KIỂM THỬ PHẦN MỀM

Số tín chỉ: 3

Lý thuyết: 30 tiết

Thực hành: 30 tiết

|  |  |
| --- | --- |
| **Bài tập** | **Điểm** |
| Bài tập 1: Kiểm thử hộp trắng   * Kiểm thử API * Kiểm thử dòng dữ liệu * Kiểm thử cấu trúc điều khiển | 10% |
| Bài tập 2: Kiểm thử hộp đen   * Kiểm thử giá trị biên * Kiểm thử lớp tương đương * Kiểm thử bằng bảng quyết định | 15% |
| Thi giữa kỳ: Thi máy   * Nắm rõ về kiểm thử tự động và kiến trúc của bộ kiểm thử tự động * Sử dụng một số công cụ kiểm thử tự động   NUnit  JUnit  Moq | 15% |
| Thi cuối kỳ: Vấn đáp | 60% |

Bài lab:

* Lab 1 & 2: Kiểm thử
* Lab 3 & 4: Thiết kế Test Case
* Lab 5 & 6: Kiểm thử trên dự án
* Lab 7: Kiểm thử tích hợp

Trang tham khảo: <https://www.toolsqa.com/software-testing-tutorial>

* **Bài tập tình huống 1**: Kiểm tra chương trình tính số giai thừa

Contents

[GIỚI THIỆU 3](#_Toc39934579)

[PHƯƠNG PHÁP KIỂM THỬ VÀ YẾU TỐ KIỂM THỬ 8](#_Toc39934580)

[I. Phương pháp kiểm thử: 8](#_Toc39934581)

[II. Yếu tố kiểm thử 11](#_Toc39934582)

[QUÁ TRÌNH KIỂM THỬ & VÒNG ĐỜI PHÁT TRIỂN PHẦN MỀM 14](#_Toc39934583)

[I. Quá trình kiểm thử 14](#_Toc39934584)

[II. Vòng đời phát triển phần mềm - Software Development Life Cycle (SDLC) 16](#_Toc39934585)

[MỨC ĐỘ VÀ PHƯƠNG PHÁP KIỂM THỬ 21](#_Toc39934586)

[1. Kiểm thử chức năng 21](#_Toc39934587)

[2. Kiểm thử phi chức năng 23](#_Toc39934588)

[3. Các phương pháp kiểm thử 23](#_Toc39934589)

[KIỂM THỬ TĨNH VÀ KIỂM THỬ ĐỘNG 25](#_Toc39934590)

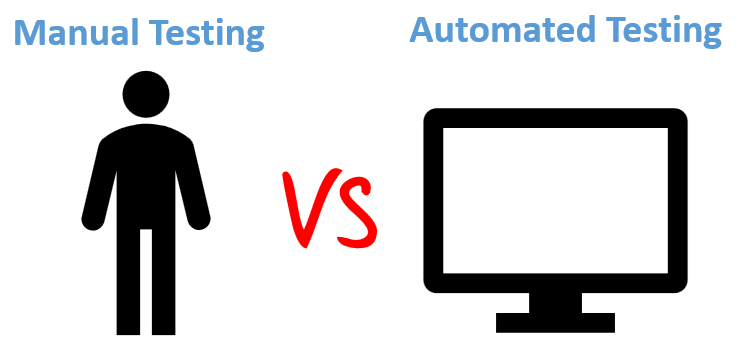
[1. Kiểm thử động 25](#_Toc39934591)

[2. Kiểm thử tĩnh 26](#_Toc39934592)

# GIỚI THIỆU

* Khái niệm *kiểm thử phần mềm* là quá trình xác định tính chính xác và chất lượng (correctness and quality) của phần mềm.
* Nói cách khác, kiểm thử phần mềm là thực thi hệ thống hoặc ứng dụng để tìm **bugs**, **defects** hoặc **errors**.
* *Ví dụ: Car manufacturer tests the car for maximum speed, fuel efficiency and safety from crash.These test later become the part of advertising strategy for car sales.*
* A close up of a sign

  Description automatically generatedMột số thuật ngữ:
* **Error**: Lỗi khi viết chương trình.
* **Defect**: Error được tìm thấy bởi tester. Có sự khác nhau giữa kết quả thực tế và kết quả mong đợi cũng được gọi là defect.
* **Bug**: Defect được chấp nhận bởi đội ngũ phát triển, sau khi nó được gọi là bug/anomaly.
* **Failure**: Một defect mà khách hàng gặp phải.
* **Missing and Wrong**: Yêu cầu của khách hàng không được cung cấp đầy đủ.
* **Fault**: Fault làm cho ứng dụng hoạt động sai.
* Kiểm thử phần mềm rất quan trọng.
* **Cách kiểm thử**: kiểm thử thủ công và kiểm thử tự động.



* **Các yếu tố kiểm thử (Test Artifacts)**: basic testing; test specification; test case; test scenario.



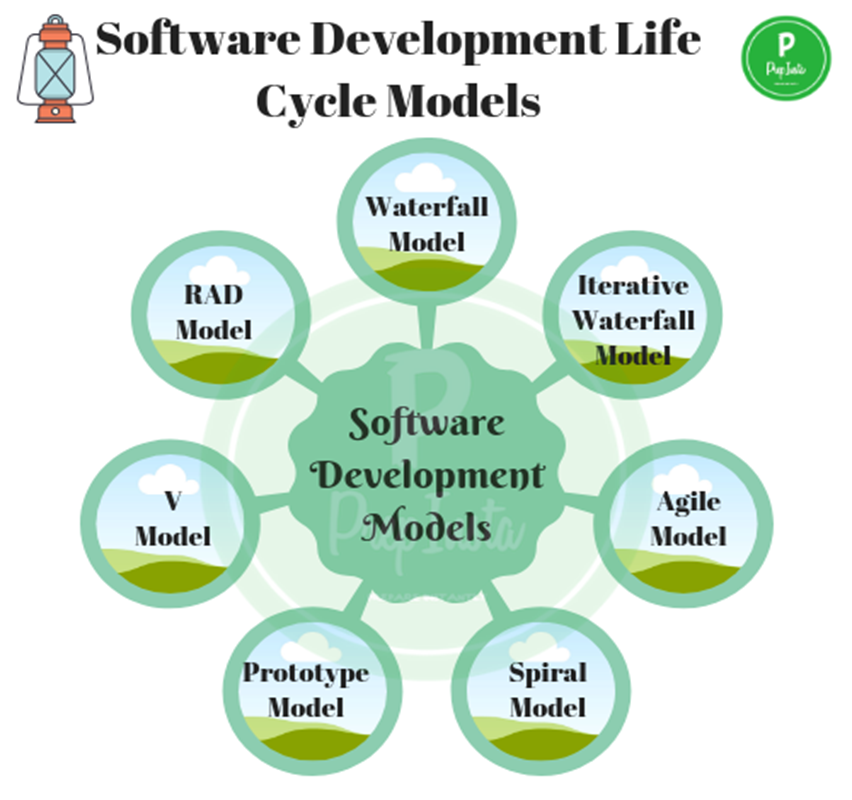
* **Vòng đời phát triển phần mềm**:



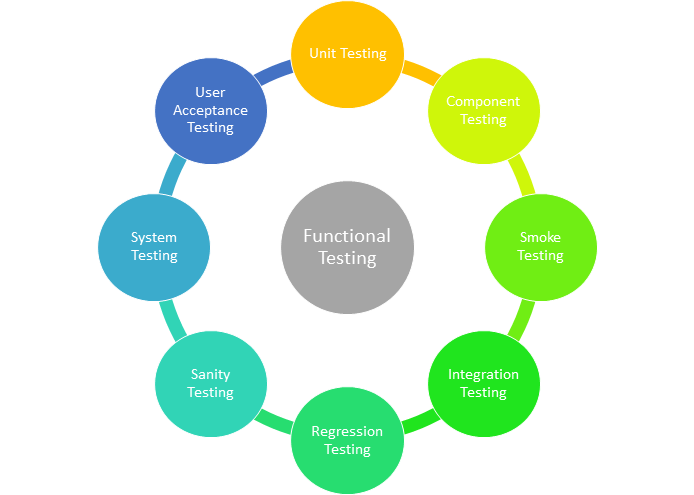
* **Vòng đời phát triển phần mềm:**



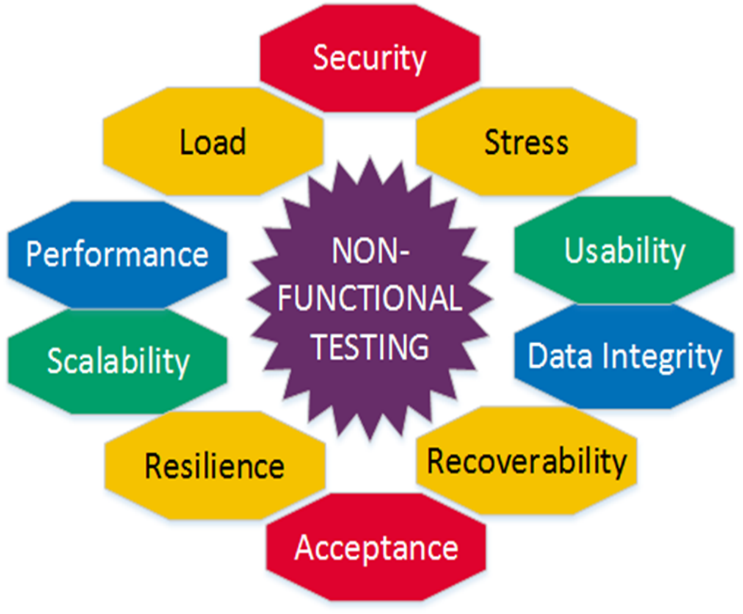
* **Mô hình phát triển phần mềm**: thác nước, RAD, V, nguyên mẫu, xoắn ốc, Agile, Iterative Waterfall.



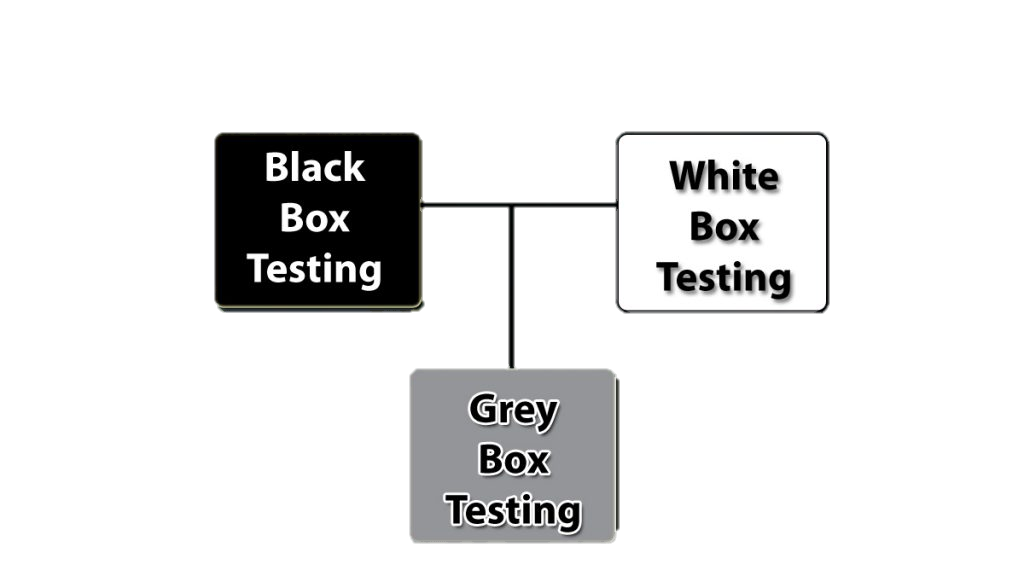
* **Cấp độ kiểm thử**: functional testing; non functional testing.
  + **Functional testing**:



* + **Non Functional Testing:**

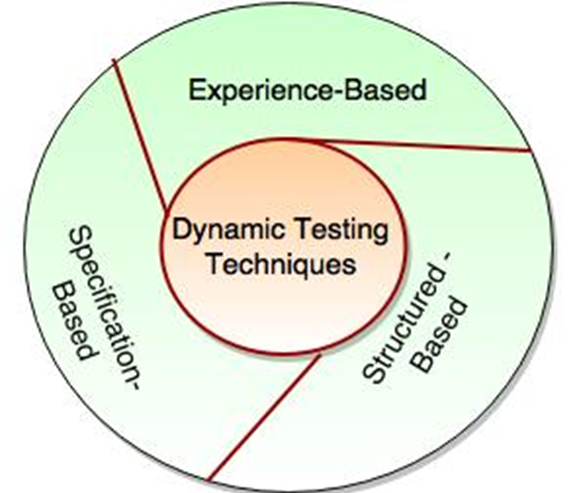
****

* **Phương pháp kiểm thử**: kiểm thử hộp đen (kiểm thử chương trình sau khi biên dịch); kiểm thử hộp trắng (kiểm thử chương trình tại mã nguồn); kiểm thử hộp xám.



* **Kỹ thuật kiểm thử động**: Kiểm thử khi chương trình thực thi.

Bao gồm: *dựa trên kinh nghiệm; dựa trên cấu trúc; dựa trên đặc tả (specification)*.

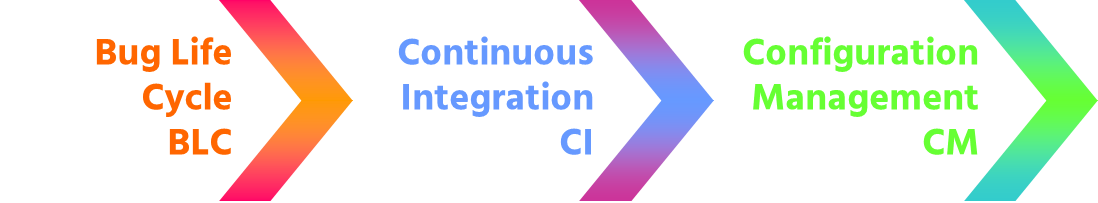


* **Kỹ thuật kiểm thử tĩnh**: Kiểm thử khi chương trình không chạy.

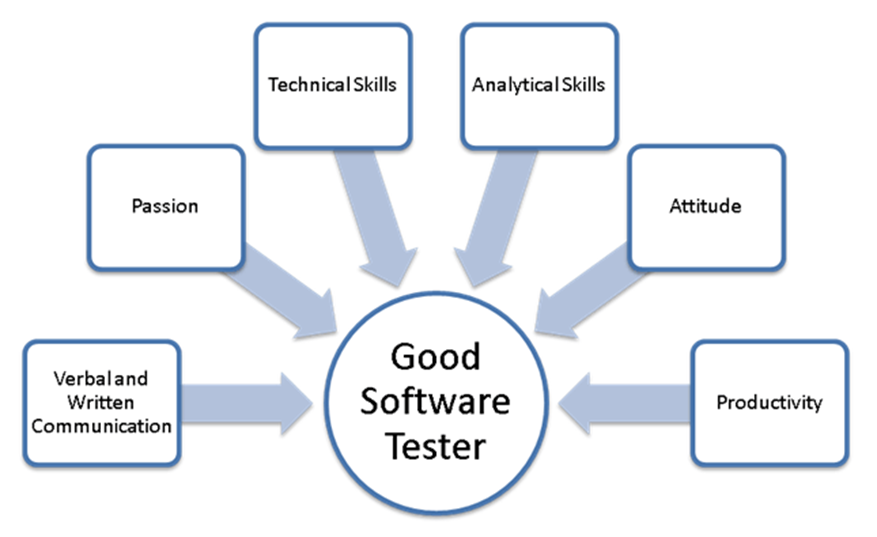
Bao gồm: *Informal, Walkthrough, Peer Review, Inspection*.



* **Quản lý kiểm thử**: *Bug Life Cycle; Continuous Intergration; Configuration Management*.



* **Một người kiểm thử tốt**:



# PHƯƠNG PHÁP KIỂM THỬ VÀ YẾU TỐ KIỂM THỬ

## Phương pháp kiểm thử:

* + Manual Testing: các test case được thực thi thủ công.
  + Automation Testing: kiểm tra được thực hiện với sự trợ giúp của các công cụ tự động.

**\* Manual testing:**

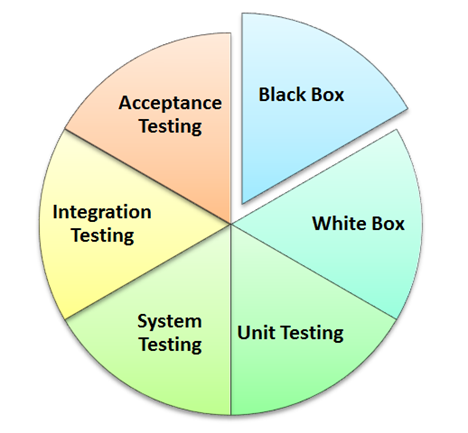
* + Người kiểm thử đóng vai trò quan trọng của người dùng cuối và xác nhận rằng các tính năng của ứng dụng là làm việc chính xác.
  + Người kiểm thử thực hiện các test case thủ công mà không sử dụng bất kỳ công cụ tự động nào.



🡺 KPI: "Key Performance Indicator" là chỉ số đo lường và đánh giá hiệu quả hoạt động của một bộ phận trong một công ty hoặc sự vận hành của cả công ty. Mỗi bộ phận trong công ty sẽ có những chỉ số KPI khác nhau.



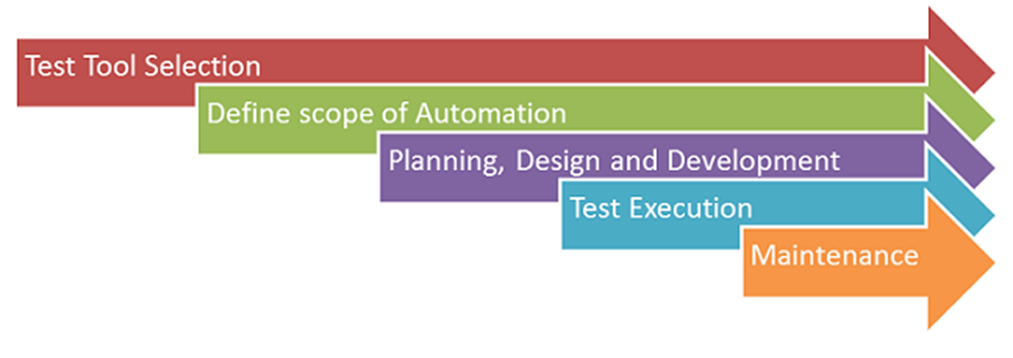
* Mục tiêu của Manual Testing:
  + Đảm bảo ứng dụng không bị lỗi và hoạt động và nó hoạt động phù hợp với các yêu cầu chức năng được chỉ định;
  + Đảm bảo các báo cáo khiếm khuyết được sửa bởi nhà phát triển và re-testing.
* Phân loại Manual Tesing:



* Các bước kiểm thử thủ công:
  + Đọc và hiểu tài liệu/hướng dẫn dự án phần mềm, ngoài ra học thêm Application Under Test (AUT).
  + Tạo bản nháp bao trùm toàn bộ yêu cầu của tài liệu.
  + Xem lại và baseline các test case với team leader, client.
  + Thực thi các test case AUT.
  + Báo cáo lỗi.
  + Khi lỗi được sửa thì thực thi test case để xác minh nó vượt qua (pass).

**\* Automation testing:**

* Kiểm tra được thực hiện với sự trợ giúp của các công cụ tự động.
* Quá trình kiểm thử tự động:



* Lợi ích:
  + Nhanh hơn 70% so với thử nghiệm thủ công.
  + Đáng tin cậy trong kết quả.
  + Đảm bảo tính nhất quán.
  + Tiết kiệm thời gian và chi phí.
  + Cải thiện độ chính xác.
  + Không cần có sự can thiệp của con người trong khi thực hiện.
  + Kịch bản kiểm tra có thể sử dụng lại.
  + Kiểm tra thường xuyên và kỹ lưỡng.
  + Nhiều chu kỳ thực hiện có thể đạt được thông qua tự động hóa.
* Các công cụ kiểm thử tự động:
  + Ranorex Studio
  + Mabl
  + Selenium
  + QTP (MicroFocus UFT)
  + Rational Functional Tester
  + WATIR
  + SilkTest

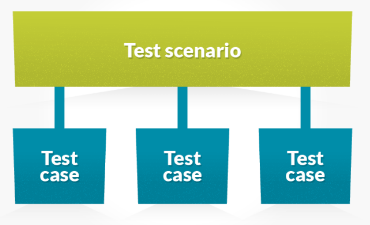
**\* Bài tập tình huống:**

- Nêu sự giống nhau và khác nhau giữa Kiểm thử thủ công và Kiểm thử tự động.

- Tìm một vài công cụ cho phép kiểm thử tự động (miễn phí) và công cụ đó phải có hướng dẫn sử dụng.

## Yếu tố kiểm thử

* Test Basis:
  + Khái niệm:
    - Cơ sở cho các bài kiểm tra được gọi là Test basis;
    - Nó có thể là một yêu cầu hệ thống, một đặc tả kỹ thuật hoặc một quá trình kinh doanh.
  + Yêu cầu:
    - Người kiểm thử nhìn vào Test Basis để xem những gì có thể kiểm tra.
    - Xem xét test basis rất là quan trọng của mô hình V ở SDLC
* Lý do đặc tả (Specification):
  + Thử nghiệm có những hạn chế nghiêm trọng và hiệu quả của thử nghiệm phụ thuộc rất nhiều vào bản chất chính xác của trường hợp thử nghiệm.
  + Xây dựng một Test Case tốt sẽ tiết lộ lỗi trong các chương trình là một hoạt động rất sáng tạo và phụ thuộc vào người thử nghiệm.
* Định danh đặc tả:
  + Mục tiêu của Test Case: Mục đích của thử nghiệm;
  + Mục thử nghiệm: Các mục (ví dụ: thông số kỹ thuật yêu cầu, thông số kỹ thuật thiết kế, mã, v.v.) được yêu cầu để chạy một trường hợp thử nghiệm cụ thể.
  + Thông số kỹ thuật đầu vào: Mô tả những gì được yêu cầu (từng bước) để thực hiện trường hợp thử nghiệm (ví dụ: các tệp đầu vào, các giá trị phải được nhập vào một trường, v.v.).
  + Thông số kỹ thuật đầu ra: Mô tả hệ thống sẽ trông như thế nào sau khi trường hợp thử nghiệm được chạy.
  + Nhu cầu môi trường: Mô tả bất kỳ nhu cầu môi trường đặc biệt. Điều này bao gồm kiến ​​trúc hệ thống, công cụ Phần cứng & Phần mềm, hồ sơ hoặc tệp, giao diện, v.v.
* Kịch bản (scenario):
  + KỊCH BẢN KIỂM TRA được định nghĩa là bất kỳ chức năng nào có thể được kiểm tra.
  + Nó cũng được gọi là Điều kiện thử nghiệm hoặc Khả năng thử nghiệm.



* Các bước viết kịch bản:
  + Đọc tài liệu yêu cầu của ứng dụng cần được kiểm tra.
  + Với mỗi yêu cầu chỉ ra các hành động người dùng có thể và mục đích của hành động đó. Xác định các khía cạnh kỹ thuật của yêu cầu. Xác định các tình huống có thể xảy ra về lạm dụng hệ thống và đánh giá người dùng bằng tư duy của hacker.
  + Sau khi đọc Tài liệu yêu cầu và làm phân tích, làm một danh sách các kịch bản cho mỗi tính năng của phần mềm.
  + Ma trận truy xuất được tạo để xác minh mỗi yêu cầu có tương ứng với kịch bản không.
  + Kịch bản được xem xét bởi người giám sát, stakeholders.
* Mẹo để tạo kịch bản thử nghiệm:
  + Mỗi Kịch bản thử nghiệm phải *được gắn với tối thiểu một Yêu cầu* hoặc Người dùng.
  + Xác minh nhiều Yêu cầu cùng một lúc, đảm bảo bạn có Kịch bản kiểm tra kiểm tra yêu cầu *đó một cách riêng biệ*t.
  + Tránh tạo các Kịch bản kiểm tra *quá phức tạp* kéo dài nhiều Yêu cầu.
  + Số lượng kịch bản có thể lớn và tốn kém để chạy. Dựa trên các ưu tiên của khách hàng, chỉ chạy các *Kịch bản thử nghiệm* được chọn.
* Ví dụ 1: Ứng dụng thương mại điện tử

Kịch bản 1: Kiểm tra chức năng đăng nhập

* + Test case:
    - Email đúng, mật khẩu đúng.
    - Email sai và mật khẩu đúng.
    - Email đúng và mật khẩu sai.
    - Email sai và mật khẩu sai.
    - Email và mật khẩu có khoảng trắng đầu.
    - Chức năng quên mật khẩu có hoạt động như mong đợi (nếu phức tạp thì tách thành một kịch bản riêng).
    - Nhập số điện thoại và mật khẩu đúng/sai (lặp lại 5 test case trên).
    - Kiểm tra chức năng Giữ đăng nhập.

Kịch bản 2: Kiểm tra chức năng tìm kiếm

Kịch bản 3: Kiểm tra trang mô tả sản phẩm

Kịch bản 4: Kiểm tra thanh toán

* Ví dụ 2: Trang ngân hàng

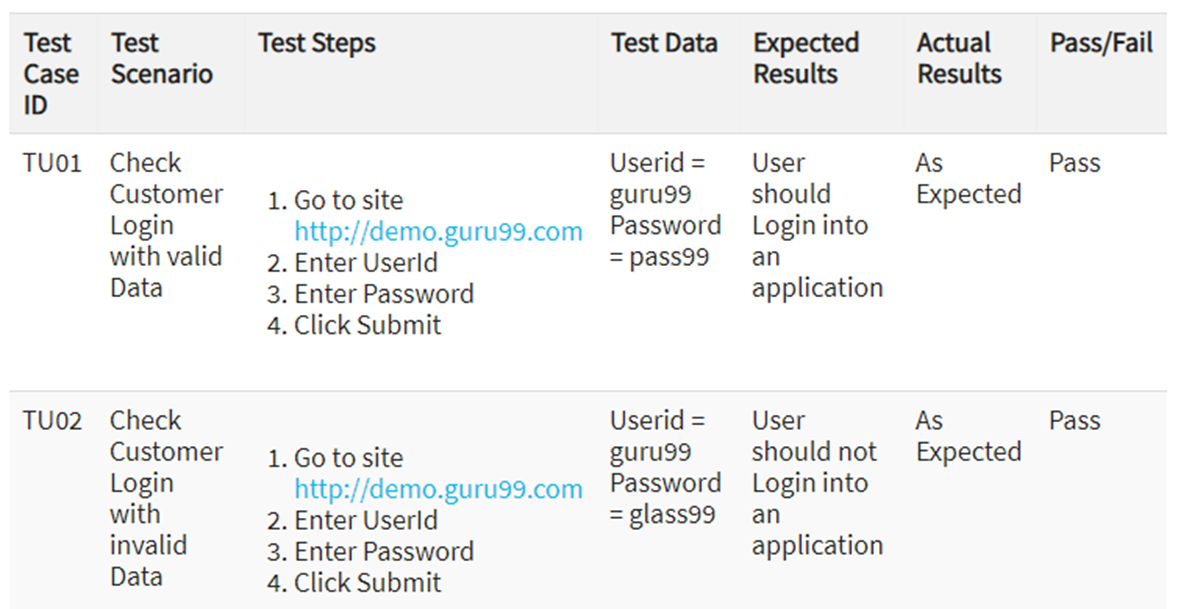
Kịch bản 1: Kiểm tra đăng nhập và chứng thực.

Kịch bản 2: Kiểm tra chuyển tiền có thành công.

Kịch bản 3: Kiểm tra xem tài khoản.

Kịch bản 4. Kiểm tra tiền gửi cố định / tiền gửi định kỳ có thể được tạo.

* **Test case**: Các trường hợp kiểm tra là một tập các hành động được thực thi để xác minh một tính năng cụ thể hoặc chức năng của phần mềm.
* Kịch bản *bao hàm* các test case.
* Các bước tạo test case cho một kịch bản:
  + **Kiểm tra email và mật khẩu hợp lệ.**
  + Để thực thi test case trên, chúng ta cần dữ liệu kiểm tra.
  + Các hành động thực thi dựa trên AUT:
    - Nhập email
    - Nhập mật khẩu
    - Nhấn nút Đăng nhập
  + Mục đích của test case là kiểm tra hành vi AUT có kết quả mong đợi: **Đăng nhập thành công (Pass)**, hoặc **Đăng nhập thất bại (Fail)**.
  + Điều kiện tiên quyết chỉ ra các công việc cần phải làm trước khi kiểm tra.



* Bài tập tình huống 2:

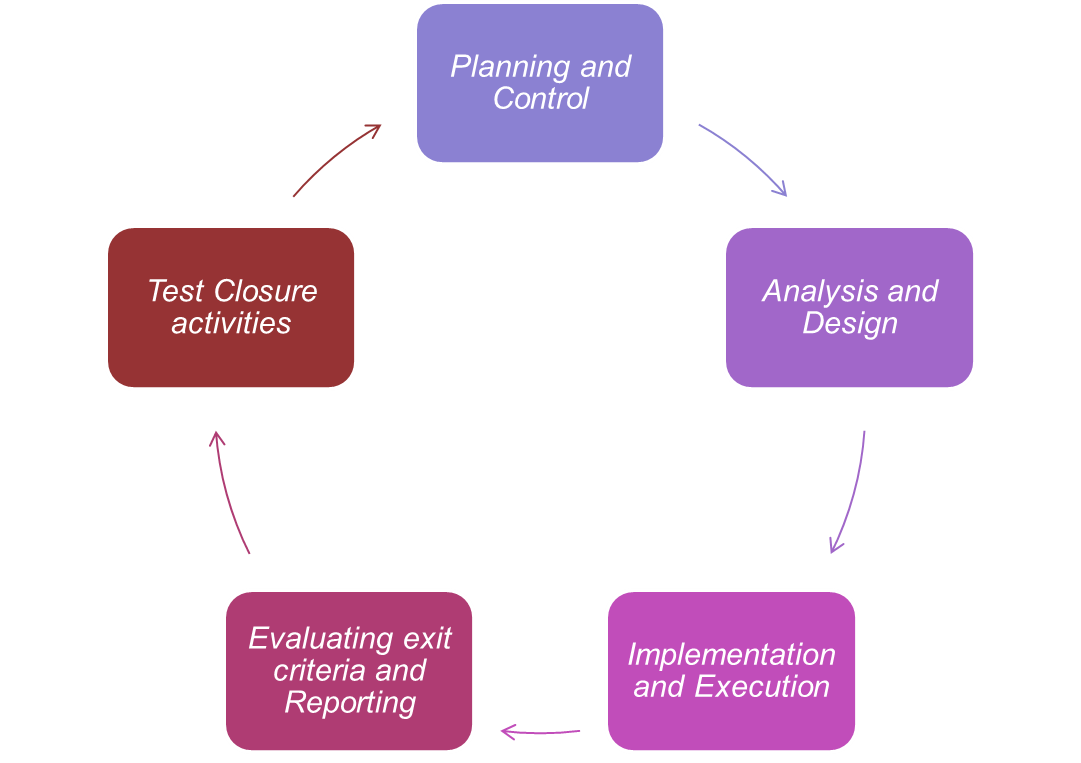
Viết hai kịch bản cho một trang thuộc dlu.edu.vn, sau đó viết các test case.

# QUÁ TRÌNH KIỂM THỬ & VÒNG ĐỜI PHÁT TRIỂN PHẦN MỀM

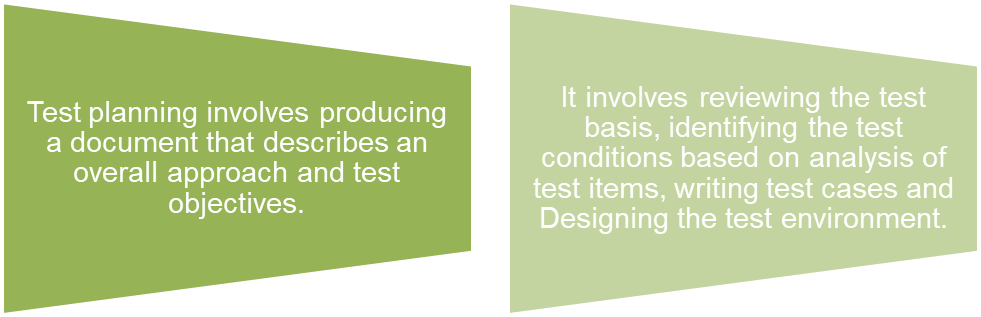
## Quá trình kiểm thử



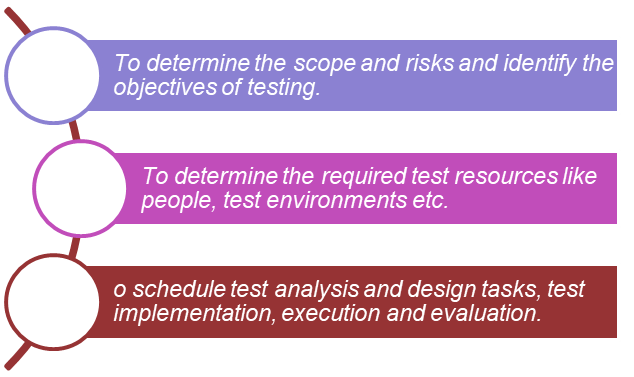
* 7 nguyên tắc kiểm thử phần mềm:
  + Testing Shows Presence of Defects: Kiểm thử cho thấy lỗi
  + Exhaustive Testing is Impossible: Kiểm tra toàn diện là không thể
    - Tốn thời gian
    - Chi phí
  + Early Testing: Kiểm thử sớm
  + Defect Clustering: Phân cụm khiếm khuyết
  + Pesticide Paradox: Nghịch lí thuốc trừ sâu
  + Testing is Context Depending: Kiểm tra phụ thuộc vào ngữ cảnh
  + Absence of Errors Fallacy: Ngụy biện vắng mặt lỗi



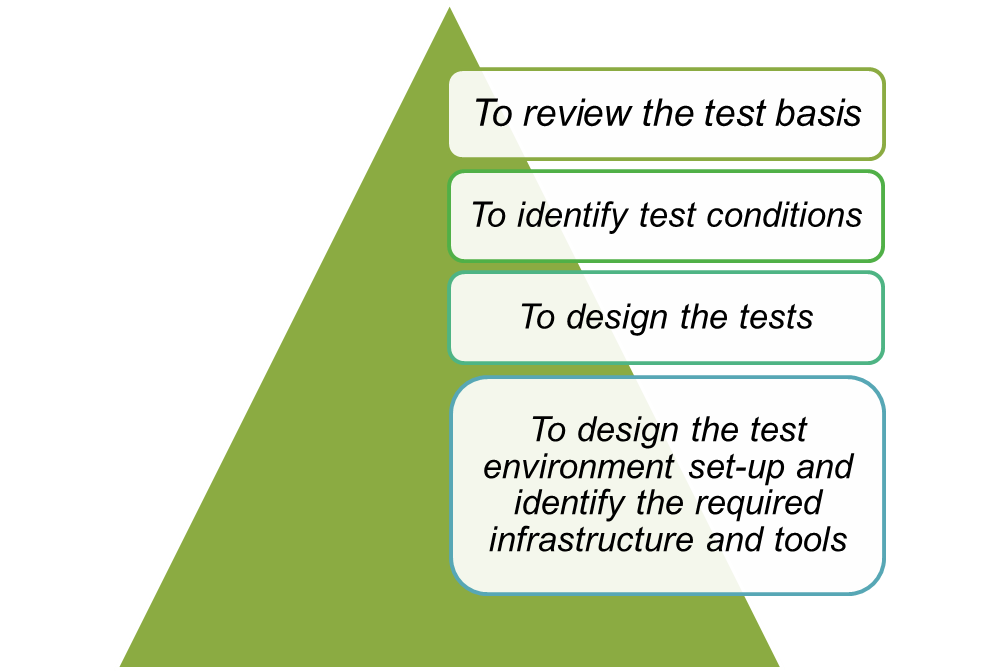
* Kế hoạch và Kiểm soát:
  + Kế hoạch:



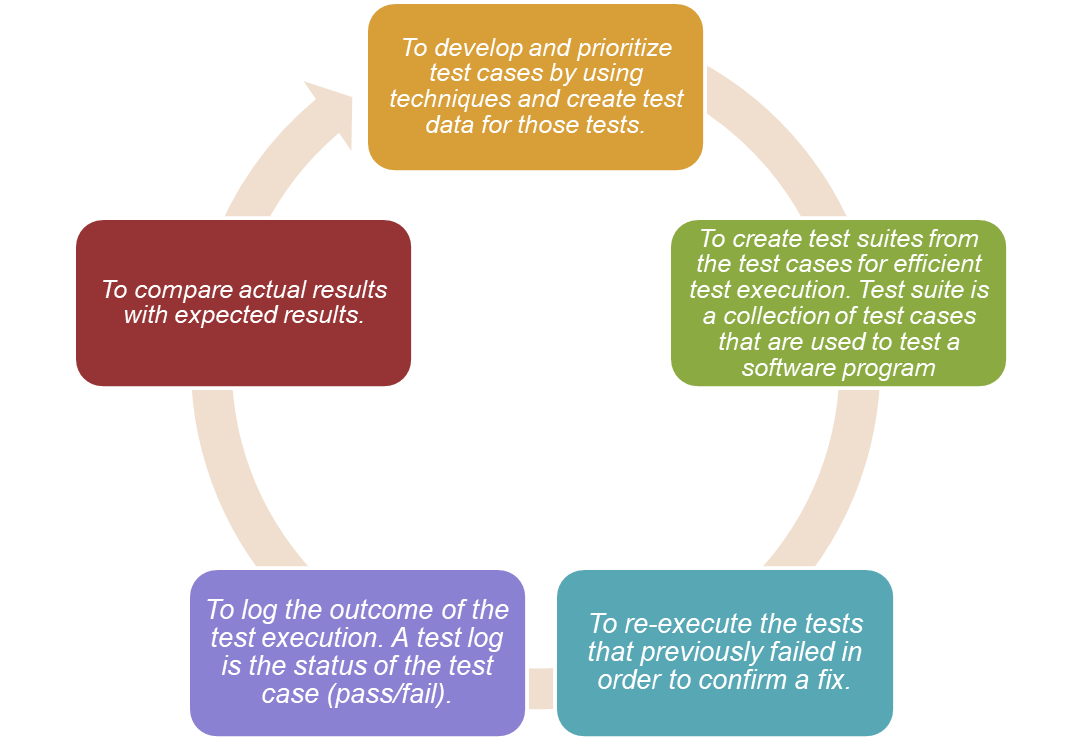
* + Mục đích:



* + Kiểm soát: This is the activity of comparing actual progress against the plan, and reporting the status, including deviations from the plan.
* Phân tích và thiết kế:

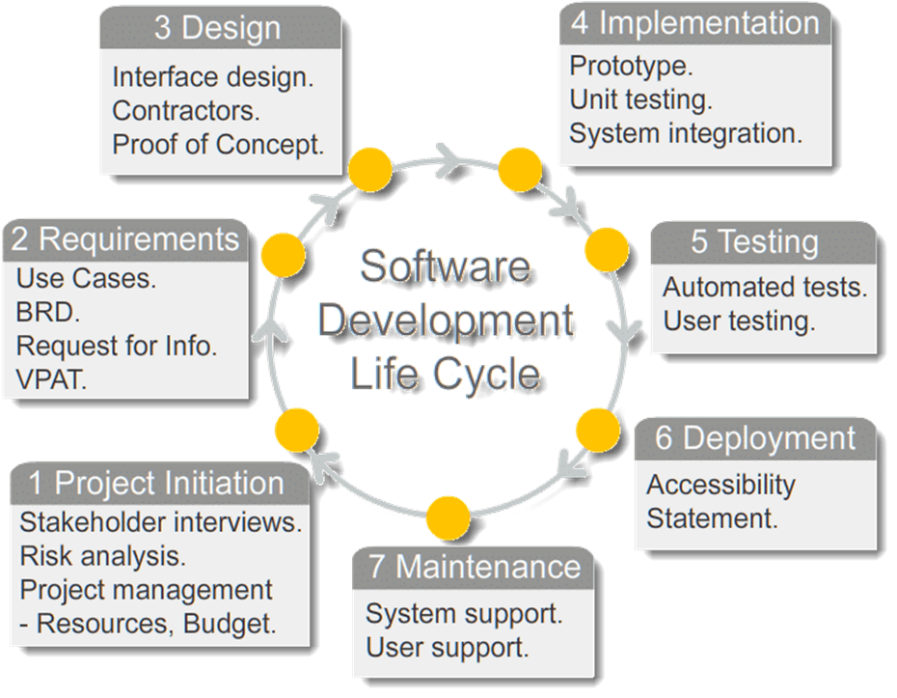


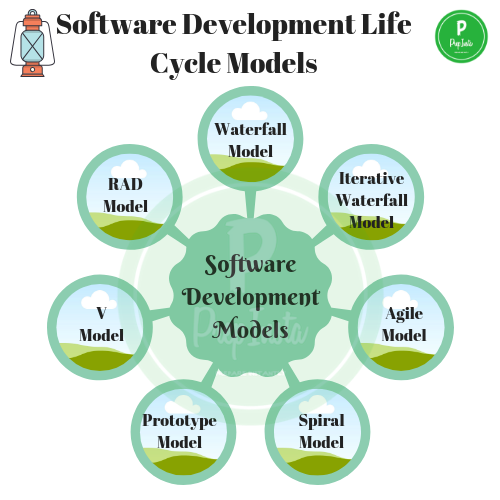
* Triển khai và Thực thi:

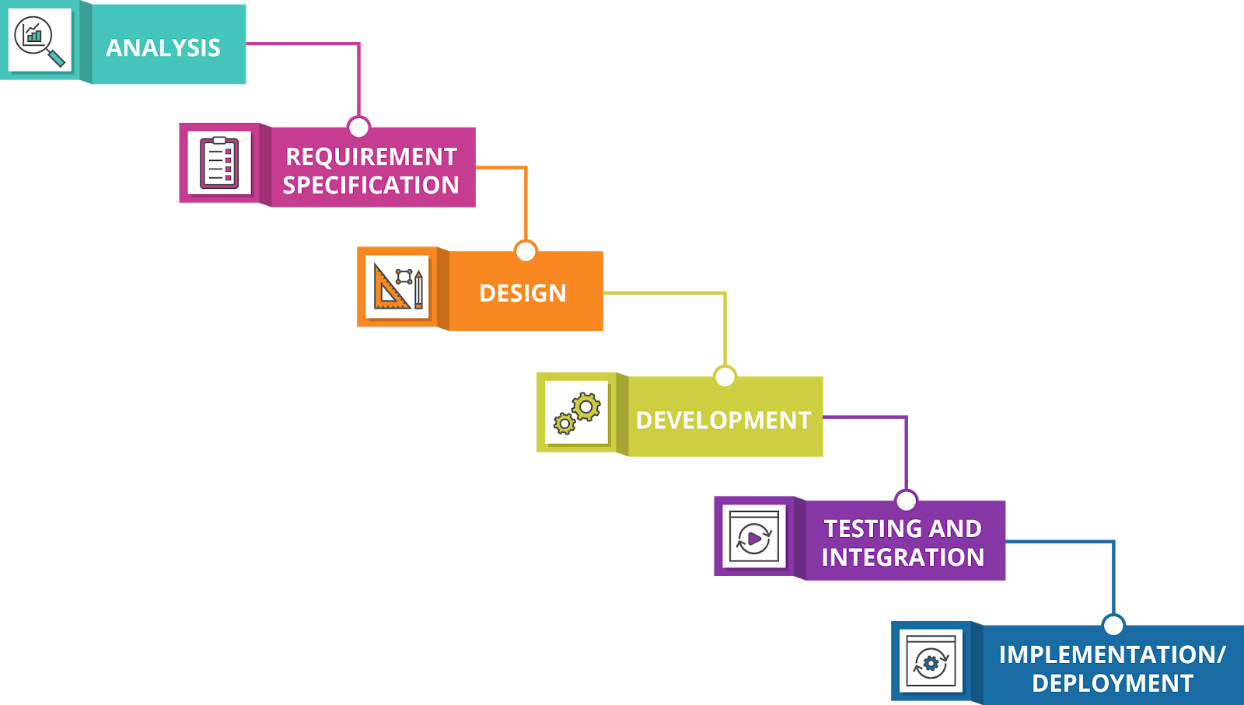


* Đánh giá điều kiện thoát và Báo cáo:
* Hoạt động đóng cửa thử nghiệm:

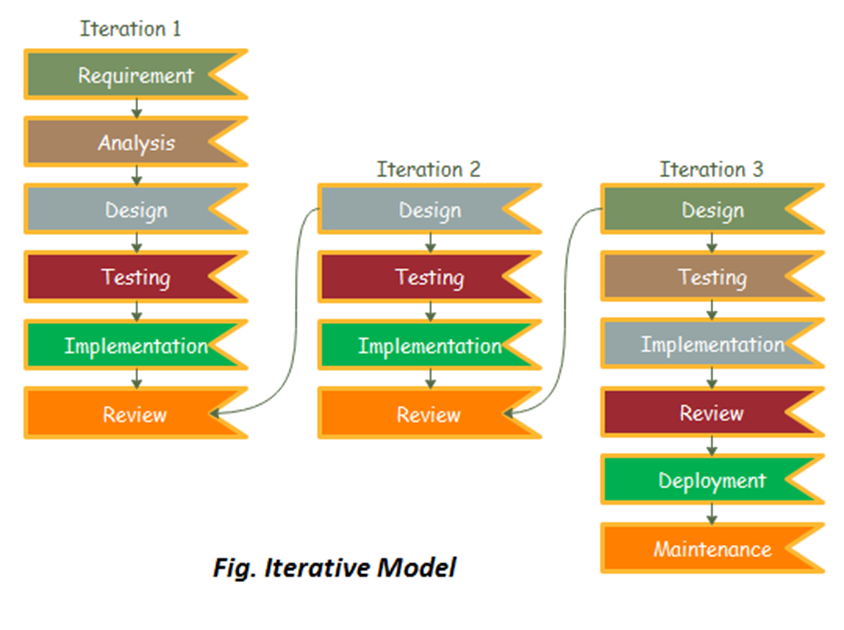
## Vòng đời phát triển phần mềm - Software Development Life Cycle (SDLC)

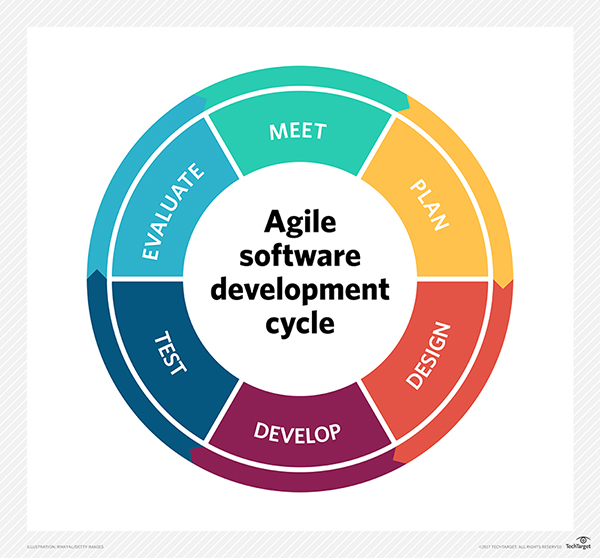


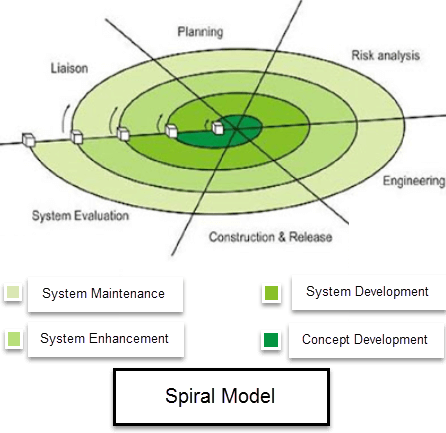


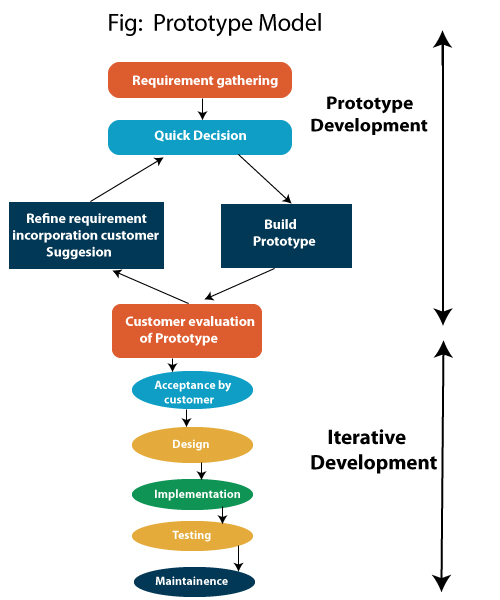


Waterfall Model

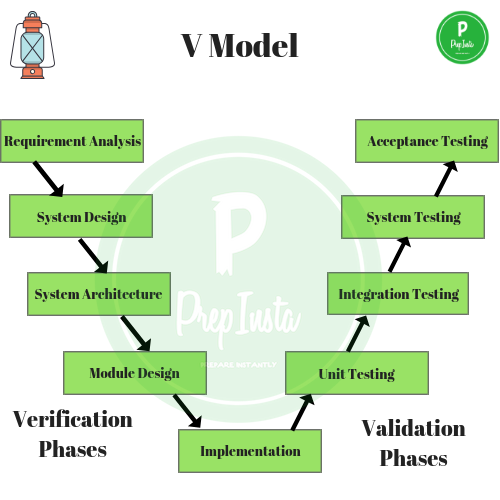


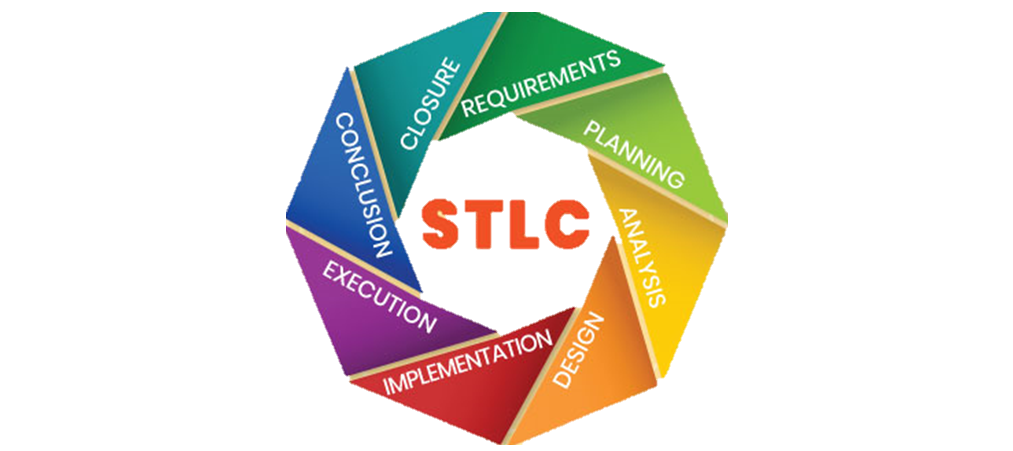






Prototype Model





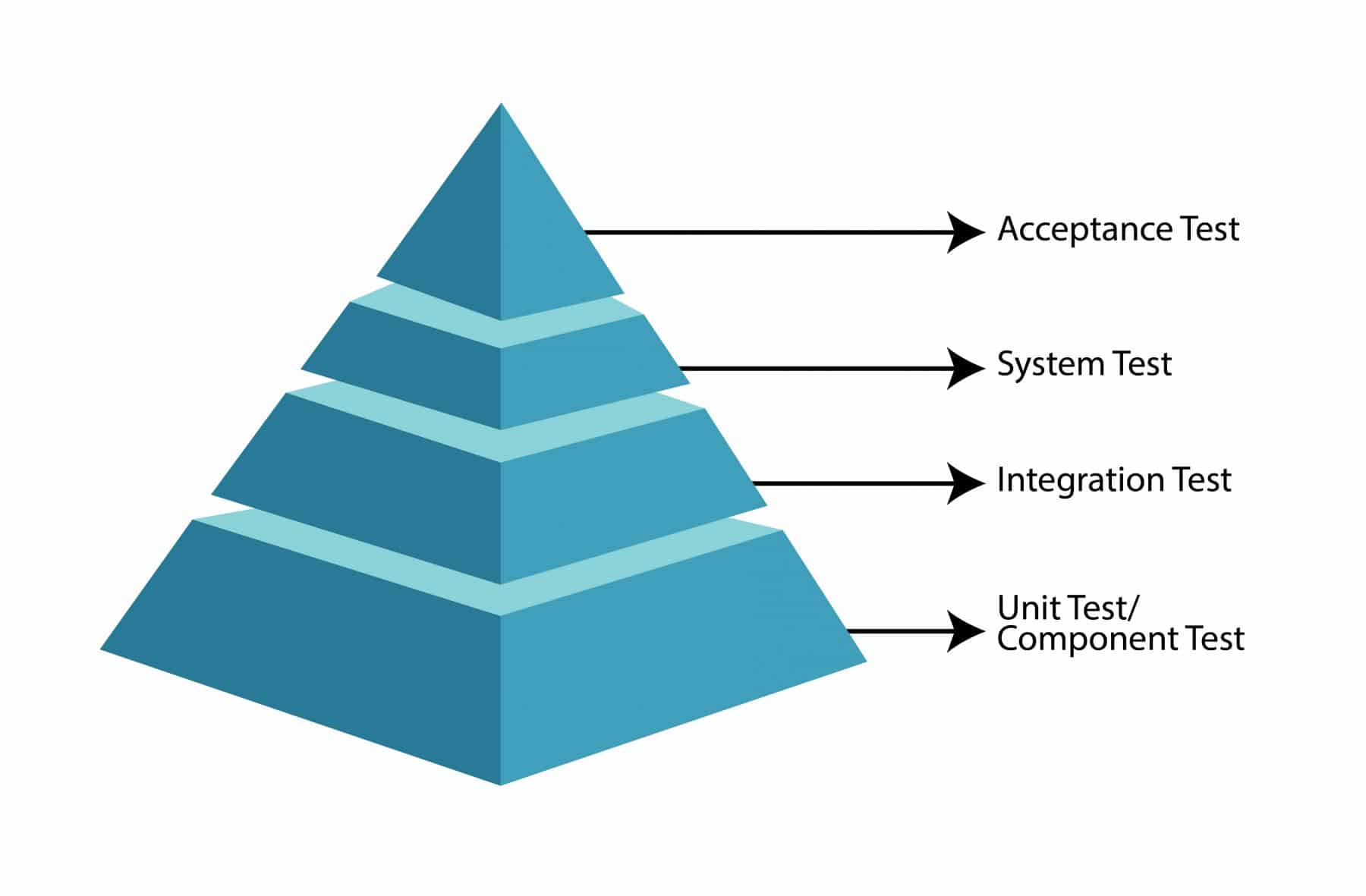
**Software Testing Life Cycle (STLC)**

# MỨC ĐỘ VÀ PHƯƠNG PHÁP KIỂM THỬ

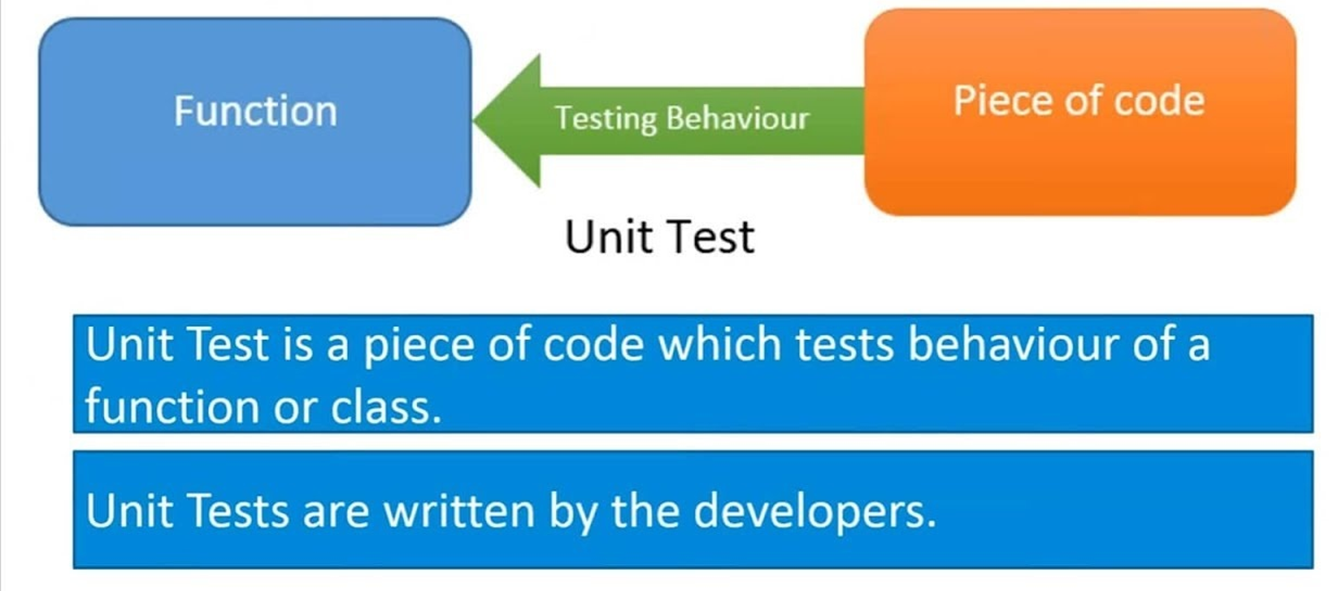
Mức độ kiểm thử: Kiểm thử chức năng và Kiểm thử phi chức năng.

## Kiểm thử chức năng

* Bao gồm:
  + Kiểm thử đơn vị
  + Kiểm thử tích hợp
  + Kiểm thử khói
  + Kiểm thử hệ thống
  + Kiểm thử chấp nhận (alpha và beta)

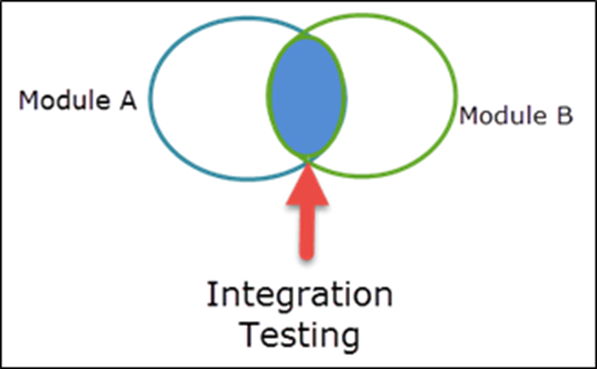


1. Kiểm thử đơn vị



1. Kiểm thử tích hợp

Kiểm thử tích hợp là kiểm thử của một hệ thống con mà bao gồm hai hay nhiều thành phần tích hợp.



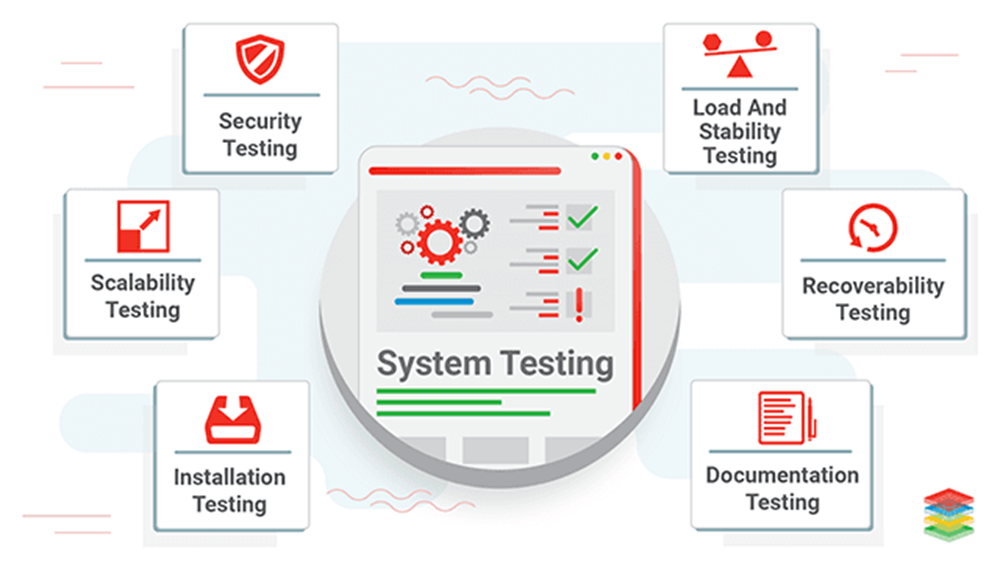
1. Kiểm thử khói

Kiểm thử khói được thực hiện sau khi xây dựng phần mềm để xác định rằng các chức năng quan trọng của chương trình đang hoạt động tốt.

Đội ngũ QA luôn chuẩn bị một tập các test case chức năng để kiểm thử bất kỳ chức năng nào.

Kiểm thử khói cho phép người kiểm thử thực thi các test case.

1. Kiểm thử hệ thống

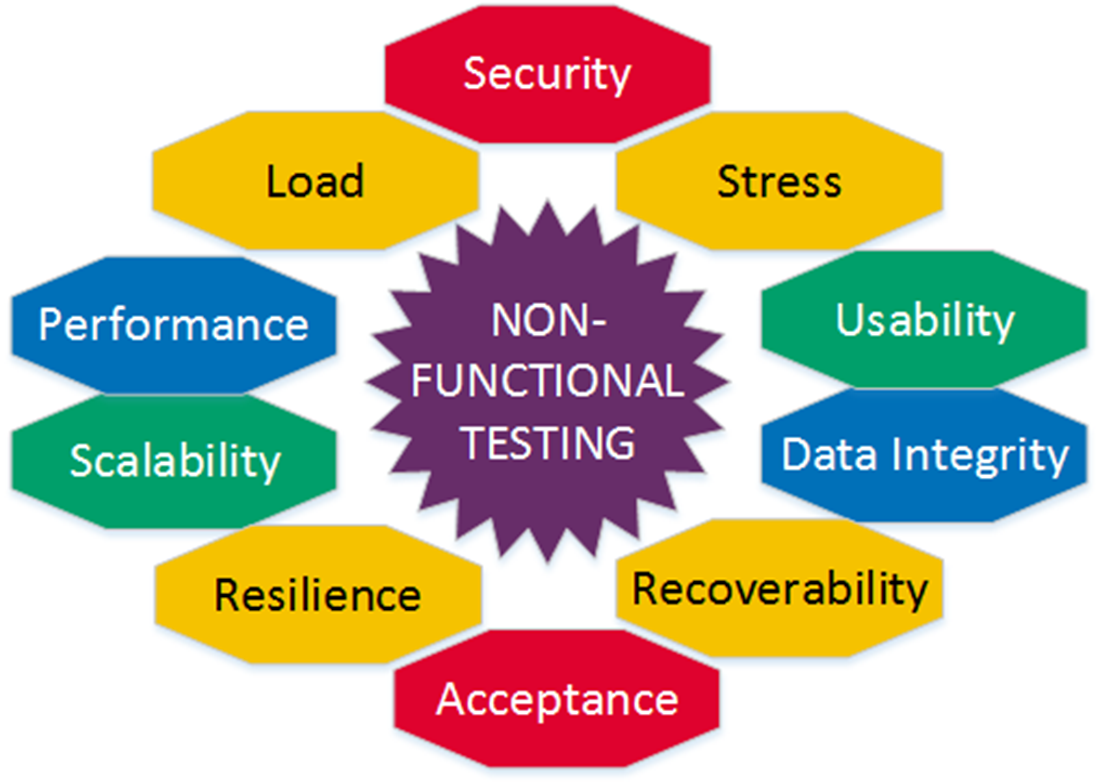


Kiểm tra hệ thống có nghĩa là kiểm tra toàn bộ hệ thống. Nói cách khác, tất cả các mô-đun / thành phần được tích hợp để xác minh xem hệ thống có hoạt động như mong đợi hay không.

1. Kiểm thử chấp nhận người dùng

* Kiểm thử alpha: Chỉ kiểm thử trong phạm vi của công ty
* Kiểm thử beta: Kiểm thử ở phía người dùng thực sự

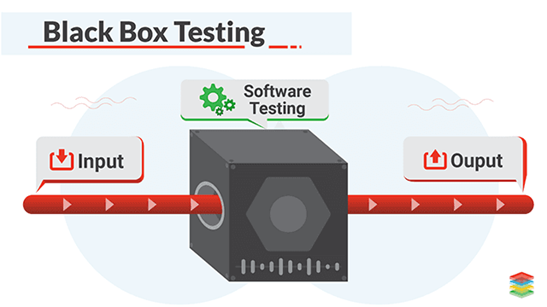
## Kiểm thử phi chức năng



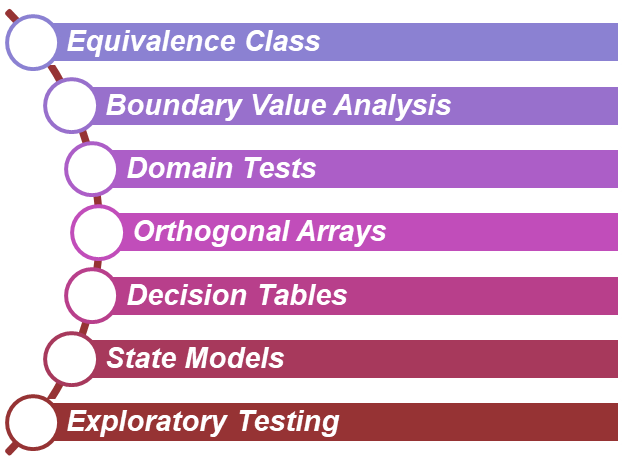
## Các phương pháp kiểm thử

1. Kiểm thử hộp đen

* Phương pháp Kiểm thử mà không cần biết các đoạn mã bên trong và cấu trúc chương trình.



* Xuất phát từ quan điểm của người dùng/ cách nhìn của người dùng đến ứng dụng. Người kiểm thử chỉ biết về đầu vào và đầu ra của ứng dụng.
* Ví dụ: tìm kiếm trên Google



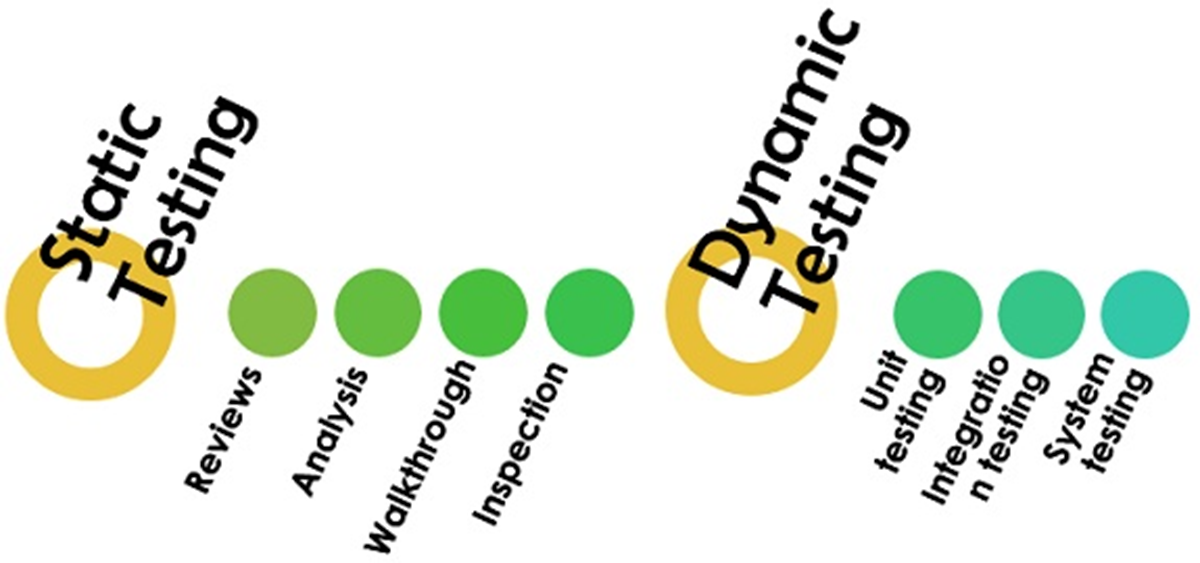
1. Kiểm thử hộp trắng

* Kiểm thử biết đến đoạn mã nội bộ và cấu trúc chương trình.
* Người kiểm thử chọn đầu vào và xác định các đầu ra thích hợp.

1. Kiểm thử hộp nâu (gray box)

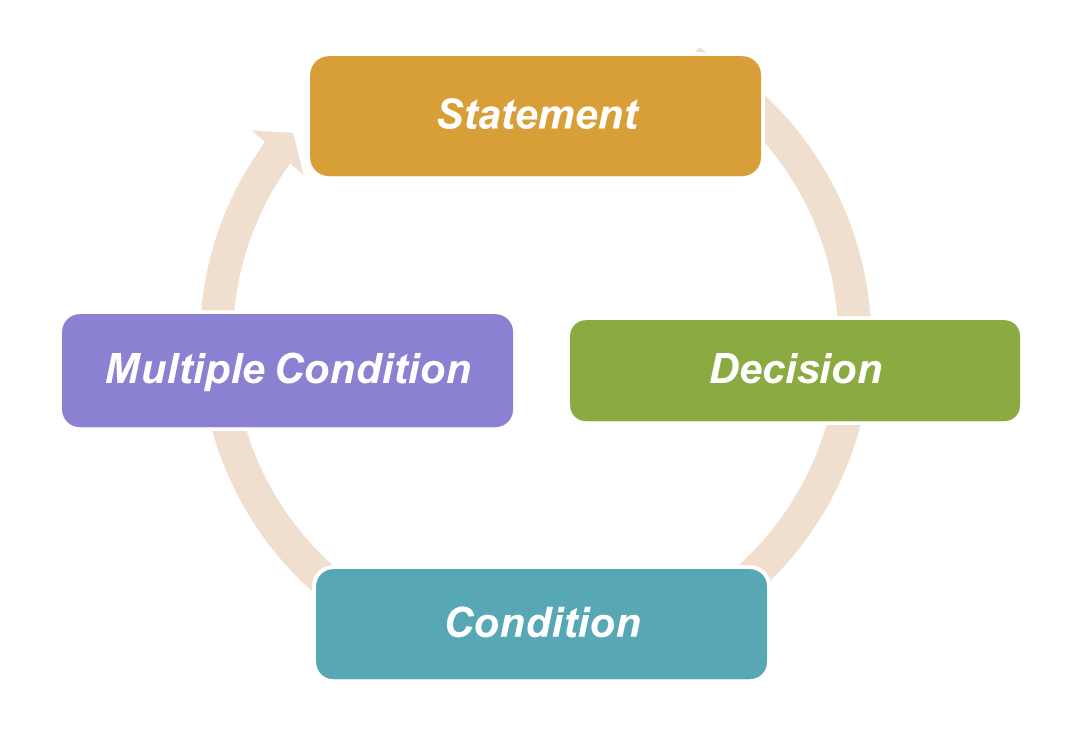
* Chỉ biết một vài phần của cấu trúc nội bộ.

# KIỂM THỬ TĨNH VÀ KIỂM THỬ ĐỘNG



## Kiểm thử động

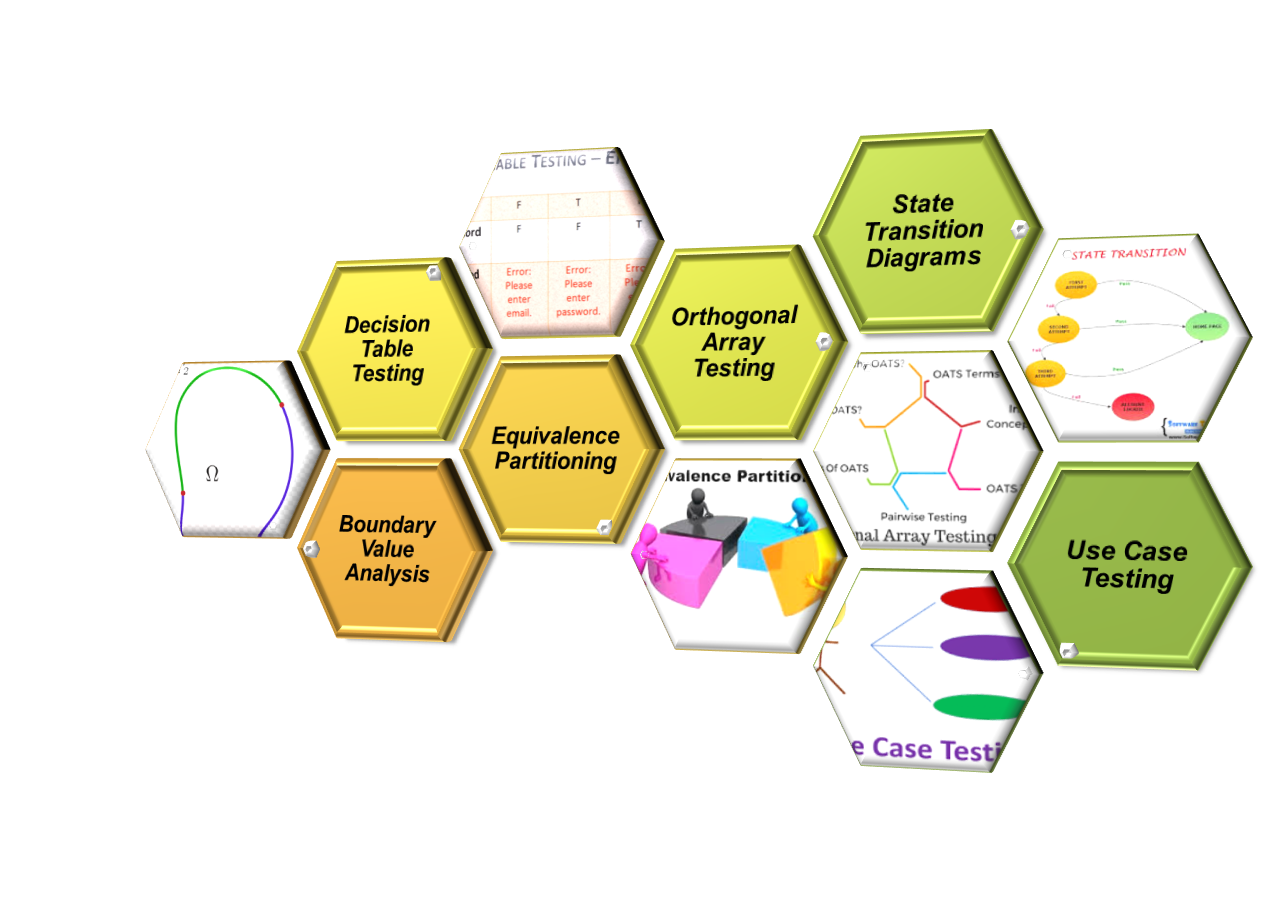
* Kiểm thử động là kỹ thuật kiểm thử được thực hiện bằng cách sử dụng một tập các giá trị đầu vào và đầu ra của nó nhằm xem xét và so sánh với các giá trị mong đợi.
* Kiểm thử tự động được áp dụng như là kỹ thuật xác định lỗi và xác định các trạng thái chất lượng của đoạn mã.
* Nó được thực hiện trong suốt quá trình đánh giá sản phẩm cuối cùng.
* Phân loại:
  + Kiểm thử dựa trên đặc tả
  + Kiểm thử dựa trên kinh nghiệm
  + Kiểm thử dựa trên cấu trúc
* Ưu điểm:
  + Tìm thấy các lỗi mà kiểm tra tĩnh không tìm thấy 🡪 kiểm tra mức cao.
  + Cải thiện chất lượng bằng cách tìm ra nhiều lỗi của ứng dụng, đảm bảo phần mềm không bị lỗi trong một số ngữ cảnh.
* Nhược điểm:
  + Tốn nhiều thời gian vì phải thực hiện nhiều test cases.
  + Tăng chi phí cho sản phẩm.
  + Đòi hiểu nhiều chi phí nhân lực để hoàn thành tác vụ.
* Kiểm thử dựa trên cấu trúc:



* Kiểm thử dựa trên kinh nghiệm:

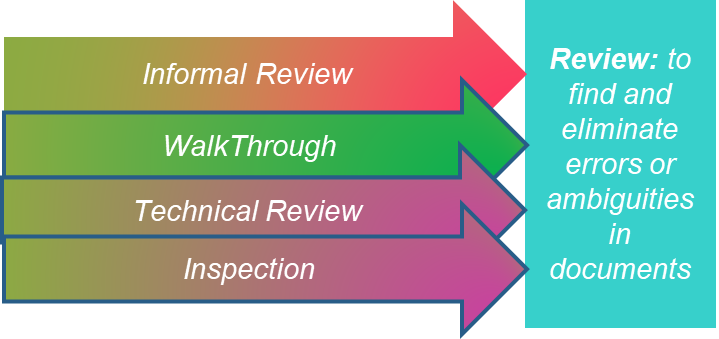


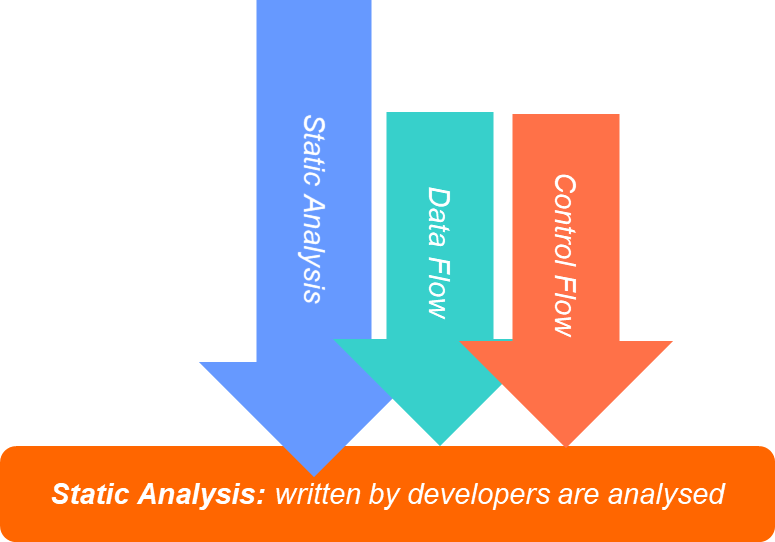
* Kiểm thử đặc tả:



## Kiểm thử tĩnh

* Kiểm thử mà không thực thi đoạn mã.
* Đánh giá: để tìm và loại bỏ lỗi hoặc sự mơ hồ trong tài liệu.





* Ưu điểm:
  + Vì thử nghiệm tĩnh có thể bắt đầu sớm trong vòng đời, phản hồi sớm về các vấn đề chất lượng có thể được thiết lập;
  + Bằng cách phát hiện các khiếm khuyết ở giai đoạn đầu, chi phí làm lại thường tương đối thấp;
  + Vì nỗ lực làm lại giảm đáng kể, số liệu năng suất phát triển có khả năng tăng lên;
  + Việc đánh giá bởi một nhóm có lợi thế bổ sung là có sự trao đổi thông tin giữa những người tham gia.
  + Các xét nghiệm tĩnh góp phần nâng cao nhận thức về các vấn đề chất lượng.
* Thường được sử dụng để tìm và loại bỏ lỗi hoặc sự mơ hồ trong các tài liệu như yêu cầu, thiết kế, trường hợp thử nghiệm, v.v.

