tự động.
  + Dễ dàng liên kết với các công cụ khác của hệ thống tích hợp liên tục thông qua các plug in.
  + Miễn phí.
  1. **Sonar**
* Khái niệm:
  + SonarQube là một nền tảng mã nguồn mỡ dùng để quản lý chất lượng mã nguồn. Nó liên tục phân tích và đo lường chất lượng kỹ thuật, từ danh mục dự án cho đến các phương pháp
* Tính năng:
  + SonarQube là một ứng dụng dựa trên nền web, cung cấp các công cụ giúp phân tích mã nguồn, tìm kiếm báo cáo lỗi.
  + Cung cấp hơn 50 plugins để cộng đồng phát triển có thể mở rộng chức năng.
  + SonarQube quản lý 7 vấn đề của chất lượng mã nguồn



* + 250,000+ tải. 25,000+ cài đặt. 1,500+ cộng đồng thành viên.
  + Sản phẩm và dịch vụ SonarSource được sử dụng bởi nhiều khách hang trên toàn thế giới.
  + Ghi lại lịch sử các số liệu và cung cấp biểu đồ tiến hóa ("cỗ máy thời gian") và quan điểm khác biệt.
  + Cung cấp các phân tích hoàn toàn tự động: tích hợp với Maven, Ant, [Gradle](https://en.wikipedia.org/wiki/Gradle) và [hội nhập liên tục](https://en.wikipedia.org/wiki/Continuous_integration) các công cụ [(Atlassian](https://en.wikipedia.org/wiki/Bamboo_(software))Bamboo, Jenkins, [Hudson, vv).](https://en.wikipedia.org/wiki/Hudson_(software))
  + Tích hợp với  môi trường phát triển [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)) .
  + Tích hợp với các công cụ bên ngoài: JIRA, Mantis, LDAP, [Fortify, vv](https://en.wikipedia.org/wiki/Fortify_Software)
  + Được mở rộng với việc sử dụng các plugins.
  + [Thực hiện](http://www.sonarsource.com/products/plugins/governance/sqale/) các [SQALE](https://en.wikipedia.org/wiki/SQALE) phương pháp để tính toán [nợ kỹ](https://en.wikipedia.org/wiki/Technical_debt)thuật

1. **So sánh:**
   1. **Maven vs Gradle vs Junit**
   2. **Selenium vs SOAP UI vs Sahi vs Watir**
   3. **Jenkins vs Sonar**