# **CHƯƠNG 1: TỔNG QUAN VỀ PHẦN MỀM VÀ KIỂM THỬ PHẦN MỀM.**

## **Phần mềm là gì?**

### **Khái niệm**

* Phần mềm là những ứng dụng chạy bên trong máy tính, (nhằm cung cấp các chức năng, công việc của người sử dụng thông qua phần cứng máy tính.
* Một phần mềm thường gồm 3 phần:
* Chương trình máy tính: mã nguồn, mã máy.
* Cấu trúc dữ liệu: cấu trúc làm việc (bộ nhớ trong), cấu trúc lưu trữ (bộ nhớ ngoài).
* Các tài liệu liên quan: tài liệu hướng dẫn sử dụng, tài liệu phát triển, tài liệu tham khảo kĩ thuật,…

### **Phân loại phần mềm**

* **Phần mềm hệ thống**: Dùng để quản lý hành vi phần cứng máy tính, cung cấp các chức năng cơ bản được người dùng yêu cầu hoặc phần mềm khác chạy đúng (nếu có). Được thiết kế để cung cấp một nền tảng để chạy phần mềm ứng dụng như hệ điều hành Windows, iOS, Android, trình điều khiển thiết bị Driver…
* **Phần mềm ứng dụng**:
* Sử dụng hệ thống máy tính để thực hiện các chức năng đặc biệt hoặc cung cấp các chức năng giải trí ngoài hoạt động cơ bản của chính máy tính.
* Phổ biến trong các phần mềm văn phòng như Microsoft Office, phần mềm game, các công cụ và tiện ích khác…
* **Phần mềm dịch mã (trình biên dịch và trình thông dịch)**: Dịch các câu lệnh từ mã nguồn của ngôn ngữ lập trình sang dạng ngôn ngữ máy sao cho thiết bị thực thi có thể hiểu được và hiện thực.
* **Nền tảng ứng dụng (ASP.net, PHP…)**: Dựa vào nền tảng ứng dụng Web của Microsoft tạo ra các ứng dụng Web, dịch vụ Web (Web Service)

## **Quy trình phát triển phần mềm**

Cũng như các ngành sản xuất khác, quy trình là một trong những yếu tố đầu tiên và cực kỳ quan trọng đem lại thành công cho các nhà phát triển phần mềm, nó giúp cho mọi thành viên trong dự án từ người cũ đến người mới, trong hay ngoài công ty đều có thể xử lý đồng bộ công việc tương ứng với trị trí của mình thông qua cách thức chung của công ty. Quy trình phát triển phần mềm có tính chất quyết định để tạo ra 1 sản phẩm có chi phí thấp và năng suất cao.

Quy trình phát triển phần mềm là một cấu trúc bao gồm tập hợp các thao tác và các kết quả tương quan sử dụng trong việc phát triển để sản xuất ra một sản phẩm phần mềm. Một quy trình phát triển phần mềm bao gồm các giai đoạn như sau:

* **Giải pháp, yêu cầu**: Thực hiện khảo sát chi tiết yêu cầu của khách hàng để từ đó tổng hợp vào tài liệu giải pháp. Tài liệu này phải mô tả đầy đủ các yêu cầu về chức năng, phi chức năng và giao diện. Kết quả đầu ra là Tài liệu đặc tả yêu cầu.
* **Thiết kế**: Thực hiện thiết kế và tổng hợp vào tài liệu thiết kế. Kết quả là tài liệu thiết kế tổng thể, thiết kế module, thiết kế CSDL.
* **Lập trình**: Lập trình viên thực hiện lập trình dựa trên tài liệu Giải pháp và Thiết kế đã được phê duyệt. Kết quả đầu ra là Source code.
* **Kiểm thử**: Tester tạo kịch bản kiểm thử (testcase) theo tài liệu đặc tả yêu cầu, thực hiện kiểm thử và cập nhật kết quả vào kịch bản kiểm thử, log lỗi trên các tool quản lý lỗi. Kết quả đầu ra là Testcase, lỗi trên hệ thống quản lý lỗi.
* **Triển khai**: Triển khai sản phẩm cho khách hàng. Kết quả đầu ra là biên bản triển khai với khách hàng.

## **Lỗi phần mềm**

### **Khái niệm lỗi phần mềm**

Một lỗi phần mềm là một lỗi, lỗ hổng, thất bại, hoặc có lỗi trong một chương trình máy tính hoặc hệ thống đó là nguyên nhân nó tạo ra kết quả không chính xác hoặc không mong muốn, hoặc vận hành theo cách không được định hướng trước.

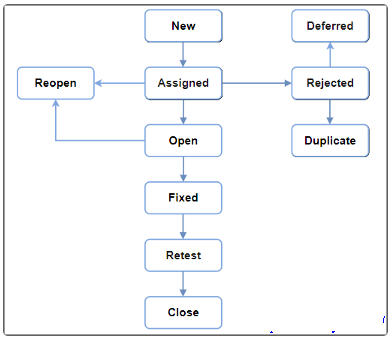
### **Quy định xác định lỗi phần mềm**

Một lỗi phần mềm xuất hiện khi vi phạm ít nhất 1 trong 5 quy tắc dưới đây:

* **Quy tắc 1**: Phần mềm không thực hiện một số thứ giống như mô tả trong bản đặc tả phần mềm.
* **Quy tắc 2**: Phần mềm thực hiện một số việc mà bản đặc tả yêu cầu nó không được thực hiện.
* **Quy tắc 3**: Phần mềm thực hiện một số chức năng mà bản đặc tả không đề cập tới.
* **Quy tắc 4**: Phần mềm không thực hiện một số việc mà bản đặc tả không đề cập tới, nhưng là những việc nên làm.
* **Quy tắc 5**: Đối với người kiểm thử, phần mềm là khó hiểu, khó sử dụng, chậm đối với người sử dụng.

### **Vòng đời lỗi**

Vòng đời của lỗi là một hành trình mà mà lỗi đi qua trong suốt cuộc đời của nó. Nó thay đổi từ tổ chức này sang tổ chức khác, từ dự án này đến dự án khác và nó được điều chỉnh bởi quy trình kiểm thử phần mềm.



*Hình 1. 1 Vòng đời của lỗi*

Vòng đời của lỗi bao gồm các trạng thái dưới đây:

* **New**: Khi mà lần lỗi được log lên đầu tiên bởi người kiểm thử.
* **Assigned**: Khi lỗi được đăng lên và chỉ định cho lập trình viên nào đó.
* **Open**: Khi lập trình viên đang sửa lỗi.
* **Fixed**: Khi lập trình viên hoàn thành việc sửa lỗi.
* **Retest**: Người kiểm thử kiểm tra lỗi đã được sửa hay chưa, có phát sinh thêm lỗi mới nào không.
* **Reopened**: Nếu lỗi vẫn còn, người kiểm thử sẽ trả lại cho lập trình viên. Lỗi phải đi lại một vòng đời như cũ.
* **Deferred**: Lỗi được sửa trong bản phát hành tiếp theo. Lý do có thể là độ ưu tiên của lỗi có thể là thấp, thiếu thời gian để phát hành hoặc lỗi có thể không ảnh hưởng lớn đến phần mềm.
* **Rejected**: Nếu lập trình viên cho rằng không phải lỗi, họ có thể chuyển sang trạng thái này.
* **Duplicate**: Lỗi được đăng trùng với nhau.
* **Closed**: Khi người kiểm thử đã thấy lỗi được sửa triệt để.
* **Not a bug/Enhancement**: Một số thay đổi trong ứng dụng, không phải là lỗi.

## **Kiểm thử phần mềm (Software Testing)**

### **Khái niệm kiểm thử phần mềm**

Kiểm thử phần mềm là một quy trình nhằm đảm bảo độ tin cậy và chất lượng của phần mềm. Mục đích của kiểm thử phần mềm là chỉ ra rằng phần mềm thực hiện đúng các chức năng mà khách hàng mong muốn.

**Có 2 loại kiểm thử phần mềm:**

* Kiểm thử phần mềm thủ công (Manual Testing).
* Kiểm thử tự động (Automation Testing).

**Mục tiêu của kiểm thử phần mềm:**

* Phát hiện ra càng nhiễu lỗi (bug) càng tốt trong thời gian kiểm thử xác định trước.
* Chứng minh rằng sản phẩm phần mềm phù hợp với các đặc tả yêu cầu của nó.
* Tạo ra các testcase chất lượng nhằm tìm ra lỗi (nếu có) với chi phí thấp nhất.

**Ai là người kiểm thử?**

Trong hầu hết các trường hợp, người kiểm thử (Tester) có thể là:

* Software Tester: Nhân viên kiểm thử phần mềm.
* Software Developer: Nhân viên phát triển phần mềm.
* Leader hoặc Manager của dự án.
* Product Owner: Người sở hữu sản phẩm ( Acceptance Testing).
* End – User: Người dùng cuối.

**Các vai trò trong kiểm thử phần mềm:**

* Test Manager: Là người đứng đầu bộ phận kiểm thử, quản lý chung về các vấn đề liên quan như quy trình làm việc, nhân sự,…
* Test Leader: Là người trực tiếp tham gia vào quá trình kiểm thử dự án cùng với tester. Test leader đảm nhiệm vai trò quản lý công việc của tester, thực hiện xác nhận các sản phẩm mà tester tạo ra cũng như báo cáo test manager khi có yêu cầu.
* Tester / QC: Là người trực tiếp thực hiện quá trình kiểm thử, đảm bảo chất lượng của sản phẩm theo những nhiệm vụ được phân công.

### **Các nguyên tắc cơ bản của kiểm thử phần mềm**

**Nguyên tắc 1**: **Kiểm thử luôn có lỗi**

* Kiểm thử có thể cho thấy rằng phần mềm đang có lỗi, nhưng không thể chứng minh rằng một phần mềm không có lỗi.
* Kiểm thử làm giảm xác suất lỗi tiềm ẩn trong phần mềm, ngay cả khi đã kiểm thử nghiêm ngặt phần mềm vẫn có thể còn lỗi. Vì vậy, cần phải tìm được càng nhiều lỗi càng tốt.

**Nguyên tắc 2**: **Kiểm thử toàn bộ là không thể**

* Kiểm thử với tất cả các kết hợp đầu vào và đầu ra, với tất cả các kịch bản là không thể trừ khi kiểm thử chỉ bao gồm một số ít trường hợp thì có thể kiểm thử toàn bộ.
* Thay vì kiểm thử toàn bộ, việc phân tích rủi ro và dựa trên sự mức độ ưu tiên người kiểm thử có thể tập trung việc kiểm thử vào một số điểm cần thiết, có nguy cơ lỗi cao hơn. Nghĩa là phải lên kế hoạch kiểm thử, thiết kế trường hợp kiểm thử sao cho có độ bao phủ nhiều nhất và giảm thiểu rủi ro sót lỗi khi đến tay người dùng.

**Nguyên tắc 3: Kiểm thử càng sớm càng tốt**

* Để tìm được lỗi sớm, các hoạt động kiểm thử nên được bắt đầu càng sớm càng tốt trong quy trình phát triển (vòng đời phát triển) phần mềm hoặc hệ thống, và nên tập trung vào các hoạt động/mục tiêu đã xác định trước.
* Các hoạt động kiểm thử được bắt đầu càng sớm thì sẽ phát hiện ra lỗi sớm khi đó ít tốn công để tìm lỗi và sửa chữa.

**Nguyên tắc 4: Sự tập trung của lỗi**

* Thông thường, lỗi tập trung vào những module, thành phần chức năng chính của hệ thống. Nếu xác định được điều này bạn sẽ tập trung vào tìm kiếm lỗi quanh khu vực được xác định. Nó được coi là một trong những cách hiệu quả nhất để thực hiện kiểm tra hiệu quả.

**Nguyên tắc 5: Nghịch lý thuốc trừ sâu**

* Nếu bạn sử dụng cùng một tập hợp các trường hợp kiểm thử liên tục, sau đó một thời gian các trường hợp kiểm thử không tìm thấy lỗi nào mới. Hiệu quả của các trường hợp kiểm thử bắt đầu giảm xuống sau một số lần thực hiện, vì vậy luôn luôn chúng ta phải luôn xem xét và sửa đổi các trường hợp kiểm thử trên một khoảng thời gian thường xuyên.

**Nguyên tắc 6: Kiểm thử phụ thuộc vào ngữ cảnh**

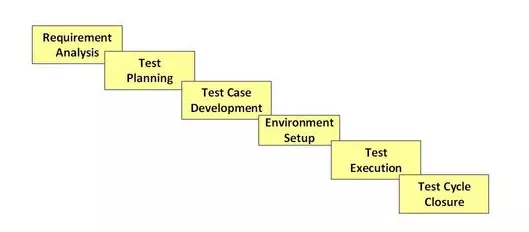
* Theo nguyên tắc này thì việc kiểm thử phụ thuộc vào ngữ cảnh và chúng ta phải tiếp cận kiểm thử theo nhiều ngữ cảnh khác nhau Nếu bạn đang kiểm thử ứng dụng web và ứng dụng di động bằng cách sử dụng chiến lược kiểm thử giống nhau, thì đó là sai. Chiến lược để kiểm thử ứng dụng web sẽ khác với kiểm thử ứng dụng cho thiết bị di động của Android.

**Nguyên tắc 7: Không có lỗi – sai lầm**

* Việc không tìm thấy lỗi trên sản phẩm không đồng nghĩa với việc sản phẩm đã sẵn sàng để tung ra thị trường. Việc không tìm thấy lỗi cũng có thể là do bộ trường hợp kiểm thử được tạo ra chỉ nhằm kiểm tra những tính năng được làm đúng theo yêu cầu thay vì nhằm tìm kiếm lỗi mới.

### **Quy trình kiểm thử phần mềm (STLC)**

Software Testing Life Cycle đề cập đến một quy trình Test (Testing process) trong đó các bước cụ thể được thực hiện theo một trình tự nhất định để đảm bảo mục tiêu chất lượng được đáp ứng. Trong quy trình kiểm thử phần mềm mỗi hoạt động được thực hiện một cách có kế hoạch và hệ thống. Mỗi một giai đoạn có các mục tiêu khác nhau.



*Hình 1. 2 Quy trình kiểm thử phần mềm*

1. **Requirement Analysis (Phân tích yêu cầu)**

* Đầu vào:

Đầu vào của giai đoạn phân tích yêu cầu bao gồm các tài liệu như: tài liệu đặc tả yêu cầu, tài liệu thiết kế hệ thống, tài liệu khách hàng yêu cầu về các tiêu chí chấp nhận của sản phẩm,…

* Hoạt động:

Trong giai đoạn này, đội test sẽ nghiên cứu tài liệu yêu cầu để đưa ra cái nhìn tổng quan về phần mềm. Từ đó có thể xác định được yêu cầu cần được kiểm tra như: yêu cầu về chức năng (xác định xem phần mềm cần phải làm gì?) hoặc là các yêu cầu phi chức năng (hiệu năng, bảo mật,…). Đội tester / QA có thể tương tác với các bên liên quan như: Khách hàng, BA (Business Analyst), Leader,… để hiểu chính xác hơn về yêu cầu của sản phẩm. Những câu hỏi này sẽ được lưu trữ vào file Q&A (Question and Answer). Các câu hỏi nên được đưa ra dưới dạng Yes/No question hoặc các lựa chọn để tiết kiệm thời gian trả lời cũng như hỗ trợ đưa ra những gợi ý hay để xây dựng sản phẩm ngay từ đầu.

* Đầu ra:

Đầu ra của giai đoạn phân tích yêu cầu bao gồm tài liệu chứa các câu hỏi và câu trả lời liên quan đến nghiệp vụ của hệ thống, tài liệu báo cáo tính khả thi, phân tích rủi ro của việc kiểm thử phần mềm.

1. **Test Planning (giai đoạn lập kế hoạch kiểm thử)**

* Đầu vào:

Đầu vào của giai đoạn lập kế hoạch kiểm thử là các tài liệu đặc tả đã được cập nhật thông qua các câu hỏi và câu trả lời được đưa ra trong giai đoạn phân tích yêu cầu, tài liệu báo cáo tính khả thi, phân tích rủi ro của việc kiểm thử phần mềm.

* Hoạt động:

Đây là giai đoạn tiếp cận dự án được thực hiện bởi Test Manager / Test Leader dựa vào các đặc tả đã phân tích, phối hợp cùng project manager lập kế hoạch test cho dự án:

* Xác định phạm vi dự án: Dự án được thực hiện trong thời gian bao lâu? Bao gồm những công việc gì cho từng khoảng thời gian xác định? Từ đó đưa ra lịch trình thực hiện cho từng công việc nhỏ sao cho phù hợp với toàn bộ đội dự án.
* Xác định phương pháp tiếp cận.
* Xác định các nguồn lực:

Con người: Có bao nhiêu người tham gia dự án, ai sẽ là người test phần nào, bao nhiêu tester tham gia?

Thiết bị: Số lượng server, máy tính,…để thực hiện test là bao nhiêu.

* Lên kế hoạch thiết kế công việc test: Bản kế hoạch kiểm thử sẽ bao gồm các nội dung:

- Liệt kê các chức năng cần kiểm thử: Để thực hiện test chức năng này thì cần làm những công việc gì, trong thời gian bao lâu, cái nào thực hiện trước, cái nào thực hiện sau, ai là người thực hiện.

- Xác định điều kiện bắt đầu: Xác định những điều kiện tối thiểu để bắt đầu hoạt động kiểm thử cho từng chức năng.

- Xác định điều kiện kết thúc: Khi có những điều kiện nào thì sẽ kết thúc việc kiểm thử.

* Đầu ra:

Đầu ra của giai đoạn lập kế hoạch bao gồm các tài liệu như test plan, test estimation, test schedule.

1. **Test Case Development (giai đoạn thiết kế kịch bản kiểm thử)**

* Đầu vào:

Đầu vào của giai đoạn thiết kế kịch bản kiểm thử là test plan, test estimation, test schedule, các tài liệu đặc tả đã được cập nhật.

* Hoạt động:
* Review tài liệu: Các kiểm thử viên cần review lại tất cả các tài liệu để xác định công việc cần làm, các công việc có khác gì so với dự án trước khách hàng đưa cho, chức năng nào cần test, chức năng nào không cần test lại nữa. Từ đó, vừa có thể tiết kiệm thời gian mà vẫn đưa ra được một kịch bản kiểm thử đầy đủ và hiệu quả.
* Viết test case/ checklist: Tester bắt tay vào việc viết test case chi tiết dựa vào kế hoạch đã đưa ra và vận dụng các kỹ thuật thiết kế kịch bản kiểm thử. Test case cần bao phủ được tất cả các trường hợp kiểm thử có thể xảy ra cũng như đáp ứng đầy đủ các tiêu chí của sản phẩm. Đồng thời, tester cũng cần đánh giá mức độ ưu tiên cho từng test case.
* Chuẩn bị dữ liệu kiểm thử: Cùng với việc tạo ra các test case chi tiết, đội kiểm thử cũng cần chuẩn bị trước các dữ liệu kiểm thử cho các trường hợp cần thiết như test data, test script.
* Review test case/ checklist: Sau khi hoàn thành, các thành viên trong đội kiểm thử hoặc test leader cũng cần review lại test case đã tạo để có thể bổ sung, hỗ trợ lẫn nhau nhằm tránh những sai sót trong thiết kế test case và rủi ro về sau.
* Đầu ra:

Sau khi hoàn thành thiết kế kịch bản kiểm thử, đội kiểm thử sẽ có các tài liệu bao gồm: test design, test case, check list, test data.

1. **Environment Setup (giai đoạn thiết lập môi trường kiểm thử)**

* Đầu vào:

Đầu vào của giai đoạn cài đặt môi trường kiểm thử là test data, test plan,smoke test case.

* Hoạt động:
* Việc cài đặt môi trường kiểm thử là giai đoạn cũng rất quan trọng trong vòng đời phát triển phần mềm. Môi trường kiểm thử sẽ được quyết định dựa trên những yêu cầu của khách hàng, hay đặc thù của sản phẩm ví dụ như server/ client/ network,...
* Tester cần chuẩn bị một vài test case để kiểm tra xem môi trường cài đặt đã sẵn sàng cho việc kiểm thử hay chưa. Đây chính là việc thực thi các smoke test case.
* Đầu ra:

Đầu ra của giai đoạn này là môi trường đã được cài đặt đúng theo yêu cầu, sẵn sàng cho việc kiểm thử và kết quả của smoke test case.

1. Test Execution (giai đoạn thực hiện kiểm thử)

* Đầu vào:

Tài liệu đầu vào của giai đoạn này là test plan, test design, test case, check list, test data, test automation script.

* Hoạt động:
* Thực hiện các test case như thiết kế và mức độ ưu tiên đã đưa ra trên môi trường đã được cài đặt.
* So sánh với kết quả mong đợi sau báo cáo các bug xảy ra lên tool quản lý lỗi và theo dõi trạng thái của lỗi đến khi được sửa thành công.
* Thực hiện re-test để verify các bug đã được fix và regression test khi có sự thay đổi liên quan.
* Trong quá trình thực hiện kiểm thử, kiểm thử viên cũng có thể hỗ trợ, đề xuất cho cả đội dự án để có giải pháp hợp lý và kết hợp công việc hiệu quả.
* Đo và phân tích tiến độ: kiểm thử viên cũng cần kiểm soát chặt chẽ tiến độ công việc của mình bằng cách so sánh tiến độ thực tế với kế hoạch, nếu chậm cần phải điều chỉnh sao cho kịp tiến độ dự án, nếu nhanh cũng cần điều chỉnh vì có thể test lead lên kế hoạch chưa sát với thực tế dự án. Từ đó có thể sửa chữa test plan cần điều chỉnh để phù hợp với tiến độ dự án đưa ra.
* Report thường xuyên cho khách hàng về tình hình thực hiện dự án: Cung cấp thông tin trong quá trình kiểm thử đã làm được những chức năng nào, còn chức năng nào, hoàn thành được bao nhiêu phần trăm công việc, báo cáo các trường hợp phát sinh sớm, tránh ảnh hưởng tiến độ công việc của cả ngày.
* Đầu ra:

Đầu ra của giai đoạn này là test results ( kết quả kiểm thử), defect reports( danh sách các lỗi tìm được).

1. **Test Cycle Closure (giai đoạn đóng chu trình kiểm thử)**

* Đầu vào:

Đầu vào của giai đoạn đóng chu trình kiểm thử là bao gồm tất cả những tài liệu liên quan đã được tổng hợp, ghi chép và hoàn thiện đầy đủ trong suốt quy trình kiểm thử của dự án: tài liệu phân tích đặc tả yêu cầu, test plan, test results, defect reports, tài liệu Q&A…

* Hoạt động:
* Đây là giai đoạn cuối cùng trong quy trình kiểm thử phần mềm.
* Ở giai đoạn này, QA team thực hiện tổng kết, báo cáo kết quả về việc thực thi test case, bao nhiêu case pass/ fail, bao nhiêu case đã được fix, mức độ nghiêm trọng của lỗi, bao nhiêu lỗi cao/ thấp, lỗi còn nhiều ở chức năng nào, dev nào nhiều lỗi. Chức năng nào đã hoàn thành test/ chưa hoàn thành test/ trễ tiến độ bàn giao.
* Đánh giá các tiêu chí hoàn thành như phạm vi kiểm tra, chất lượng, chi phí, thời gian, mục tiêu kinh doanh quan trọng.
* Ngoài ra, giai đoạn này cũng thảo luận tất cả những điểm tốt, điểm chưa tốt và rút ra bài học kinh nghiệm cho những dự án sau, giúp cải thiện quy trình kiểm thử.
* Đầu ra:

Đầu ra của giai đoạn này bao gồm các tài liệu: Test report, Test results (final).

### **1.4.4 Phân loại kiểm thử**

Có 2 phương pháp kiểm thử chính là: Static testing (Kiểm thử tĩnh) và Dynamic testing (Kiểm thử động).

**a) Static testing (Kiểm thử tĩnh)**

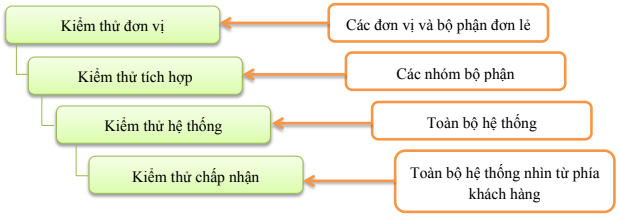
* Kiểm thử tĩnh là một hình thức của kiểm thử phần mềm mà phần mềm không được sử dụng thực sự. Thường thì nó không kiểm thử chi tiết mà chủ yếu kiểm tra tính đúng đắn của code (mã lệnh), thuật toán hay tài liệu. Chủ yếu kiểm tra cú pháp của code/ hoặc review code (kiểm tra xem code có được viết đúng tiêu chuẩn code. Đây là loại kiểm thử có thể được sử dụng bởi DEV (những người lập trình), làm việc một cách độc lập. Các kỹ thuật review code , kiểm tra và walkthroughs cũng được sử dụng trong test tĩnh này. Kiểm thử tĩnh liên quan đến việc xem xét các yêu cầu và các tài liệu thiết kế chi tiết.
* Kiểm thử tĩnh cũng có thể được tự động hóa. Nó sẽ thực hiện kiểm tra toàn bộ bao gồm các chương trình được phân tích bởi một trình thông dịch hoặc biên dịch mà xác nhận tính hợp lệ về cú pháp của chương trình.
* Các kỹ thuật kiểm thử tĩnh giúp nâng cao chất lượng phần mềm bằng cách hỗ trợ các nhà phát triển nhận ra và sửa chữa các sai sót của họ trong quá trình phát triển phần mềm.

**b) Dynamic testing (Kiểm thử động)**

* Kiểm thử tự động là việc sử dụng phần mềm đặc biệt (tách biệt với phần mềm đang được kiểm thử) để kiểm soát việc thực hiện các bài kiểm tra và so sánh kết quả thực tế với kết quả dự đoán. Kiểm thử tự động có thể tự động hóa một số nhiệm vụ lặp đi lặp lại nhưng cần thiết trong một quá trình kiểm thử đã được chính thức hóa, hay là các kiểm thử bổ sung nhưng sẽ khó thực hiện thủ công. Kiểm thử động kiểm tra cách thức hoạt động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian. Trong kiểm thử động, phần mềm phải thực sự được biên dịch và chạy. Kiểm thử động thực sự bao gồm: làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem liệu đầu ra có như mong muốn hay không.
* Các phương pháp kiểm thử động gồm có: Unit test (Kiểm thử đơn vị), Integration Tests (Kiểm thử tích hợp), System Tests (Kiểm thử hệ thống) và Acceptance Tests (Kiểm thử chấp nhận).

### **1.4.5 Các cấp độ kiểm thử phần mềm**

* Kiểm thử phần mềm gồm có 4 cấp độ: Unit test (Kiểm thử đơn vị), Integration Tests (Kiểm thử tích hợp), System Tests (Kiểm thử hệ thống) và Acceptance Tests (Kiểm thử chấp nhận). Tùy theo yêu cầu và đặc trưng của từng hệ thống, khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người quản lý dự án sẽ quyết định những loại kiểm thử được sử dụng.



*Hình 1. 3 Các cấp độ kiểm thử phần mềm*

**a) Unit Test (Kiểm thử đơn vị)**

* Unit (Đơn vị) là một thành phần phần mềm nhỏ nhất có thể kiểm thử được. Các hàm (Function), thủ tục (Procedure), lớp (Class) hay phương thức (Method) đều có thể được xem là Unit. Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, vì vậy thường không khó khăn gì trong việc tổ chức kiểm thử, ghi nhận và phân tích kết quả kiểm thử. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn vị đang kiểm tra. Một nguyên lý đúc kết từ thực tiễn: thời gian tốn cho Unit Test sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức kiểm thử sau đó.
* Mục đích: Đảm bảo thông tin được xử lý đúng và có đầu ra chính xác trong mối tương quan giữa dữ liệu nhập và chức năng của đơn vị.
* Người thực hiện: Do việc kiểm thử đơn vị đòi hỏi phải kiểm tra từng nhánh lệnh, nên đòi hỏi người kiểm thử có kiến thức về lập trình cũng như về thiết kế của hệ thống nên người thực hiện thường là lập trình viên.
* Cùng với các mục kiểm thử khác, unit test cũng đòi hỏi phải chuẩn bị trước test case (ca kiểm thử) hoặc test script (kịch bản kiểm thử), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các test case và test script này nên được giữ lại để tái sử dụng.

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

* Kiểm thử tích hợp là kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thiện. Trong khi unit test kiểm tra các thành phần và đơn vị riêng lẻ thì kiểm thử tích hợp kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.
* Mục đích:
* Phát hiện lỗi giao tiếp xảy ra giữa các đơn vị cũng như lỗi của bản thân từng đơn vị (nếu có).
* Tích hợp các đơn vị riêng lẻ thành các hệ thống nhỏ(subsystem) và cuối cùng là tích hợp các hệ thống nhỏ thành một hệ thống hoàn chỉnh (system) để chuẩn bị cho kiểm thử hệ thống.
* Người thực hiện: Thường là lập trình viên.
* Lưu ý:
* Kiểm thử tích hợp chỉ nên thực hiện trên từng đơn vị đã được kiểm tra cẩn thận trước đó bằng kiểm thử đơn vị, và tất cả các lỗi mức đơn vị đã được sửa chữa.
* Nên tích hợp dần từng đơn vị: Một đơn vị nên được tích hợp vào một nhóm các đơn vị khác đã được tích hợp và hoàn thành kiểm thử tích hợp trước đó vì khi đó chỉ cần kiểm tra giao tiếp giữa đơn vị mới được thêm vào với nhóm các đơn vị đã được tích hợp trước đó.

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

* Kiểm thử hệ thống bắt đầu khi tất cả các đơn vị của hệ thống được tích hợp thành công. Đây là công đoạn kiểm thử tốn nhiều công sức và thời gian hơn cả. Và đặc biệt, công đoạn này thường đòi hỏi được thực bởi một nhóm nhân viên tách biệt với nhóm phát triển, có chuyên môn và kinh nghiệm kiểm thử.
* Kiểm thử hệ thống gồm nhiều loại kiểm thử khác nhau, trong số đó, các mục tiêu kiểm thử quan trọng nhất là:
* Kiểm thử chức năng.
* Kiểm thử hiệu năng.
* Kiểm thử an toàn thông tin.
* Mục đích: kiểm tra xem hệ thống được làm ra có thỏa mãn yêu cầu hay không về nhiều khía cạnh: hoạt động, độ tin cậy, hiệu năng của hệ thống.
* Người thực hiện: Nhóm nhân viên kiểm thử.

Lưu ý: Việc lập kế hoạch cho kiểm thử hệ thống nên bắt đầu từ giai đoạn bắt đầu dự án.

**d) Acceptance Test (Kiểm thử chấp nhận)**

* Mục đích: Kiểm thử chấp nhận còn gọi là kiểm thử nghiệm thu nhằm mục đích chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng đã chấp nhận sản phẩm.
* Người thực hiện: Khách hàng.
* Có 2 phương pháp kiểm thử chấp nhận: Kiểm thử Alpha và kiểm thử Beta.
* Kiểm thử Alpha: người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm dưới sự hỗ trợ của nhân viên kiểm thử, nhân viên kiểm thử sẽ ghi nhận các lỗi hoặc phản hồi của khách hàng và báo lại với đơn vị phát triển phần mềm để lên kế hoạch sửa chữa.
* Kiểm thử Beta: Phần mềm sẽ được gửi tới cho người dùng để kiểm thử trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi lại cho đơn vị phát triển phần mềm để lên kế hoạch sửa chữa.

### **1.4.6 Thiết kế 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.
* Mục đích sử dụng:
* Tạo ra các ca kiểm thử tốt nhất có khả năng phát hiện ra lỗi, sai sót của phần mềm một cách nhiều nhất.
* Tạo ra các ca kiểm thử có chi phí rẻ nhất, đồng thời tốn ít thời gian và công sức nhất.
* Giúp người tester ít kinh nghiệm có thể biết những việc cần làm để đảm bảo chất lượng của phần mềm.
* Giúp nhà quản lý dự án biết được chức năng nào có lỗi, chức năng nào lỗi nhiều, lỗi ít.
* Cấu trúc một test case thường bao gồm các thông tin:
* *Test Case ID (Mã và tên của test case)*: Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.
* *Test Items (Mục đích kiểm thử)*: Mô tả mục đích sử dụng của test case. Giúp Tester hiểu và thực hiện đúng khi kiểm thử phần mềm theo test case mô tả.
* *Pre-condition (Điều kiện tiên quyết)*: Mô tả điều kiện cần có để có thể thực hiện test case này.
* *Test Steps (Mô tả các bước)*: Mô tả cụ thể các bước cần thực hiện để tái hiện nội dung test case khi Tester thực hiện kiểm thử phần mềm.
* *Test Data (Dữ liệu đầu vào)*: Là dữ liệu nhập vào các trường của phần mềm để thực hiện kiểm thử.
* *Expected results (Kết quả mong đợi)*: Một test case được viết tốt cần phải đề cập một cách rõ ràng kết quả mong đợi của ứng dụng hoặc hệ thống. Chỉ ra những gì mong đợi như là đầu ra của bước kiểm tra đó.

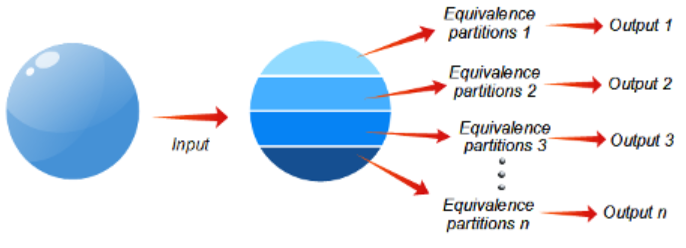
**a) Các bước viết một test case**

* *Bước 1*: Xác định mục đích test, cần hiểu rõ đặc tả yêu cầu của khách hàng.
* *Bước 2*: Xác định chức năng testing, cần phải biết làm thế nào phần mềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau. Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button.
* *Bước 3*: Xác định các yêu cầu phi chức năng, yêu cầu phần cứng, hệ điều hành, các khía cạnh an ninh.
* *Bước 4*: Xác định biểu mẫu cho test case, bao gồm giao diện UI, chức năng, khả năng tương thích và hiệu suất…
* *Bước 5*: Xác định tính ảnh hưởng giữa các nguyên tắc module test case nên được thiết kế để có thể bao phủ được sự ảnh hưởng của các module với nhau ở mức độ cao nhất.

**b) Các kỹ thuật viết test case**

**Equivalence Partitioning (Phân vùng tương đương)**

* Ý tưởng: 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 kiểm tra một giá trị đại diện trong vùng tương đương.



*Hình 1. 4 Kỹ thuật phân vùng tương đương*

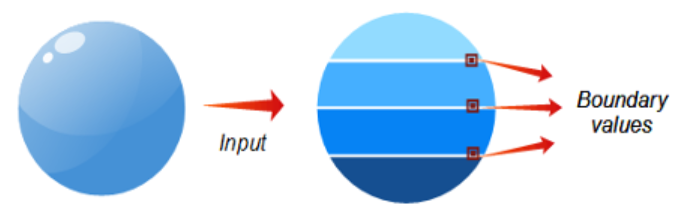
* Thiết kế test case bằng kỹ thuật phân vùng tương đương tiến hành theo 2 bước:
* *Bước 1*: Xác định các lớp tương đương.
* *Bước 2*: Xác định các ca kiểm thử.
* Ưu/ Nhược điểm của kỹ thuật phân vùng tương đương:
* *Ưu điểm*: Vì mỗi vùng tương đương ta chỉ cần test trên các phần tử đại diện nên số lượng test case được giảm đi khá nhiều nhờ đó mà thời gian thực hiện test cũng giảm đáng kể.
* *Nhược điểm*: Không phải với bất kỳ bài toán nào đều có thể áp dụng kỹ thuật này. Có thể bị sót lỗi ở biên nếu chỉ chọn giá trị ở khoảng giữa của miền tương đương. Vì vậy việc kết hợp linh hoạt giữa kỹ thuật phân vùng tương đương và phân tích giá trị biên dưới đây sẽ mang lại hiệu quả cao hơn để vừa tối ưu số lượng test case và vẫn đảm bảo đươc chất lượng phần mềm.

**Boundary-value Analysis (Phân tích giá trị biên)**

- Ý tưởng: Phân tích giá trị biên là trường hợp đặc biệt của phân vùng tương đương, 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.

- Các case chuẩn được lựa chọn dựa vào quy tắc sau:

* Giá trị biên nhỏ nhất ± 1.
* Giá trị biên nhỏ nhất.
* Giá trị biên lớn nhất.
* Giá trị biên lớn nhất ± 1.



*Hình 1. 5 Kỹ thuật phân tích giá trị biên*

- Ưu/ Nhược điểm của kỹ thuật phân tích giá trị biên:

* *Ưu điểm*: Thay vì phải kiểm tra hết toàn bộ các giá trị trong từng vùng tương đương, kỹ thuật phân tích giá trị biên tập trung vào việc kiểm thử các giá trị biên của miền giá trị inputs để thiết kế test case do “lỗi thường tiềm ẩn tại các ngõ ngách và tập hợp tại biên”. Do đó, tiết kiệm thời gian thiết kế test case và thực hiện test.
* *Nhược điểm*: Phương pháp này chỉ hiệu quả trong trường hợp các đối số đầu vào độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn.

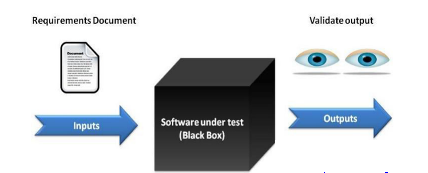
**Error Guessing (Đoán lỗi)**

* Ý tưởng: Phương pháp này không có quy trình cụ thể vì có tính trực giác cao và không thể dự đoán trước. Phương pháp chỉ phù hợp với những Tester có kinh nghiệm. Họ phỏng lỗi phần mềm dựa vào trực giác, dựa vào kinh nghiệm, dữ liệu lịch sử về các lỗi đã từng xảy ra với chương trình trước đó… và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.
* Ưu/ Nhược điểm:
* *Ưu điểm*: Sử dụng phương pháp này có thể giúp tester tìm ra những lỗi điển hình thường xảy ra trong phần mềm hoặc những lỗi không thể tìm thấy khi thiết kế test case theo hình thức formal.
* *Nhược điểm*: Kỹ thuật này thường được thực hiện bởi các Tester có kinh nghiệm và không theo một quy tắc nhất định, thiết kế test case dựa nhiều vào cảm tính.
* Ngoài 3 kỹ thuật thiết kế các trường hợp kiểm thử đã nói ở trên, còn rất nhiều các kỹ thuật kiểm thử khác như: thiết kế các trường hợp kiểm thử dựa trên đồ thị nguyên nhân – kết quả (Cause-Effect Diagram), dựa trên bảng quyết định (Decision Tables)…
* Để giảm thiểu số case đến mức tối ưu mà vẫn đảm bảo chất lượng phần mềm, mỗi tester cần linh hoạt trong việc lựa chọn các kỹ thuật thiết kế các trường hợp kiểm thử.

## **Các kỹ thuật kiểm thử phần mềm**

### **Kiểm thử hộp đen (Black box testing)**

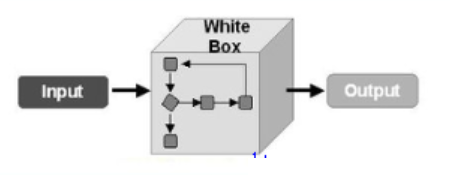
* Kiểm thử hộp đen hay còn gọi là kiểm thử hướng dữ liệu. Là phương pháp kiểm thử không quan tâm đến cấu trúc bên trong của phần mềm (cấu trúc dữ liệu, thuật toán, xuer lý…) mà chỉ kiểm tra dữ liệu đầu vào, dữ liệu đầu ra ứng với các trường hợp trong đặc tả (Testcase).
* Kiểm thử hộp đen không có “mối ràng buộc” nào với code và nhận thức của một tester rất đơn giản : một source code có rất nhiều lỗi => Các tester sẽ tìm thấy được nhiều lỗi nơi mà Developer không tìm thấy.



*Hình 1. 6 Kiểm thử hộp đen*

### **Kiểm thử hộp trắng (White box testing)**

* Kỹ thuật kiểm thử hộp trắng hay còn gọi là “kiểm thử cấu trúc” là kỹ thuật kiểm thử dựa vào giải thuật cụ thể, cấu trúc dữ liệu bên trong của đơn vị phần mềm cần kiểm thử để xác định đơn vị phần mềm đó có thực hiện đúng hay không?

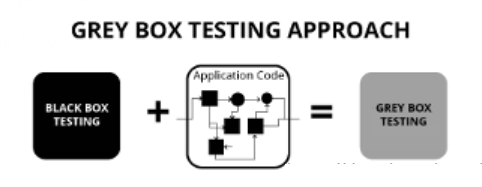


*Hình 1. 7 Kiểm thử hộp trắng*

* Kiểm thử hộp trắng thường tốn rất nhiều thời gian và công sức nếu phần mềm quá lớn (thí dụ trong kiểm thử tích hợp hay kiểm thử chức năng). Do đó kỹ thuật này chủ yếu được dùng để kiểm thử đơn vị, kiểm thử từng tác vụ của một lớp chức năng.
* Tiêu chuẩn của kiểm thử hộp trắng phải đáp ứng các yêu cầu như sau:
* Bao phủ dòng lệnh: mỗi dòng lệnh ít nhất phải được thực thi 1 lần.
* Bao phủ nhánh: mỗi nhánh trong sơ đồ điều khiển phải được đi qua một lần.
* Bao phủ đường: tất cả các đường từ điểm khởi tạo đến điểm cuối cùng trong sơ đồ dòng điều khiển phải được đi qua.
* Ưu/ Nhược điểm:
* Ưu điểm: Buộc các chuyên gia kiểm thử phải suy luận cẩn thận về việc kiểm thử lỗi vì vậy lỗi sẽ được triệt để, cho phép tìm kiếm các lỗi ẩn bên trong. Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.
* Nhược điểm: Khá mất thời gian và công sức nhưng vẫn sẽ tồn tại lỗi. Đòi hỏi người kiểm thử có kinh nghiệm và am hiểu về kiểm thử cũng như về cấu trúc bên trong của phần mềm được thử nghiệm.

### **Kiểm thử hộp xám**

* Kiểm thử hộp xám là một phương pháp kiểm thử phần mềm được kết hợp giữa phương pháp kiểm thử hộp đen và kiểm thử hộp trắng. Trong kiểm thử hộp đen tester kiểm thử các hạng mục mà không cần biết cấu trúc bên trong của chương trình, còn kiểm thử hộp trắng thì tester biết được cấu trúc bên trong của chương trình. Trong kiểm thử hộp xám, cấu trúc bên trong sản phẩm chỉ được biết một phần. Tester có thể truy cập vào cấu trúc dữ liệu bên trong và thuật toán của chương trình với mục đích là để thiết kế các trường hợp kiểm thử, nhưng khi thực hiện kiểm thử thì test như người dùng cuối hoặc là ở mức hộp đen.



*Hình 1. 8 Kiểm thử hộp xám*

* Mặc dù phương pháp kiểm thử hộp xám có thể được dùng trong các mức khác nhau của kiểm thử, tuy nhiên nó chủ yếu được sử dụng trong kiểm thử tích hợp.
* Kiểm thử hộp xám nhằm tìm ra tối đa số lỗi về cấu trúc dữ liệu của hộp trắng và lỗi chức năng của hộp đen. Trong kiểm thử hộp xám viết các trường hợp kiểm thử dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình). Thực hiện kiểm thử trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới kiểm thử được, không can thiệp vào code).
* Ưu/ Nhược điểm:
* Ưu điểm: Hoạt động tốt cho các đoạn mã lớn, các vai trò được xác định rõ ràng cho người dùng và nhà phát triển trong quá trình thử nghiệm. Thử nghiệm dựa trên yêu cầu của người dùng chứ không phải người thiết kế.
* Nhược điểm: Hầu hết các trường hợp kiểm thử đều khó thiết kế. Kiểm thử hộp xám không được coi là một phương pháp hiệu quả vì không có nhiều kịch bản để kiểm thử