Tổng quan về lý thuyết kiểm thử

Bởi:

Trần Thị Huệ

TỔNG QUAN LÝ THUYẾT KIỂM THỬ PHẦN MỀM

Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là việc kiểm tra kết quả thực hiện của chương trình máy tính xem có đúng với mục tiêu đã đặt ra với nó không thông qua việc thực hiện ở một số mẫu thử.

Kiểm thử phần mềm là việc tìm ra lỗi trong bản thân phần mềm, việc kiểm thử này trong phần mềm sẽ biểu thị ra những thiếu sót mà ta có thể nhận thấy trong hành vi của phần mềm, và tìm ra những phần mềm không tuân theo quy định, đi lệch ra khỏi những yêu cầu của phần mềm.

Theo một số nhà nghiên cứu thì kiểm thử phần mềm được định nghĩa như sau:

- Theo ông Dijkstra: Kiểm thử sẽ hiện thị lỗi hiện có nhưng không hiện thị lỗi chưa thấy.
- Theo ông Beizer:
- Định luật 1: Mọi phương pháp bạn sử dụng để ngăn ngừa hoặc tìm thấy lỗi bỏ đi một phần lỗi rắc rối, cái mà những phương thức cần
- Định luật 2: Phần mềm phức tạp lớn hơn những giới hạn khả năng quản lí.
- Theo hiệp hội IEEE: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.
- Theo ông Myers: Kiểm thử là tiến trình thực thi chương trình với mục đích tìm thấy lỗi.

Vai trò của kiểm thử phần mềm

Kiểm thử để tìm ra lỗi, ghi nhận thông tin về lỗi nhưng không sửa lỗi.

Kiểm thử phần mềm không chỉ cần tìm lỗi phần mềm, mà còn là quá trình kiểm tra và xác minh một phần mềm đã đáp ứng được yêu cầu và mong đợi của khách hàng.

Một số lỗi phần mềm đã gây ra thiệt hại nghiêm trọng trong lịch sử như:

- Máy bay Airbus A300 do lỗi Phần mềm và bị tai nạn ngày 26/4/1994 giết chết 264 người
- Năm 1985, Máy xạ trị Therac-25 của Canada do lỗi phần mềm mà phát ra tia gây chết người đã giết chết 3 người và làm 3 người khác bị thương nặng.
- Vào tháng Tư năm 1999, một lỗi phần mềm gây ra sự thất bại của một vụ phóng vệ tinh quân sự gây thiệt hại 1,2 tỷ USD, vụ tai nạn đắt đỏ nhất trong lịch sử.
- 5/1996, Một lỗi phần mềm gây ra các tài khoản ngân hàng của 823 khách hàng của một ngân hàng lớn của Hoa Kỳ được ghi với 920 triệu đô la Mỹ.

Chính vì vậy, việc kiểm thử phần mềm là vô cùng quan trọng vì lỗi phần mềm nếu để lọt thì ko chỉ thiệt hại về kinh tế mà còn thiệt hại đến tính mạng con người.

Mục tiêu của kiểm thử phần mềm

Việc thực hiện kiểm thử nhằm mục tiêu:

- Bằng việc kiểm thử sẽ tìm ra lỗi trong phần mềm (Myers, 1979) và thiết lập chất lượng của phần mềm (Helzel, 1988).
- Việc kiểm thử thành công khi bạn tìm được ít nhất một lỗi và đưa ra sự đánh giá với độ tin cậy lớn.
- Đảm bảo Phần mềm đáp ứng đúng yêu cầu đề ra;
- KTPM giúp chúng ta biết rằng phần mềm đang được thử nghiệm thành công;
- KTPM giúp xác nhận được rằng Phần mềm đủ điều kiện đến tay người sử dụng;
- KTPM để đảm bảo chất lượng phần mềm;
- KTPM cung cấp và duy trì một sản phẩm chất lượng cho khách hàng.

Nguyên tắc kiểm thử phần mềm

Để kiểm thử đạt hiệu quả thì khi tiến hành kiểm thử phần mềm cần phải tuân thủ một số nguyên tắc sau:

Nguyên tắc 1: Kiểm thử chỉ ra sự hiện diện của lỗi.

Kiểm thử có thể chỉ ra có mặt của lỗi, nhưng không thể chứng minh rằng phần mềm không có lỗi. Việc kiểm thử làm giảm xác suất các khuyết tật chưa được tìm thấy còn lại trong phần mềm, nhưng ngay cả khi không có lỗi được tìm thấy, đó cũng không phải là một bằng chứng đúng đắn để khẳng định rằng phần mềm không có lỗi.

Nguyên tắc 2: Kiểm thử toàn bộ, đầy đủ là không thể.

Kiểm thử toàn bộ (kết hợp tất cả các yếu tố đầu vào và điều kiện tiên quyết) là không khả thi trừ những trường hợp nhỏ và đơn giản. Thay vì kiểm thử đầy đủ, nên sử dụng những đánh giá rủi ro và nỗ lực để ưu tiên tập trung kiểm thử.

Nguyên tắc 3: Cần bắt đầu giai đoạn kiểm thử càng sớm càng tốt.

Hoạt động kiểm thử nên bắt đầu càng sớm càng tốt trong chu trình phát triển mềm và cần được tập trung vào các mục tiêu xác định.

Nguyên tắc 4: Phân nhóm lỗi để xác định một số module tập trung lỗi nhiều nhất.

Nguyên tắc 5: Pesticide paradox

Nguyên tắc kiểm thử cũng giống như nguyên tắc sử dụng thuốc trừ sâu, khi chúng ta sử dụng một loại thuốc trừ sâu mãi thì sẽ bị nhờn thuốc nên phải thay đổi loại thuốc khác. Trong kiểm thử phần mềm, khi dùng đi dùng lại một bộ kịch bản kiểm thử thì sẽ đến lúc không thể tìm ra lỗi mới nữa. Chính vì vậy các bộ kịch bản kiểm thử phải được thường xuyên xem xét và cập nhật, phù hợp với từng thành phần khác nhau của phần mềm, mang lại khả năng tìm thấy lỗi lớn nhất.

Nguyên tắc 6: Kiểm thử được thực hiện khác nhau trong những bối cảnh khác nhau.

Kiểm thử phụ thuộc vào tình huống/trường hợp như Win app hay Web app.

Nguyên tắc 7: Suy nghĩ "Không có lỗi" là một sai lầm.

Việc tìm và sửa lỗi là không có ý nghĩa nếu phần mềm không đáp ứng yêu cầu và yêu cầu của người sử dụng.

Qui trình kiểm thử phần mềm trong suốt Vòng đời phát triển phần mềm

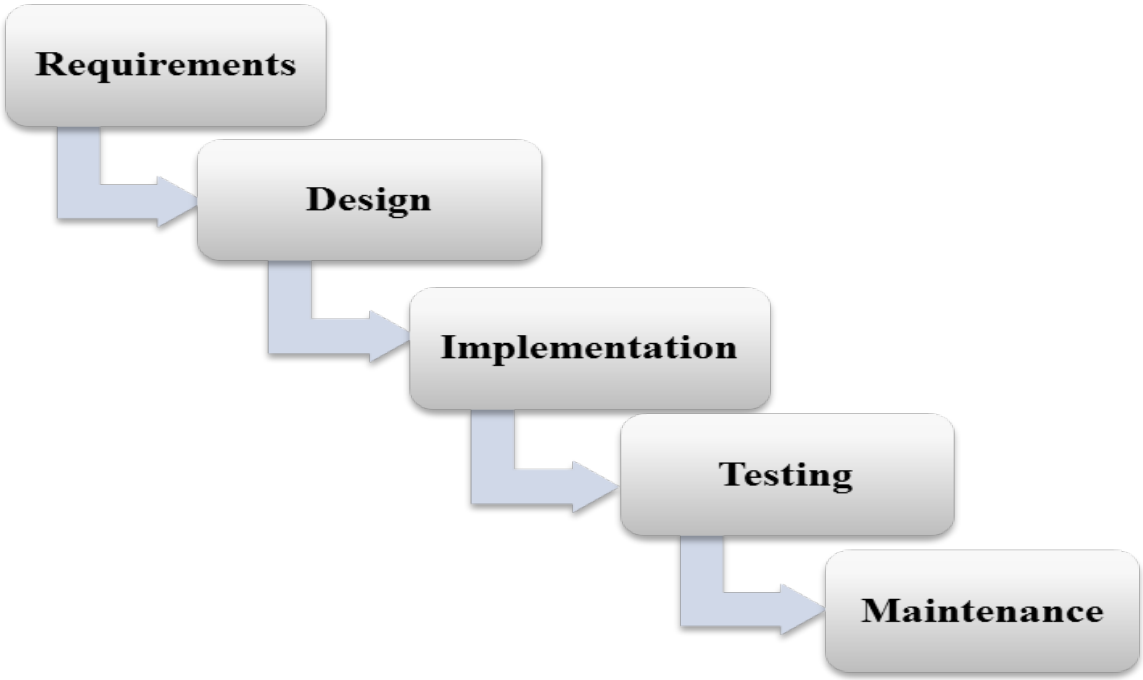
Kiểm thử phần mềm không phải là một hoạt động độc lập. Nó có vị trí của nó trong một mô hình vòng đời phát triển phần mềm, do đó mô hình vòng đời được áp dụng cho một dự án sẽ có tác động lớn đến việc thử nghiệm được thực hiện như thế nào.

Mô hình phát triển áp dụng cho một dự án sẽ phụ thuộc vào mục đích và mục tiêu dự án. Có rất nhiều mô hình đã được phát triển để đáp ứng các mục tiêu khác nhau. Các mô hình xác định các giai đoạn khác nhau của qui trình và thứ tự mà chúng được thực hiện.Các hoạt động kiểm thử liên quan rất lớn đến các hoạt động phát triển phần mềm, nó sẽ xác định những gì, ở đâu, và khi nào thực hiện theo kế hoạch kiểm thử, nó ảnh hưởng tới quá trình kiểm thử hồi quy và xác định những kỹ thuật kiểm thử sẽ được sử

dụng. Việc kiểm thử phải được tổ chức phù hợp với vòng đời phát triển nếu không nó sẽ không mang lại lợi ích

Mô hình thác nước (Waterfall Model)

Mô hình thác nước (waterfall model) là một mô hình của quy trình phát triển phần mềm, trong đó quy trình phát triển trông giống như một dòng chảy, với các pha được thực hiện theo trật tự nghiêm ngặt và không có sự quay lui hay nhảy vượt pha là: phân tích yêu cầu, thiết kế, triển khai thực hiện, kiểm thử, liên kết và bảo trì.



Hình 1.. Mô hình thác nước

Theo mô hình thác nước, người phát triển phải thực hiện từng giai đoạn theo thứ tự nghiêm ngặt. Trước hết, giai đoạn "xác định yêu cầu" phải được hoàn tất, kết quả nhận được sẽ là danh sách các yêu cầu đối với phần mềm. Sau khi các yêu cầu đã hoàn toàn được xác định, sẽ chuyển sang pha thiết kế, ở pha này người ta sẽ tạo ra các tài liệu dành cho lập trình viên, trong đó mô tả chi tiết các phương pháp và kế hoạch thực hiện các yêu cầu đã được làm rõ ở pha trước. Sau khi pha thiết kế hoàn tất, lập trình viên sẽ triển khai thực hiện (mã hóa, viết mã) đồ án họ nhận được. Giai đoạn tiếp theo là liên kết các thành phần riêng lẻ đã được những đội lập trình viên khác nhau thực hiện thành một sản phẩm hoàn chỉnh. Sau khi pha triển khai và pha liên kết hoàn tất, sẽ diễn ra pha kiểm thử và chỉnh sửa sản phẩm; ở giai đoạn này những khiếm khuyết ở các giai đoạn trước đó sẽ bị loại bỏ. Sau đó, sản phẩm phần mềm sẽ được đưa vào sử dụng; phần bảo trì phần mềm cũng sẽ được bảo đảm bằng cách bổ sung chức năng mới và loại trừ các lỗi. Vì vậy, việc chuyển từ pha phát triển này sang pha khác sẽ diễn ra chỉ sau khi các pha

trước đó đã kết thúc hoàn toàn thành công, và không thể quay lui về pha trước đó hay nhảy vượt pha.

Ưu điểm của mô hình thác nước:

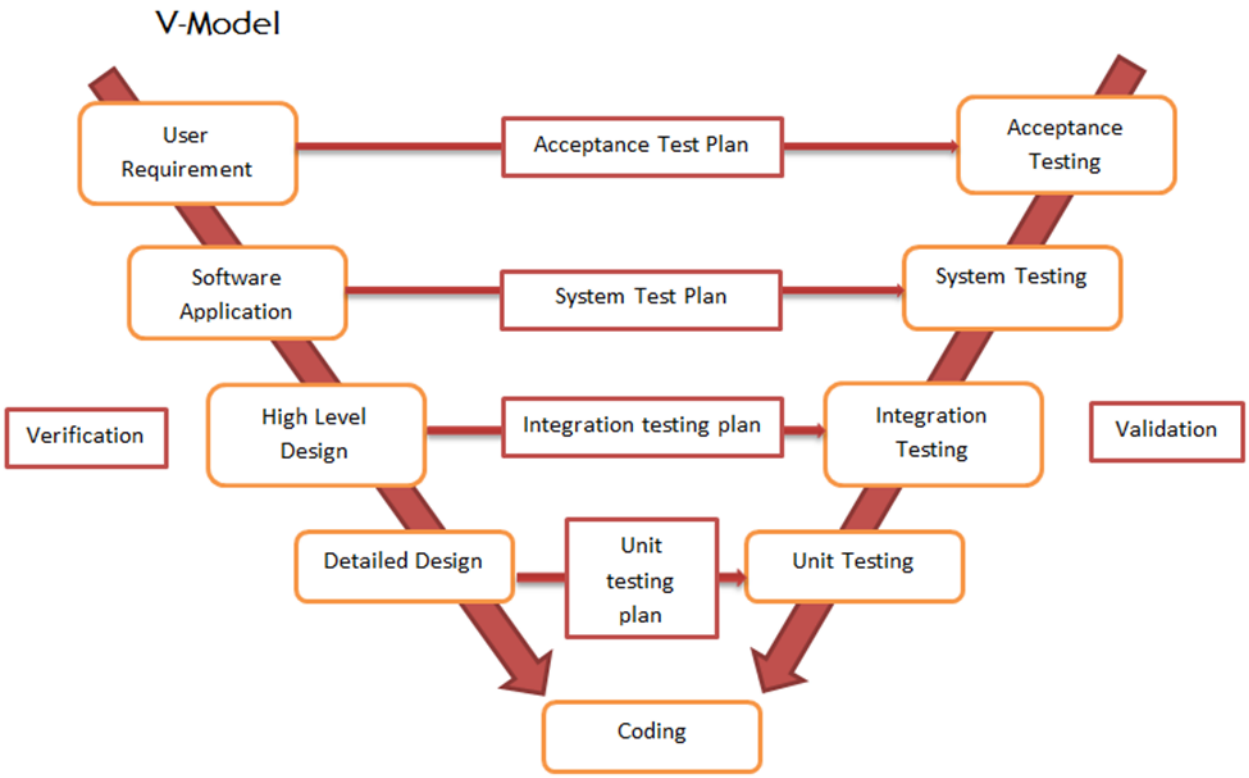
- Các giai đoạn được định nghĩa, với đầu vào và đầu ra rõ ràng nên dễ phân công công việc, phân bổ chi phí, giám sát công việc.
- Quá trình phát triển đơn giản nên phù hợp với những dự án có ít thay đổi.
- Giảm thiểu các lỗi mắc phải trong giai đoạn thiết kế.

Tuy nhiên mô hình này vẫn có một số nhược điểm:

- Mối quan hệ giữa các giai đoạn (pha) không được thể hiện.
- Thời gian thực hiện lâu.
- Quy trình sửa lỗi khó khăn và tốn nhiều chi phí. Bởi vì nếu hệ thống xuất hiện lỗi thì không thể biết được là do giai đoạn nào, nếu lỗi xuất hiện ở những giai đoạn đầu thì việc quay lại sửa lỗi sẽ tốn rất nhiều thời gian, chi phí.

Mô hình chữ V (V-model)

Mô hình chữ V là mô hình phát triển mở rộng của mô hình phát triển phần mềm theo mô hình thác nước. Toàn bộ quy trình được chia làm 2 nhóm: phát triển và kiểm thử. Mỗi giai đoạn phát triển sẽ tiến hành song song với một giai đoạn kiểm thử tương ứng dẫn đến các lỗi được phát hiện ngay từ đầu.



Hình 1.. Mô hình thác nước

Giai đoạn phát triển:

- Xác định yêu cầu của người dùng (User Requirement): Xác định yêu cầu cần thiết mà hệ thống đòi hỏi, đưa ra bản đặc tả.
- Phân tích hệ thống (System Analysis): Phân tích các yêu cầu mà hệ thống cần có và đưa ra giải pháp tích hợp các yêu cầu đó vào hệ thống.
- Thiết kế chi tiết (Detailed Design): Chi tiết hoá các bước thực hiện (Về nội dung và hình thức)
- Phát triển (Development): Thực hiện việc viết code.

Giai đoạn kiểm thử:

- Kiểm tra từng thành phần và tích hợp (Unit $ Intergration Testing): Kiểm tra các module của hệ thống tương ứng với pha thiết kế chi tiết
- Kiểm thử toàn hệ thống (System Testing): Kiểm thử hoạt động của hệ thống (Về chức năng, giao diện)
- Nghiệm thu (Acceptance Testing): Kiểm tra lần cuối cùng và đưa sản phẩm vào sử dụng

Ưu điểm của mô hình chữ V:

- Có thể làm 1 số việc song song. Ví dụ: Nếu làm yêu cầu đúng thì có thể làm song song với việc thiết kế test.
- Đạt được phần mềm chất lượng, các pha tương thích với nhau, hỗ trợ cho nhau.
- Các hoạt động kiểm thử được chú trọng và thực hiện song song với các hoạt động liên quan đến đặc tả yêu cầu và thiết kế. Hay nói cách khác, mô hình này khuyến khích các hoạt động liên quan đến kế hoạch kiểm thử được tiến hành sớm trong chu kỳ phát triển, không phải đợi đến lúc kết thúc giai đoạn hiện thực.

Tuy nhiên mô hình chữ V cũng có một số nhược điểm:

- Đòi hỏi tất cả yêu cầu phần mềm phải được xác định rõ ràng ngay từ đầu dự án.
- Pha sau thực chỉ được thực hiện khi pha trước kết thúc, không thể quay ngược trở lại pha trước.
- Người sử dụng không có cơ hội tham gia trong suốt thời gian của các giai đoạn trung gian từ thiết kế cho đến kiểm thử.
- Kiểm thử đơn vị (Unit Test)

Unit Test là kiểm tra từng bộ phận nhỏ, từng bộ phận riêng biệt trong source code của chương trình, kiểm tra xem các bộ phận đó hoạt động có chính xác không. Một Unit Test là một phần của source code, thực thi một phần coode khác và so sánh kết quả thực tế so với kết quả mong đợi.

Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ PTPM. Mục đích của Unit Test là đảm bảo thông tin được xử lý là chính xác, phát hiện lỗi sớm và kịp thời chỉnh sửa, tiết kiệm thời gian và chi phí cho việc kiểm tra và sửa lỗi ở các mức kiểm tra sau đó.

- ***Kiểm thử tích hợp (Integration Test)***

Integration Test kết hợp các thành phần của một ứng dụng và kiểm tra như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Integration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.

Integration có 2 mục tiêu chính:

- Phát hiện lỗi giao tiếp xảy ra giữa các Unit;
- Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (Subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (System) chuẩn bị cho kiểm tra ở mức hệ thống (System Test).

Trừ một số ngoại lệ, Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test và tất cả các lỗi ở mức Unit đã được sửa chữa. Một

số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit test với các giao tiếp giả lập thì không cần phải thực hiện Integration test nữa. Nhưng trên thực tế, việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác nhau.

Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần dần từng Unit. Tức là ta chỉ cần kiểm tra giao tiếp giữa Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó (passed). Điều này làm cho số lượng kiểm tra sẽ giảm đi rất nhiều và sai sót sẽ giảm đi đáng kể.

Có 4 loại kiểm tra trong Integration Test:

- **Kiểm tra cấu trúc (Structure test):** Tương tự White Box Test (kiểm tra nhằm đảm bảo các thành phần bên trong của một chường trình chạy đúng), chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình cẳng hạn các lệnh và nhánh bên trong.
- **Kiểm tra chức năng (Funtional test):** Tương tự Black Box Test (kiểm tra chỉ chú trọng đến chức năng của chương trình, không quan tâm tới cấu trúc bên trong), chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
- **Kiểm tra hiệu năng (Performance test):** Kiểm tra việc vận hành của hệ thống.
- **Kiểm tra khả năng chịu tải (Stress test or Load test):** Kiểm tra các giới hạn của hệ thống.

- ***Kiểm thử hệ thống (System Test)***

Mục đích của System Test là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. System Test bắt đầu khi tất các bộ phần của phần mềm đã được tích hợp thành công.

Điểm khác nhau then chốt giữa Integration Test và System Test là System test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc với nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để đảm bảo mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Có nhiều loại kiểm tra trong System Test, phổ biến:

- **Kiểm tra chức năng (Funtional test):** Đảm bảo các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
- **Kiểm tra khả năng chịu tải (Stress test or Load test**): Đảm bảo hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người cùng truy xuất 1 lúc). Stress test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường…

- **Khả năng bảo mật (Security Test):** Đảm bảo tính toàn vẹn, tính bảo mật dữ liệu của hệ thống.

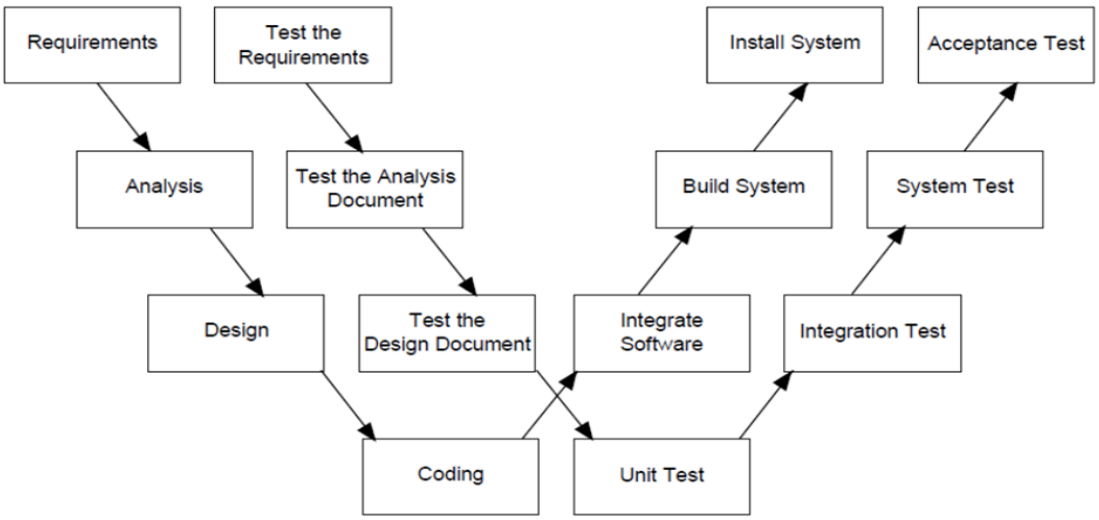
- ***Kiểm thử chấp nhận (Acceptence Test)***

Thông thường sau giai đoạn System Test là Acceptance Test, được khách hàng thực hiện (hoặc ủy quyền cho 1 nhóm người thứ 3 thực hiện). Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả các yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.

Có 2 loại kiểm tra là Alpa Test và Beta Test:

- **Với Alpha Test:** Người sử dụng (tiềm năng) kiểm tra phần mềm ngay tại nơi PTPM, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi và lên kế hoạch sửa chữa.
- **Với Beta Test:** Phần mềm sẽ được gửi tới người dùng (tiềm năng) để kiểm tra ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ được gửi ngược lại cho lập trình viên để sửa chữa.

Mô hình chữ W (W-model)



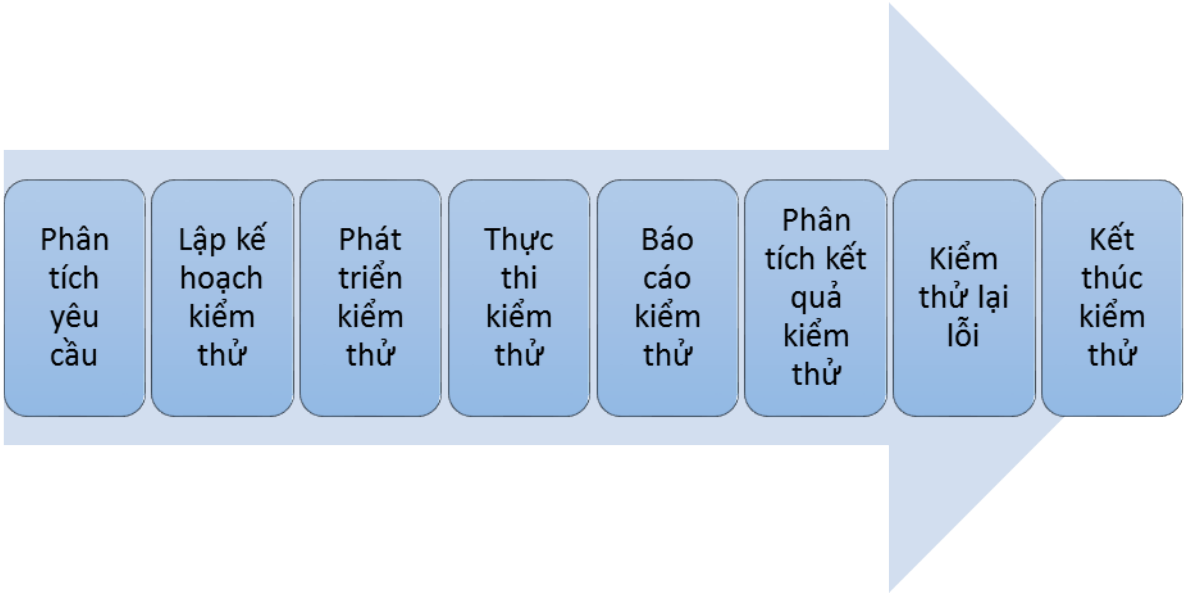
Hình 1.. Mô hình chữ W

Mô hình chữ W (W-model) là mô hình được phát triển dựa trên mô hình chữ V.

Nếu như trong mô hình V chỉ tập trung vào kiểm thử động (dynamic testing), không chú trọng đến lợi ích và hiệu quả của kiểm thử tĩnh (static testing) như: đánh giá, kiểm tra và phân tích code thì mô hình W nhằm giải quyết những thiếu sót trong mô hình V.

Trong mô hình W ta thấy được kỹ thuật kiểm thử tĩnh được áp dụng trong đoạn đầu của sự phát triển. Tất cả các hoạt động của quá trình phát triển phần mềm ánh xạ đến một hoạt động kiểm thử. Các mô hình W đặt quá trình trình kiểm thử bình đẳng với quá trình phát triển phần mềm.

Quy trình kiểm thử phần mềm (Test Process)



Hình 1.. Quy trình kiểm thử phần mềm

Mô tả các bước trong quy trình kiểm thử phần mềm:

1. Phân tích yêu cầu: Xác định phạm vi test.
2. Lập kết hoạch kiểm thử: Chiến lược kiểm thử (test strategy), test plan.
3. Phát triển kiểm thử: Viết test procedure, test scenario, test case, test data và test script.
4. Thực thi kiểm thử: Tester thực thi phần mềm dựa trên test plan và test case.
5. Báo cáo kiểm thử: Tester điền kết quả test vào test case và tạo báo cáo kết quả test.
6. Phân tích kết quả kiểm thử: Hoặc còn gọi là phân tích lỗi để quyết định lỗi nào sẽ được sửa và lỗi nào sẽ không sửa.
7. Kiểm thử lại lỗi: Sau khi một lỗi (defect) được DEV sửa xong, chuyển phần mềm cho tester test lại.
8. Kết thúc kiểm thử: Khi test đã đáp ứng được điều kiện dừng. Từ đó rút ra các bài học kinh nghiệm.

Nghề kiểm thử phần mềm – Tester

Kiểm thử viên – Tester là người thực hiện kiểm thử. Công việc của các kiểm thử viên là tìm kiếm các lỗi hay khiếm khuyết của hệ thống phần mềm, hoặc thẩm định nhằm đảm bảo hiệu quả hoạt động tối ưu để giảm phí tổn do lỗi của phần mềm gây ra cho khách hàng.

Kiểm thử viên có các vai trò và trách nhiệm chính sau đây: phát triển Test Case và các thủ tục Kiểm thử, tạo bộ dữ liệu cho việc kiểm thử, thực hiện kiểm thử, sử dụng công cụ để thực hiện kiểm thử tự động, chuẩn bị các văn bản cho kiểm thử, theo dõi lỗi và báo cáo kết quả kiểm thử.

Những tố chất để làm tốt công việc của một kiểm thử viên bao gồm kỹ năng kỹ thuật và kỹ năng về hành vi ứng xử. Một kiểm thử viên cần có kiến thức về phát triển phần mềm, quy trình kiểm thử phần mềm và các văn bản liên quan đến quy trình kiểm thử. Ngoài ra một kiểm thử viên cần có tư duy logic tốt, có tinh thần trách nhiệm cao, có các phương pháp tổ chức và cẩn thận, tỉ mỉ, ham học hỏi …

Khi nào nên dừng kiểm thử

Rất khó để xác định khi nào thì dừng kiểm thử, kiểm thử là một quá trình có thể không bao giờ kết thúc. Vì vậy cần phải có 1 vài yếu tố để dựa vào đó và xác định việc dừng kiểm thử:

- Đáp ứng yêu cầu của khách hàng;
- Hoàn thành toàn bộ kịch bản kiểm thử và đóng hết lỗi;
- Hoàn thành kiểm thử các chức năng và đạt được mức độ, tiêu chí hoàn hành kiểm thử;
- Tỉ lệ lỗi giảm và không còn các lỗi có độ ưu tiên cao;
- Hết ngân sách dành cho kiểm thử;
- Quyết định của quản lý.