Quy Trình Kiểm Thử Phần Mềm

Software Testing Life Cycle – STLC

Người trình bày:

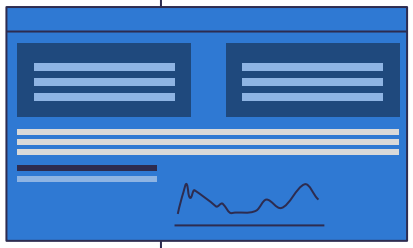
Đoàn Thị Kim Nhung

BA Team

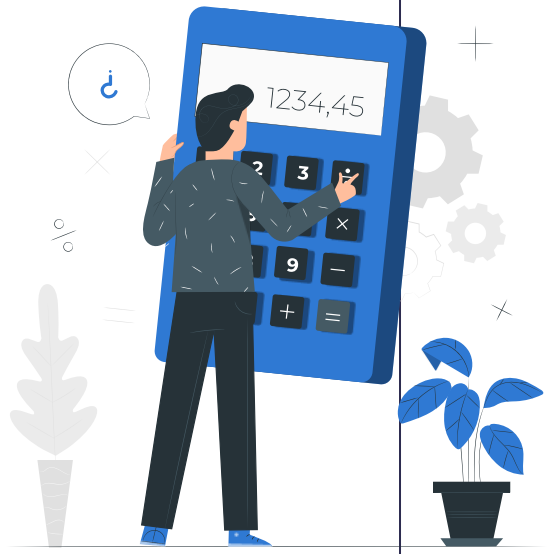
Mục Lục

Từ ngữ viết tắt và định nghĩa

#	Từ ngữ viết tắt	Định nghĩa
1	PTPM	Phát triển phần mềm
2	BE	Backend
3		
4		
5		
6		
7		
8		



01



Vòng đời/Quy trình Kiểm thử phần mềm

Vòng đời/Quy trình KTPM

1. Requirements Analysis

Giai đoạn phân tích yêu cầu

2. Test Planning

Giai đoạn lập kế hoạch kiểm thử

3. Test Case Development

Giai đoạn phát triển ca kiểm thử

4. Environment Setup

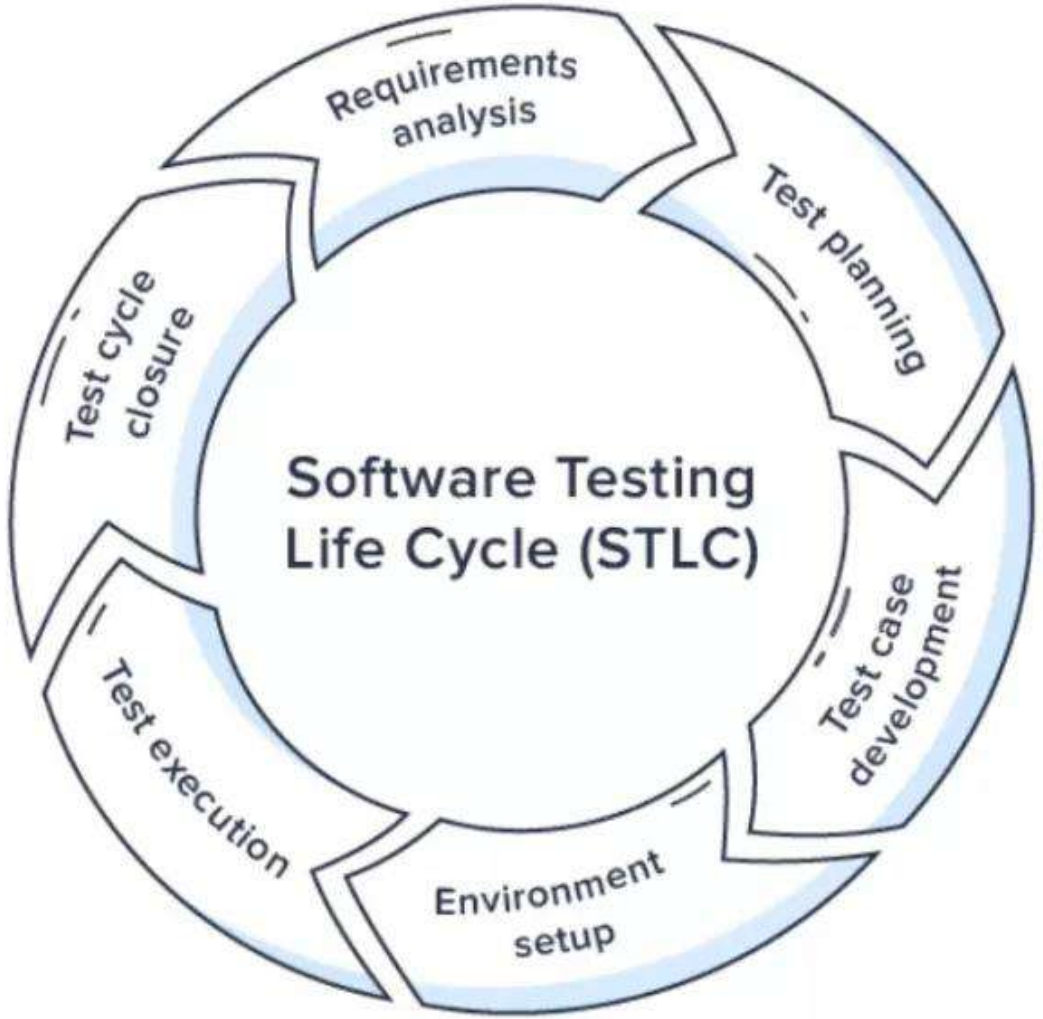
Giai đoạn thiết lập môi trường kiểm thử

5. Test Execution

Giai đoạn thực hiện kiểm thử

6. Test Cycle Closure

Giai đoạn kết thúc kiểm thử



Vòng đời/Quy trình PTPM

1. Requirements Analysis: Giai đoạn phân tích yêu cầu

- Xác định và hiểu yêu cầu của khách hàng hoặc người dung.
- Phân tích và đánh giá tính khả thi của yêu cầu.
- Xác định yêu cầu chức năng và phi chức năng.
- Xác định các trường hợp kiểm thử cơ bản dựa trên yêu cầu.

#	Đầu vào	Đầu ra
1	Tài liệu yêu cầu (BRD/URD/User Strory/SRS/FSD).	Bảng yêu cầu kiểm thử.
2	Hội thảo với khách hàng hoặc người dùng.	Danh sách các tính năng và yêu cầu cần kiểm thử.
3		Các trường hợp kiểm thử sơ bộ.

Vòng đời/Quy trình PTPM

2. Test Planning: Giai đoạn lập kế hoạch kiểm thử

- Xác định phạm vi và mục tiêu của kiểm thử.
- Xác định tài nguyên và kế hoạch lịch trình.
- Xác định các phương pháp và kỹ thuật kiểm thử.
- Xác định tiêu chí chấp nhận và điều kiện dừng kiểm thử.
- Tạo kế hoạch kiểm thử chi tiết.

#	Đầu vào	Đầu ra
1	Bảng yêu cầu kiểm thử.	Kế hoạch kiểm thử.
2	Kế hoạch dự án.	Chiến lược kiểm thử.
3	Ngân sách và tài nguyên.	Lịch trình kiểm thử.
4		Đánh giá rủi ro.

Vòng đời/Quy trình PTPM

3. Test Case Development: Giai đoạn phát triển ca kiểm thử

- Dựa trên yêu cầu và kế hoạch kiểm thử, phát triển các ca kiểm thử chi tiết.
- Mô tả các bước cần thực hiện để thực hiện ca kiểm thử.
- Xác định dữ liệu kiểm thử cần thiết.

#	Đầu vào	Đầu ra
1	Bảng yêu cầu kiểm thử.	Các ca kiểm thử chi tiết.
2	Kế hoạch kiểm thử.	Tài liệu mô tả ca kiểm thử.
3	Tài liệu thiết kế.	

Vòng đời/Quy trình PTPM

4. Environment Setup: Giai đoạn thiết lập môi trường kiểm thử

- Xây dựng một môi trường kiểm thử có thể tái tạo lại môi trường sản xuất.
- Cài đặt và cấu hình các công cụ kiểm thử cần thiết.
- Chuẩn bị dữ liệu kiểm thử và mô phỏng dữ liệu sản xuất.

#	Đầu vào	Đầu ra
1	Kế hoạch kiểm thử.	Môi trường kiểm thử.
2	Yêu cầu cấu hình môi trường.	Hệ thống kiểm thử đã cấu hình.
3	Các công cụ kiểm thử.	

Vòng đời/Quy trình PTPM

5. Test Execution: Giai đoạn thực hiện kiểm thử

- Thực hiện các ca kiểm thử theo kế hoạch.
- Ghi nhận kết quả kiểm thử và các lỗi phát sinh.
- Đảm bảo rằng tất cả các ca kiểm thử đã được thực hiện và kết quả đã được ghi lại.

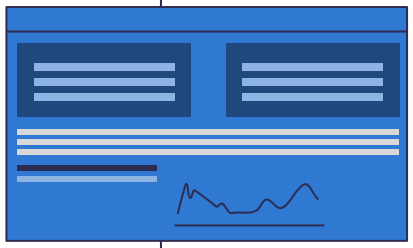
#	Đầu vào	Đầu ra
1	Môi trường kiểm thử.	Kết quả kiểm thử.
2	Các ca kiểm thử.	Báo cáo lỗi.
3	Dữ liệu kiểm thử.	Báo cáo tiến độ kiểm thử.

Vòng đời/Quy trình PTPM

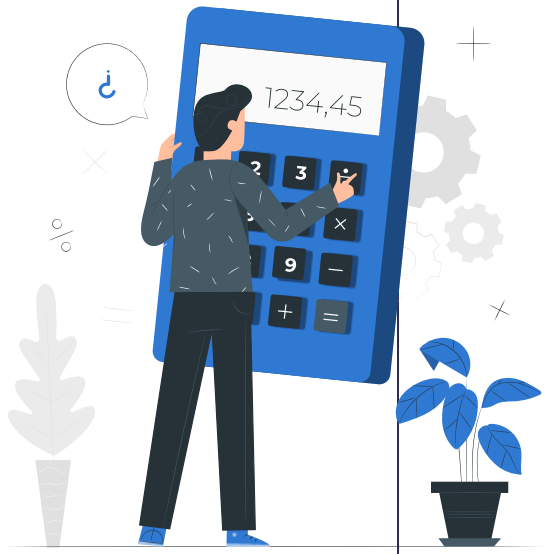
6. Test Cycle Closure: Giai đoạn kết thúc kiểm thử

- Đánh giá kết quả kiểm thử và so sánh với tiêu chí chấp nhận.
- Tạo báo cáo về kết quả kiểm thử và các lỗi đã tìm thấy.
- Lưu trữ tất cả các tài liệu liên quan đến kiểm thử.
- Tổng kết và học hỏi từ quá trình kiểm thử để cải thiện trong tương lai.

#	Đầu vào	Đầu ra
1	Kết quả kiểm thử.	Báo cáo kết quả kiểm thử.
2	Báo cáo lỗi.	Bản rút kinh nghiệm và cải tiến.
3	Báo cáo tiến độ kiểm thử.	

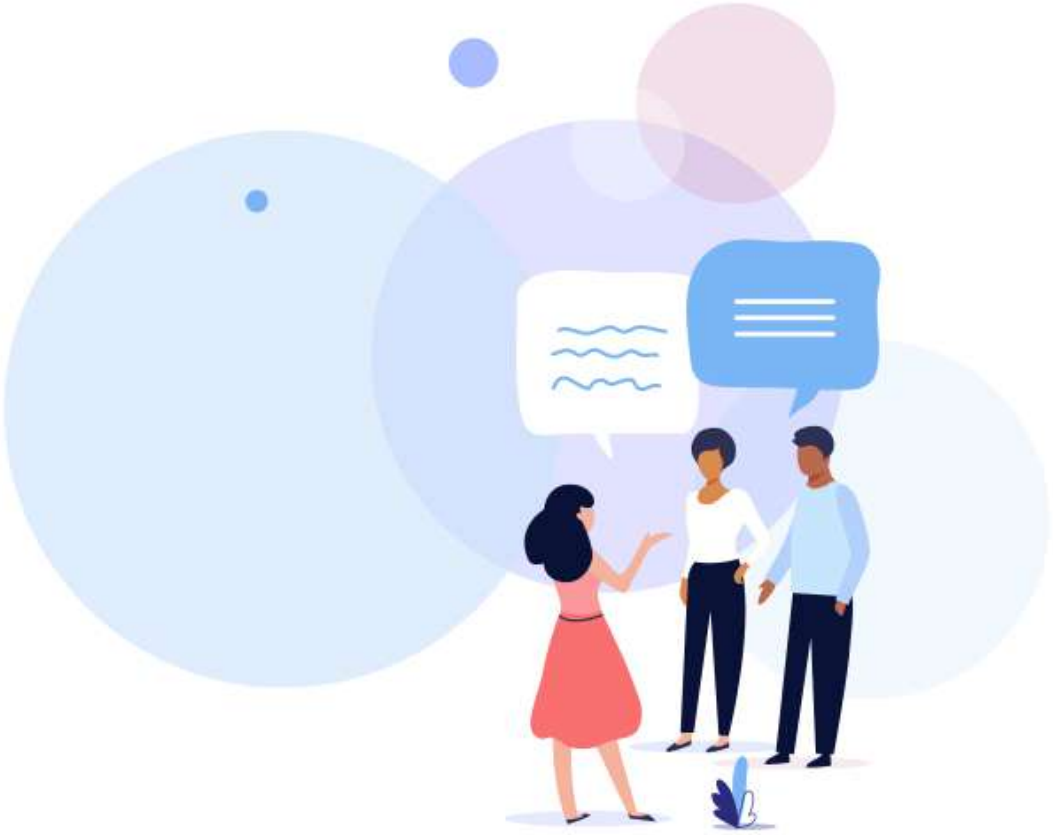


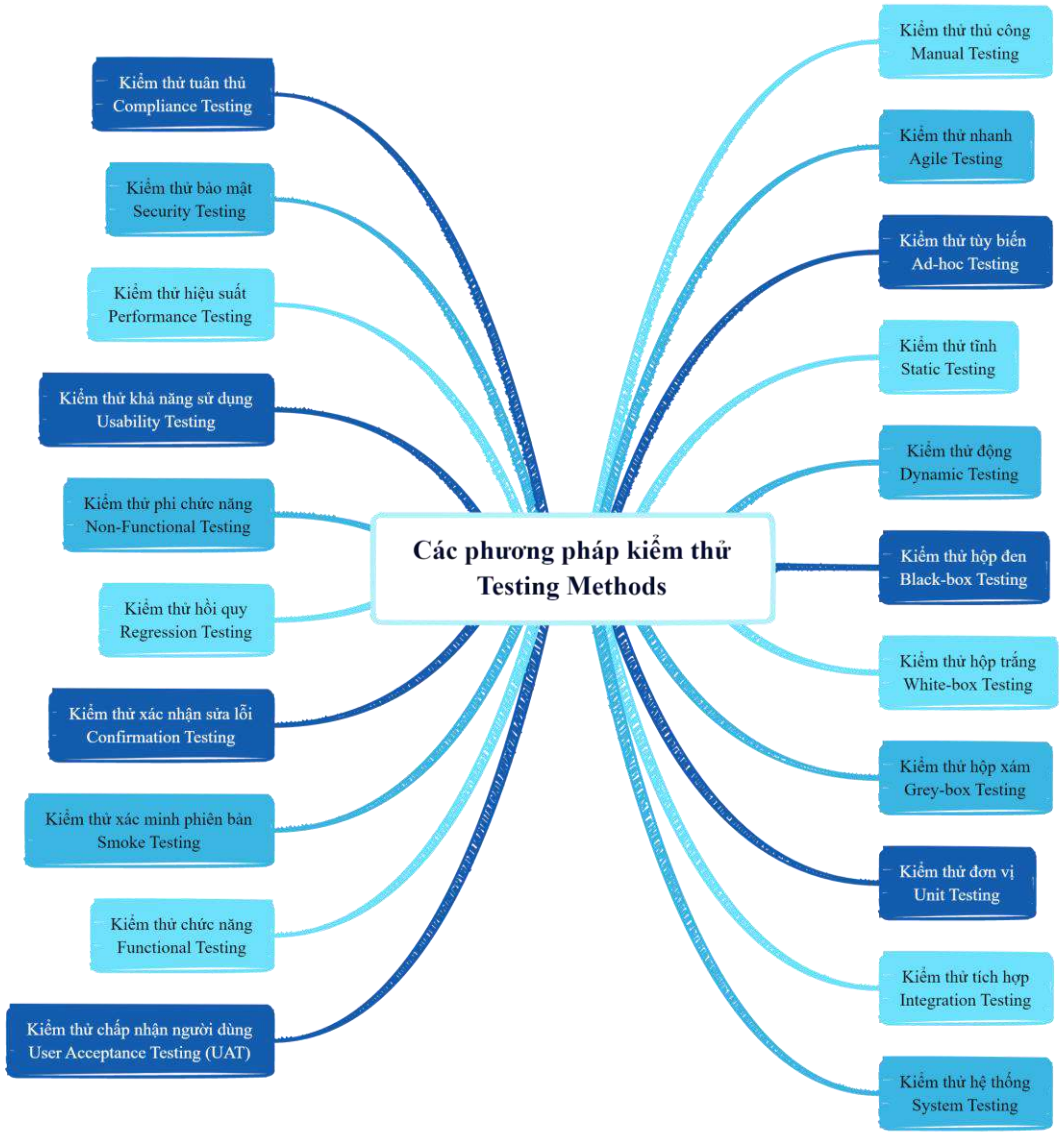
02



Các phương pháp Kiểm thử phần mềm

Các phương pháp KTPM





Các phương pháp KTPM

1. Kiểm thử thủ công (Manual Testing)

Khái niệm:

Kiểm thử thủ công (Manual Testing) là một phương pháp kiểm thử phần mềm, đây là quá trình kiểm tra các tính năng, chức năng và hiệu suất của phần mềm một cách thủ công, tức là bằng cách sử dụng con người để thực hiện các ca kiểm thử mà không có sự hỗ trợ từ các công cụ tự động hóa.

Ý nghĩa và mục đích:

- Phát hiện lỗi.
- Đảm bảo chất lượng.
- Đảm bảo tính tương thích.
- Đảm bảo tính người dùng tốt nhất.
- Tạo bản báo cáo và tài liệu.

Áp dụng khi nào?

Khi không có tự động hóa; Kiểm thử thăm dò; Usability Testing – đo độ thân thiện, hiệu quả, hoặc thuận tiện phần mềm hoặc sản phẩm cho người dùng cuối; Kiểm thử Ad-hoc – không có phương pháp cụ thể; Kiểm thử động và sáng tạo.

Các phương pháp KTPM

2. Kiểm thử tự động (Automated Testing)

Khái niệm:

Kiểm thử tự động (Automated Testing) là một phương pháp kiểm thử phần mềm, sử dụng công cụ và kịch bản kiểm thử tự động để thực hiện việc kiểm tra và xác nhận tính năng của phần mềm một cách tự động, thay vì phải thực hiện bằng tay như trong Manual Testing.

Ý nghĩa và mục đích:

- Tự động hóa quy trình kiểm thử.
- Tăng tốc độ và hiệu quả.
- Tăng độ chính xác.
- Tái sử dụng và duy trì dễ dàng.
- Phát hiện lỗi sớm.
- Hỗ trợ kiểm thử liên tục.

Áp dụng khi nào?

Khi có một số lượng lớn các ca kiểm thử; Khi cần kiểm thử liên tục; Khi cần kiểm thử đa nền tảng và đa môi trường; Khi cần kiểm thử hiệu suất và khả năng mở rộng; Khi cần kiểm thử tích hợp.

Các phương pháp KTPM

3. Kiểm thử nhanh (Agile Testing)

Khái niệm:

Kiểm thử nhanh (Agile Testing) là một phương pháp kiểm thử phần mềm, đây là việc tiến hành kiểm thử trong một dự án phát triển phần mềm theo phương pháp Agile.

Ý nghĩa và mục đích:

- Đồng bộ với phương pháp và quy trình phát triển phần mềm Agile.
- Xác định và sửa chữa lỗi sớm ngay từ các giai đoạn đầu.
- Đảm bảo chất lượng liên tục.
- Tăng cường sự hợp tác giữa nhóm phát triển và khách hàng.
- Thực hiện các kiểm thử cần thiết và có giá trị cao nhất.
- Tạo được lòng tin từ khách hàng.

Áp dụng khi nào?

Khi dự án áp dụng quy trình Agile; Khi có các yêu cầu thay đổi thường xuyên; Sản phẩm phát triển theo hướng linh hoạt; Khi cần tương tác cao với khách hàng và người dùng cuối; Yêu cầu phải kiểm thử sản phẩm liên tục.

Các phương pháp KTPM

4. Kiểm thử tùy biến (Ad-hoc Testing)

Khái niệm:

Kiểm thử tùy biến (Ad-hoc Testing) là một phương pháp kiểm thử phần mềm, một phương pháp kiểm thử không có kế hoạch cụ thể hoặc tài liệu kiểm thử trước. Điều này có nghĩa là các bước kiểm thử được thực hiện một cách ngẫu nhiên, không theo quy trình cụ thể hoặc kế hoạch trước đó. Mục đích chính của Ad-hoc Testing là tìm kiếm các lỗi và vấn đề không được dự kiến một cách nhanh chóng và hiệu quả.

Ý nghĩa và mục đích:

- Phát hiện lỗi một cách nhanh chóng.
- Tăng khả năng phát hiện lỗi.
- Kiểm thử đa dạng.

Áp dụng khi nào?

Khi cần kiểm tra tính năng mới; Khi thời gian bị hạn chế; Khi cần kiểm tra các trường hợp biên; Khi cần kiểm tra tính năng không được xác định rõ; Khi cần kiểm thử ngẫu nhiên.

Các phương pháp KTPM

5. Kiểm thử tĩnh (Static Testing)

Khái niệm:

Kiểm thử tĩnh (Static Testing) là một phương pháp kiểm thử phần mềm, được thực hiện mà không cần thực thi chương trình. Thay vào đó, nó tập trung vào việc kiểm tra các yếu tố không chạy của phần mềm như mã nguồn, tài liệu, mô hình thiết kế, yêu cầu, và các tài liệu khác liên quan đến dự án.

Ý nghĩa và mục đích:

- Phát hiện lỗi sớm.
- Kiểm tra chuẩn mực đã được soạn thảo theo tài liệu trước đó.
- Cải thiện chất lượng từ những giai đoạn đầu tiên của quy trình.
- Tiết kiệm thời gian và chi phí.
- Tăng độ tin cậy.

Áp dụng khi nào?

Khi cần phát hiện lỗi sớm; Khi cần đảm bảo chất lượng và tuân thủ tiêu chuẩn; Khi cần tăng hiệu suất và hiệu quả của quy trình phát triển; Khi cần tăng độ tin cậy của phần mềm.

Các phương pháp KTPM

6. Kiểm thử động (Dynamic Testing)

Khái niệm:

Kiểm thử động (Dynamic Testing) là một phương pháp kiểm thử phần mềm thực thi mã nguồn để kiểm tra các tính năng và chức năng của phần mềm dưới các điều kiện chạy thực tế. Mục đích chính của Dynamic Testing là đảm bảo rằng phần mềm hoạt động đúng đắn và đáp ứng được các yêu cầu của người dùng cuối.

Ý nghĩa và mục đích:

- Kiểm tra tính đúng đắn theo các yêu cầu đã được xác định.
- Kiểm tra tính năng và chức năng để đảm bảo hoạt động như mong đợi.
- Kiểm tra hiệu suất của phần mềm dưới các điều kiện khác nhau để đảm bảo rằng nó có thể xử lý tải cao.
- Kiểm tra khả năng phục hồi sau khi gặp lỗi và xác định xem có thể hoạt động ổn định trong môi trường sản xuất hay không.
- **Áp dụng khi nào?**

Sau khi phát triển phần mềm; Khi có thay đổi trong mã nguồn; Trong quá trình kiểm tra và đánh giá chất lượng của sản phẩm để xác định các lỗi và vấn đề có thể xuất hiện và sửa chữa chúng; Trước khi phát hành sản phẩm để đảm bảo phần mềm hoạt động đúng đắn; Khi thực hiện kiểm thử tự động.

Các phương pháp KTPM

7. Kiểm thử hộp đen (Black-box Testing)

Khái niệm:

Kiểm thử hộp đen (Black-box Testing) là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các Nhà kiểm thử kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp. Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.

Ý nghĩa và mục đích:

- Kiểm tra xem các chức năng và tính năng của hệ thống hoạt động đúng cách theo yêu cầu.
- Xác minh yêu cầu.
- Kiểm tra tính tương thích trên các môi trường khác nhau, thiết bị khác nhau.
- Có thể sử dụng để kiểm tra các lỗ hổng bảo mật bằng cách nhập liệu không hợp lệ hoặc bất thường.

Áp dụng khi nào?

Khi không có thông tin chi tiết về cấu trúc nội bộ của phần mềm; Khi cần kiểm tra chức năng từ góc độ người dùng; Khi cần kiểm tra tính tương thích của hệ thống trên nhiều môi trường khác nhau; Khi cần tăng tính khách quan trong quá trình kiểm thử.

Các phương pháp KTPM

8. Kiểm thử hộp trắng (White-box Testing)

Khái niệm:

Kiểm thử hộp trắng (White-box Testing) là một phương pháp kiểm thử phần mềm, hay còn được gọi là kiểm thử cấu trúc hoặc kiểm thử logic, nhằm kiểm tra các thành phần bên trong của phần mềm như mã nguồn, cấu trúc dữ liệu, logic điều khiển và các thành phần khác.

Ý nghĩa và mục đích:

- Giúp xác định và sửa chữa các lỗi trong mã nguồn và cấu trúc của ứng dụng trước khi nó được triển khai.
- Đảm bảo rằng mã nguồn và cấu trúc của ứng dụng được kiểm tra một cách toàn diện.
- Kiểm tra tích hợp giữa các thành phần của ứng dụng, như giao diện người dùng và các thành phần BE.
- Kiểm tra và kiểm soát cấu trúc của mã nguồn.

Áp dụng khi nào?

Sử dụng trong quá trình kiểm thử unit, nơi các thành phần nhỏ nhất của phần mềm được kiểm tra độc lập để đảm bảo tính đúng đắn và hoạt động chính xác; Kiểm thử tích hợp; Kiểm tra toàn bộ hệ thống từ góc nhìn bên trong; Kiểm thử hồi quy để đảm bảo rằng các thay đổi hoặc cải tiến không làm ảnh hưởng đến các phần khác của hệ thống; Kiểm tra và tối ưu hóa hiệu suất của ứng dụng từ góc nhìn của mã nguồn và cấu trúc.

Các phương pháp KTPM

9. Kiểm thử hộp xám (Grey-box Testing)

Khái niệm:

Kiểm thử hộp xám (Grey-box Testing) là một phương pháp kiểm thử phần mềm, là một phương pháp kiểm thử mà Nhà kiểm thử có một phần thông tin về cấu trúc nội bộ hoặc thiết kế của phần mềm. Mặc dù không có kiến thức chi tiết như trong kiểm thử hộp trắng, nhưng cũng không hoàn toàn không biết như trong kiểm thử hộp đen.

Ý nghĩa và mục đích:

Kết hợp các lợi ích của cả hai phương pháp kiểm thử hộp đen và hộp trắng. Điều này giúp tối ưu hóa quá trình kiểm thử bằng cách sử dụng thông tin cấu trúc bên trong để tạo ra các ca kiểm thử chính xác hơn, đồng thời vẫn giữ được tính độc lập và khả năng phản ứng linh hoạt.

Áp dụng khi nào?

Khi không có đủ thông tin cấu trúc nội bộ; Khi cần tăng độ phủ kiểm thử; Khi cần tối ưu chi phí kiểm thử; Khi cần tối ưu hóa hiệu suất kiểm thử;

Các phương pháp KTPM

10. Kiểm thử đơn vị (Unit Testing)

Khái niệm:

Kiểm thử đơn vị (Unit Testing) là một phương pháp kiểm thử phần mềm, 1 trong 4 mức độ kiểm thử, tập trung vào việc kiểm tra từng thành phần cơ bản (đơn vị) của mã nguồn, như các hàm, phương thức, lớp, hay module.

Ý nghĩa và mục đích:

Mục tiêu chính của Unit Testing là đảm bảo rằng mỗi phần của mã nguồn hoạt động đúng như mong đợi khi được thực thi độc lập.

Áp dụng khi nào?

Khi một đơn vị mã nguồn mới được viết; Có thay đổi trong mã nguồn; Trước khi thực hiện tích hợp; Khi xảy ra lỗi hoặc sự cố trong quá trình phát triển hoặc kiểm thử, việc viết các test case unit mới hoặc cập nhật các test case hiện có giúp phát hiện và khắc phục lỗi một cách nhanh chóng và chính xác.

Các phương pháp KTPM

11. Kiểm thử tích hợp (Integration Testing)

Khái niệm:

Kiểm thử tích hợp (Integration Testing) là một phương pháp kiểm thử phần mềm, 1 trong 4 mức độ kiểm thử, nơi các thành phần được kết hợp và kiểm tra hoạt động cùng nhau như thế nào. Mục tiêu chính của Integration Testing là đảm bảo rằng các thành phần đã được tích hợp hoạt động đúng cách khi được kết nối lại với nhau. Trong quá trình này, các lỗi về giao diện và tương tác giữa các thành phần thường được phát hiện và sửa chữa.

Ý nghĩa và mục đích:

- Kiểm tra tích hợp chức năng: Các chức năng của hệ thống hoạt động đúng khi được tích hợp với nhau.
- Kiểm tra giao diện: Xác định và sửa các lỗi liên quan đến việc truyền dữ liệu giữa các thành phần.
- Kiểm tra tương tác: Các thành phần tương tác với nhau theo cách mong muốn và không gây ra lỗi.
- Kiểm tra xử lý lỗi: Hệ thống xử lý các trường hợp lỗi một cách chính xác và nhất quán khi tích hợp.

Áp dụng khi nào?

Integration Testing thường được thực hiện sau khi các thành phần phần mềm đã được độc lập kiểm thử trong Unit Testing và trước khi chúng được tích hợp vào hệ thống hoàn chỉnh để kiểm thử (System Testing).

Các phương pháp KTPM

12. Kiểm thử hệ thống (System Testing)

Khái niệm:

Kiểm thử hệ thống (System Testing) là một phương pháp kiểm thử phần mềm, 1 trong 4 mức độ kiểm thử, tập trung vào việc kiểm tra toàn bộ hệ thống phần mềm đã được tích hợp hoàn chỉnh.

Ý nghĩa và mục đích:

Trong quá trình này, mục tiêu chính là đảm bảo rằng hệ thống hoạt động đúng đắn và đáp ứng đúng các yêu cầu kỹ thuật và chức năng đã được xác định.

Áp dụng khi nào?

System Testing thường được thực hiện sau khi tất cả các thành phần của hệ thống đã được tích hợp và kiểm tra ở cấp độ đơn vị và tích hợp (Unit và Integration Testing).

Các phương pháp KTPM

13. Kiểm thử chấp nhận người dùng (User Acceptance Testing - UAT)

Khái niệm:

Kiểm thử chấp nhận người dùng (User Acceptance Testing - UAT) là một phương pháp kiểm thử phần mềm, 1 trong 4 mức độ kiểm thử, quá trình kiểm thử cuối cùng được thực hiện trước khi phần mềm hoặc hệ thống được triển khai cho người dùng cuối. Trong quá trình UAT, người dùng cuối hoặc các bên liên quan sẽ kiểm tra và xác nhận rằng hệ thống hoạt động đúng theo yêu cầu kỹ thuật và đáp ứng được nhu cầu kinh doanh.

Ý nghĩa và mục đích:

Đảm bảo rằng phần mềm hoặc hệ thống đã được phát triển đáp ứng đúng các yêu cầu và mong đợi của người dùng cuối, đồng thời kiểm tra tính chất sử dụng và trải nghiệm người dùng.

Áp dụng khi nào?

UAT thường được thực hiện trong giai đoạn cuối của quy trình phát triển phần mềm, sau khi các loại kiểm thử khác như Unit Testing, Integration Testing và System Testing đã hoàn thành. UAT thường diễn ra trước khi phần mềm hoặc hệ thống được triển khai vào môi trường sản xuất hoặc sử dụng rộng rãi.

Các phương pháp KTPM

14. Kiểm thử chức năng (Functional Testing)

Khái niệm:

Kiểm thử chức năng (Functional Testing) là một phương pháp kiểm thử phần mềm, phương pháp đảm bảo rằng phần mềm hoạt động đúng theo yêu cầu chức năng đã được xác định.

Ý nghĩa và mục đích:

- Kiểm tra tính đúng đắn và hoạt động của các chức năng của phần mềm.
- Phát hiện và báo cáo về các lỗi và sự không phù hợp trong chức năng của phần mềm.
- Tăng tính ổn định của phần mềm.
- Phát hiện rủi ro trong quá trình phát triển và triển khai phần mềm.
- Đảm bảo rằng sản phẩm phần mềm hoạt động như mong đợi và đáp ứng được nhu cầu của khách hàng.

Áp dụng khi nào?

Functional Testing được thực hiện ở mức độ kiểm thử hệ thống (System Testing). Đây là giai đoạn trong quy trình kiểm thử phần mềm, nơi phần mềm hoàn chỉnh được kiểm tra hoạt động như mong đợi và đáp ứng các yêu cầu chức năng đã được xác định. Trong quá trình này, các chức năng của phần mềm được kiểm tra để đảm bảo tính đúng đắn, tính hoạt động và hiệu suất.

Các phương pháp KTPM

15. Kiểm thử xác minh phiên bản (Smock Testing)

Khái niệm:

Kiểm thử xác minh phiên bản (Smock Testing) là một phương pháp kiểm thử phần mềm, còn được gọi là Build Verification Testing (BVT), là một loại kiểm thử hạt nhân dùng để kiểm tra sự ổn định cơ bản của một phiên bản phần mềm sau khi đã được cài đặt xong và trước khi bắt đầu kiểm thử chi tiết.

Ý nghĩa và mục đích:

- Đảm bảo rằng phiên bản phần mềm mới cài đặt đã hoạt động như mong đợi ở mức độ cơ bản. Chỉ tập trung vào việc xác định xem có vấn đề nào nghiêm trọng cần xử lý ngay lập tức hay không.
- Nếu Smoke Test thành công, nghĩa là phiên bản phần mềm đó đạt được một số tiêu chí cơ bản và có thể tiếp tục vào các giai đoạn kiểm thử chi tiết hơn. Ngược lại, có thể yêu cầu các nhà phát triển kiểm tra và sửa lỗi trước khi tiếp tục với các hoạt động kiểm thử chi tiết.

Áp dụng khi nào?

Smoke Testing thường được thực hiện tại cấp độ System Testing. Đây là giai đoạn trong quy trình kiểm thử phần mềm khi các thành phần đã được tích hợp và kiểm tra tính tương thích giữa chúng; Nó giúp đảm bảo rằng phiên bản phần mềm có thể hoạt động ổn định ở mức độ cơ bản trước khi kiểm thử chi tiết hơn.

Các phương pháp KTPM

16. Kiểm thử xác nhận sửa lỗi (Confirmation Testing)

Khái niệm:

Kiểm thử xác nhận sửa lỗi (Confirmation Testing) là một phương pháp kiểm thử phần mềm, còn gọi là Re-testing, đây là quá trình kiểm tra lại các phần của phần mềm đã được sửa đổi hoặc cải thiện sau khi một lỗi đã được báo cáo và sửa đổi.

Ý nghĩa và mục đích:

- Mục đích chính của Confirmation Testing là xác nhận rằng các sửa đổi đã được thực hiện đúng cách và không gây ra các vấn đề mới trong phần mềm.
- Xác nhận sửa đổi.
- Đảm bảo tính ổn định.
- Giảm rủi ro.
- Xác nhận chất lượng.

Áp dụng khi nào?

Confirmation Testing thường được thực hiện ở mức độ System Testing.

Các phương pháp KTPM

17. Kiểm thử hồi quy (Regression Testing)

Khái niệm:

Kiểm thử hồi quy (Regression Testing) là một phương pháp kiểm thử phần mềm, là hoạt động kiểm thử phần mềm được thực hiện để đảm bảo rằng các thay đổi, sửa đổi hoặc bổ sung mới không ảnh hưởng đến các phần đã kiểm thử trước đó.

Ý nghĩa và mục đích:

- Đảm bảo tính ổn định và chất lượng của hệ thống sau khi thực hiện các thay đổi, đồng thời giảm thiểu rủi ro phát sinh từ các thay đổi.
- Regression Testing cũng giúp phát hiện và ngăn chặn sự trở lại của các lỗi đã được sửa đổi trước đó.

Áp dụng khi nào?

Regression Testing thường được thực hiện ở mức độ hệ thống hoặc cấp cao hơn, bao gồm cả Integration Testing và System Testing. Trong một số trường hợp, nó có thể được thực hiện ở cấp độ Unit Testing cho các đoạn mã quan trọng hoặc các module quan trọng. Đôi khi, Regression Testing cũng có thể được thực hiện ở cấp độ Acceptance Testing để đảm bảo rằng các thay đổi không ảnh hưởng đến yêu cầu của người dùng cuối.

Các phương pháp KTPM

18. Kiểm thử phi chức năng (Non-functional Testing)

Khái niệm:

Kiểm thử phi chức năng (Non-functional Testing) là một phương pháp kiểm thử phần mềm, là loại kiểm thử tập trung vào các yếu tố không liên quan đến chức năng của phần mềm mà thường tập trung vào các thuộc tính khác như hiệu suất, bảo mật, khả năng mở rộng, khả năng chịu tải, độ tin cậy và sự sẵn sàng.

Ý nghĩa và mục đích:

- Đảm bảo rằng phần mềm không chỉ hoạt động đúng về mặt chức năng mà còn đáp ứng được các yêu cầu không chức năng, đồng thời cải thiện trải nghiệm người dùng và đảm bảo hiệu suất ổn định của hệ thống trong mọi điều kiện.

Áp dụng khi nào?

Non-functional Testing thường được thực hiện ở mức độ System Testing và có thể cũng được thực hiện ở mức độ Integration Testing trong quá trình kiểm thử phần mềm.

Các phương pháp KTPM

19. Kiểm thử khả năng sử dụng (Usability Testing)

Khái niệm:

Kiểm thử khả năng sử dụng (Usability Testing) là một phương pháp kiểm thử phần mềm, là quá trình kiểm tra khả năng sử dụng và trải nghiệm người dùng của một sản phẩm hoặc ứng dụng.

Ý nghĩa và mục đích:

- Đảm bảo rằng sản phẩm phát triển có thể được sử dụng một cách hiệu quả và dễ dàng bởi người dùng cuối.
- Nâng cao trải nghiệm người dung.
- Giảm thiểu lỗi và sửa chữa sau này.
- Tăng khả năng chấp nhận từ người dung.
- Nâng cao hiệu suất và hiệu quả sử dụng.
- Tăng sự hài lòng của khách hàng.

Áp dụng khi nào?

Usability Testing thường được thực hiện ở mức độ System Testing hoặc (UAT). Trong quá trình này, sản phẩm hoặc ứng dụng sẽ được đưa ra cho người dùng cuối sử dụng và đánh giá để đảm bảo rằng giao diện người dùng (UI) và trải nghiệm người dùng (UX) đáp ứng được yêu cầu và mong đợi của họ. Điều này giúp đảm bảo rằng sản phẩm sẽ được chấp nhận và sử dụng một cách hiệu quả sau khi được triển khai.

Các phương pháp KTPM

20. Kiểm thử hiệu suất (Performance Testing)

Khái niệm:

Kiểm thử hiệu suất (Performance Testing) là một phương pháp kiểm thử phần mềm, là quá trình kiểm thử để đánh giá hiệu suất của một hệ thống hoặc ứng dụng phần mềm trong điều kiện cụ thể, như tải trọng cao hoặc số lượng người dùng lớn.

Ý nghĩa và mục đích:

Đảm bảo rằng hệ thống hoặc ứng dụng có thể hoạt động đáp ứng đủ và hiệu quả dưới tải trọng dự kiến mà nó sẽ phải đối mặt trong điều kiện thực tế.

Áp dụng khi nào?

Performance Testing thường được thực hiện ở mức độ System Testing hoặc Integration Testing. Trong quá trình này, các phần mềm hoặc hệ thống được kiểm tra để đảm bảo rằng chúng đáp ứng được yêu cầu về hiệu suất trong điều kiện tải trọng và áp lực cao. Đây là một phần quan trọng của quy trình kiểm thử phần mềm, giúp đảm bảo rằng ứng dụng hoặc hệ thống có thể hoạt động đáp ứng đủ và hiệu quả trong mọi tình huống.

Các phương pháp KTPM

21. Kiểm thử bảo mật (Security Testing)

Khái niệm:

Kiểm thử bảo mật (Security Testing) là một phương pháp kiểm thử phần mềm, là quá trình kiểm tra và đánh giá tính bảo mật của một hệ thống thông tin để xác định các lỗ hổng bảo mật và đảm bảo rằng hệ thống đáp ứng được các yêu cầu về bảo mật.

Ý nghĩa và mục đích:

- Bảo vệ thông tin quan trọng và ngăn chặn các cuộc tấn công từ các hacker, tin tặc hoặc người dùng không đáng tin cậy.
- Cung cấp sự an toàn và tin cậy cho hệ thống thông tin, dữ liệu và người dùng của nó, đồng thời giảm thiểu rủi ro và tổn thất liên quan đến việc xâm nhập, sự cố bảo mật và mất dữ liệu.

Áp dụng khi nào?

Security Testing thường được thực hiện ở mức độ System Testing và Acceptance Testing. Tuy nhiên, nó cũng có thể được tích hợp vào các mức độ kiểm thử khác như Integration Testing và Unit Testing, tùy thuộc vào yêu cầu cụ thể của dự án và quy trình kiểm thử được áp dụng.

Các phương pháp KTPM

22. Kiểm thử tuân thủ (Compliance Testing)

Khái niệm:

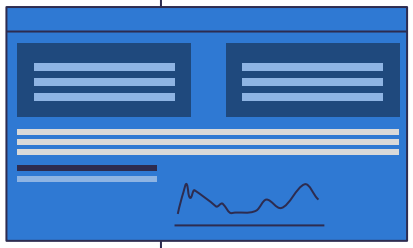
Kiểm thử tuân thủ (Compliance Testing) là một phương pháp kiểm thử phần mềm, là quá trình kiểm tra và đảm bảo rằng một hệ thống hoặc ứng dụng tuân thủ các quy định, chuẩn mực, và quy tắc liên quan.

Ý nghĩa và mục đích:

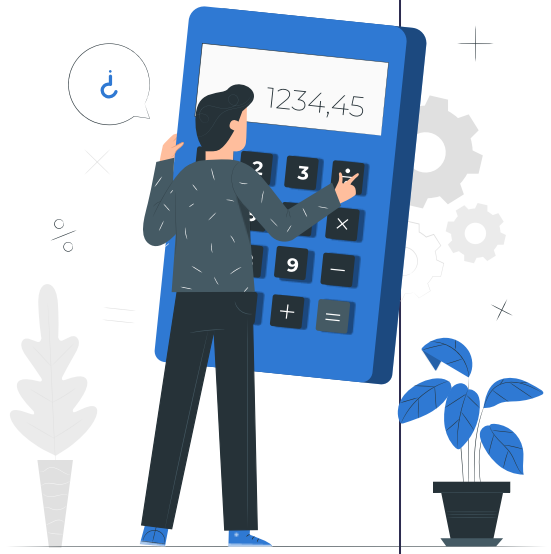
Đảm bảo rằng một ứng dụng hoặc hệ thống tuân thủ các quy định pháp lý, các chuẩn mực ngành nghề, và các yêu cầu bảo mật cụ thể.

Áp dụng khi nào?

Compliance Testing thường được thực hiện ở mức độ System Testing hoặc Acceptance Testing.



03



Các mục khác về Kiểm thử phần mềm

Các mục khác về KTPM

1. Các nhân tố trong kiểm thử	
2. Khiếm khuyết phần mềm	3. Số liệu đo lường
4. Tiêu chuẩn kiểm thử	5. Công cụ kiểm thử

1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

1. Kế hoạch kiểm thử (Test Plan):

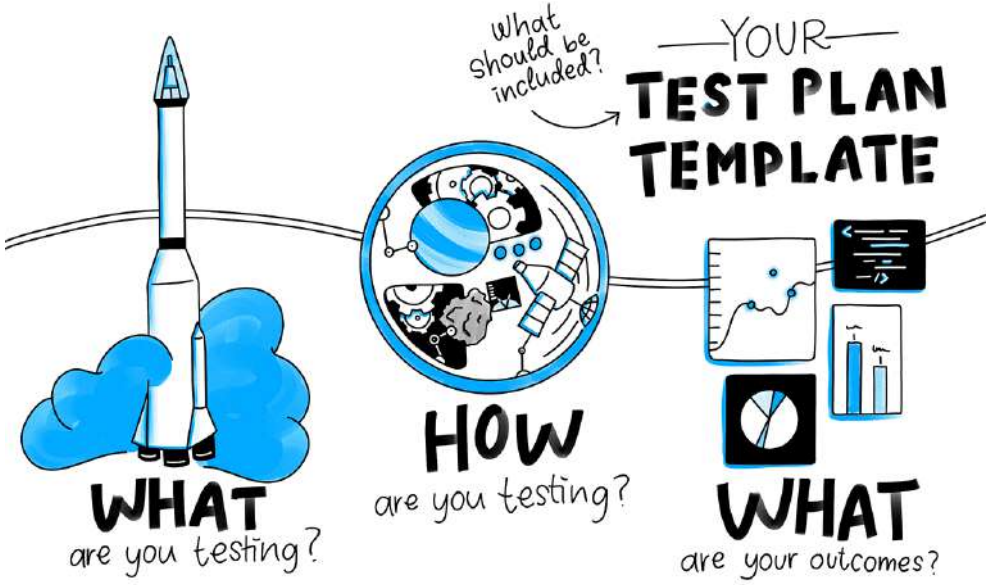
Khái niệm:

Kế hoạch kiểm thử (Test Plan) là một tài liệu chi tiết phác thảo chiến lược thử nghiệm, mục tiêu, nguồn lực cần thiết, lịch trình và tiêu chí thành công để thử nghiệm một tính năng hoặc phần mềm mới cụ thể.

Loại Test Plan:

Ứng với từng hệ thống, phần mềm và giai đoạn thì sẽ có những Test Plan tương ứng với hệ thống và giai đoạn đó, tùy thuộc vào yêu cầu và phạm vi của dự án phần mềm cụ thể.

Ví dụ như Kế hoạch kiểm thử hệ thống (System Test Plan); Kế hoạch kiểm thử chấp nhận (Acceptance Test Plan); Kế hoạch kiểm thử tích hợp (Integration Test Plan);...



1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

1. Kế hoạch kiểm thử (Test Plan):

Mẫu Test Plan:

1. Mục tiêu và Phạm vi: Mô tả mục tiêu và phạm vi của dự án kiểm thử.

2. Phương pháp kiểm thử: Xác định các kỹ thuật và phương pháp sẽ được sử dụng.

3. Lịch trình kiểm thử: Đặt ra lịch trình chi tiết cho các hoạt động kiểm thử, bao gồm các bước kiểm thử cụ thể và các thời hạn liên quan.

4. Nguyên tắc kiểm thử: Xác định các tiêu chí và quy tắc sẽ được áp dụng trong quá trình kiểm thử, bao gồm cách xác định lỗi và cách đánh giá hiệu suất kiểm thử.

5. Tài nguyên kiểm thử: Liệt kê tài nguyên con người, công cụ và môi trường cần thiết để thực hiện kiểm thử.

6. Rủi ro và Ưu tiên: Đánh giá các rủi ro tiềm ẩn và xác định các biện pháp giảm thiểu rủi ro.

7. Báo cáo và Thư mục kiểm thử: Mô tả cách thức báo cáo kết quả kiểm thử và quản lý các tài liệu kiểm thử.

1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

1. Kế hoạch kiểm thử (Test Plan):

Mẫu Test Plan:

Table of Contents

1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

2. Tình huống kiểm thử (Test Case):

Khái niệm:

Test Case được hiểu là tài liệu mô tả quá trình dữ liệu đầu vào (input), hành động (active) và kết quả (expected response) của một ứng dụng, phần mềm nào đó để xác thực mức độ chính xác. Mỗi test case mô tả một trường hợp kiểm thử cụ thể hoặc một loạt các bước để kiểm tra tính năng hoặc yêu cầu của phần mềm.

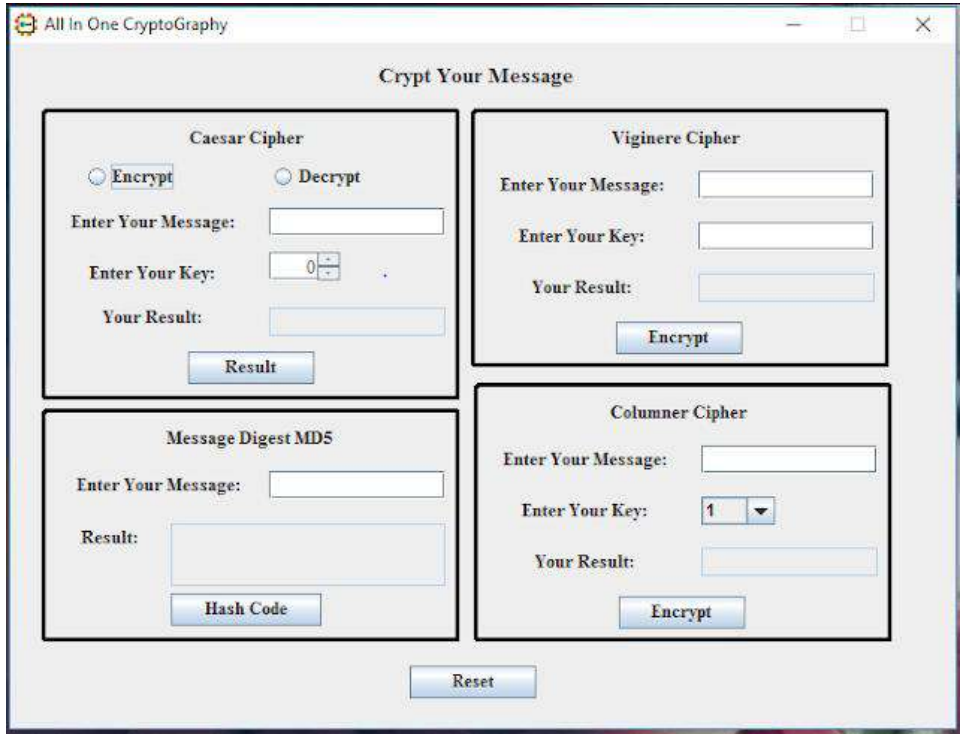
Thành phần:

- Mã Test Case (ID Test Case): Một mã số hoặc mã định danh duy nhất để xác định từng TC.
- Mục đích kiểm thử (Summary): Mô tả ngắn gọn về mục tiêu hoặc yêu cầu của TC.
- Dữ liệu thử nghiệm (Test Data): Dữ liệu hoặc điều kiện ban đầu cần thiết cho TC.
- Các bước thực hiện (Steps To Reproduce): Các bước cụ thể để thực hiện kiểm thử cho TC.
- Kết quả mong muốn (Expected Results): Kết quả mà bạn mong đợi từ việc thực hiện TC.
- Kết quả thực tế (Test Results): Kết quả thực tế sau khi thực hiện TC.

1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

2. Tình huống kiểm thử (Test Case):

Mẫu Test Case:



P = Plain Text | C = Cipher Text | K = Key

Test Case ID	Test Case Objective	Pre requisite	Steps	Input Data	Expected Output	Actual Output	Status
TC_01	Test Caesar Cipher Algorithm (For Encryption)	Textfield should be enabled	1. Select Encrypt Button 2. Enter Plain Text 3. Enter Numeric Key 4. Submit	P: hello K: 3	khoor	khoor	PASS
TC_02	Test Caesar Cipher Algorithm (For Decryption)	Textfield should be enabled	1. Select Decrypt Button 2. Enter Cipher Text 3. Enter Numeric Key 4. Submit	C: khoor K: 3	hello	hello	PASS
TC_03	Test Vignere Cipher	Text Fields should be enabled	1. Enter Plain Text 2. Enter String Key 3. Submit	P: hello K: abcds	hfnog	fhgon	FAIL
TC_04	Test MD5	Text Fields should be enabled	1. Enter Plain Text 2. Submit	T: hello	5d41402a bc4b2a76 b9719d91 1017c592	5d41402a bc4b2a76 b9719d91 1017c592	PASS
TC_05	Test Columner Cipher	Text Fields should be enabled	1. Enter Plain Text 2. Enter Numeric Key 3. Submit	P: hello K: 3	hleol	hleol	PASS

1. Các nhân tố trong kiểm thử (Artfacts Of Testing)

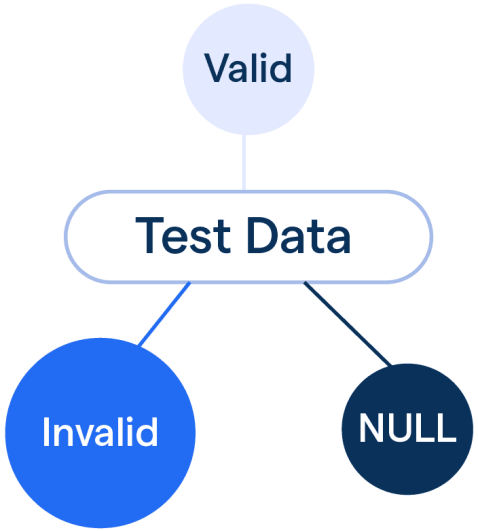
2. Dữ liệu kiểm thử (Test Data):

Khái niệm:

Dữ liệu kiểm thử (Test Data) là các dữ liệu được sử dụng để thực hiện kiểm thử trên phần mềm hoặc hệ thống. Dữ liệu kiểm thử đóng vai trò quan trọng trong việc kiểm tra tính đúng đắn và hoạt động của phần mềm.

Loại dữ liệu:

- Dữ liệu hợp lệ.
- Dữ liệu không hợp lệ.
- Dữ liệu Chữ/Số/Ký tự đặc biệt/Ngày tháng năm.
- Dữ liệu là Hình ảnh/Tệp/Âm thanh.



Chuẩn dữ liệu:

Chuẩn của dữ liệu phụ thuộc vào yêu cầu cụ thể của dự án và loại phần mềm được kiểm thử. Ví dụ như chuẩn định dạng email “example@email.com”, định dạng ngày DDMMYYYY,...

2. Khiếm khuyết phần mềm (Defects/Bugs)

1. Khuyết điểm (Defect):

Khái niệm:

Khuyết điểm (Defect) là một sai sót hoặc lỗi trong phần mềm mà gây ra sự không hoạt động, không đúng hoặc không thỏa mãn yêu cầu.

Một số loại khuyết điểm phổ biến:

- Lỗi Logic: Cách thức phần mềm được thiết kế/triển khai, dẫn đến hoạt động không đúng.
- Lỗi Giao Diện: Giao diện người dùng không hoạt động đúng cách, có thể là vấn đề về thiết kế hoặc triển khai.
- Lỗi Tích Hợp: Các phần của phần mềm không hoạt động cùng nhau như kỳ vọng.
- Lỗi Hiệu Năng: Tốc độ hoặc hiệu suất của phần mềm, ví dụ như thời gian phản hồi chậm.
- Lỗi Bảo Mật: Các lỗ hổng bảo mật có thể bị lợi dụng để xâm nhập/gây ra các vấn đề an ninh.
- Lỗi Dữ Liệu: PM không xử lý/lưu trữ dữ liệu đúng cách, dẫn đến mất mát/biến đổi dữ liệu.
- Lỗi Tài Liệu: Liên quan đến tài liệu hướng dẫn, như thông tin không chính xác/thiếu sót.

2. Khiếm khuyết phần mềm (Defects/Bugs)

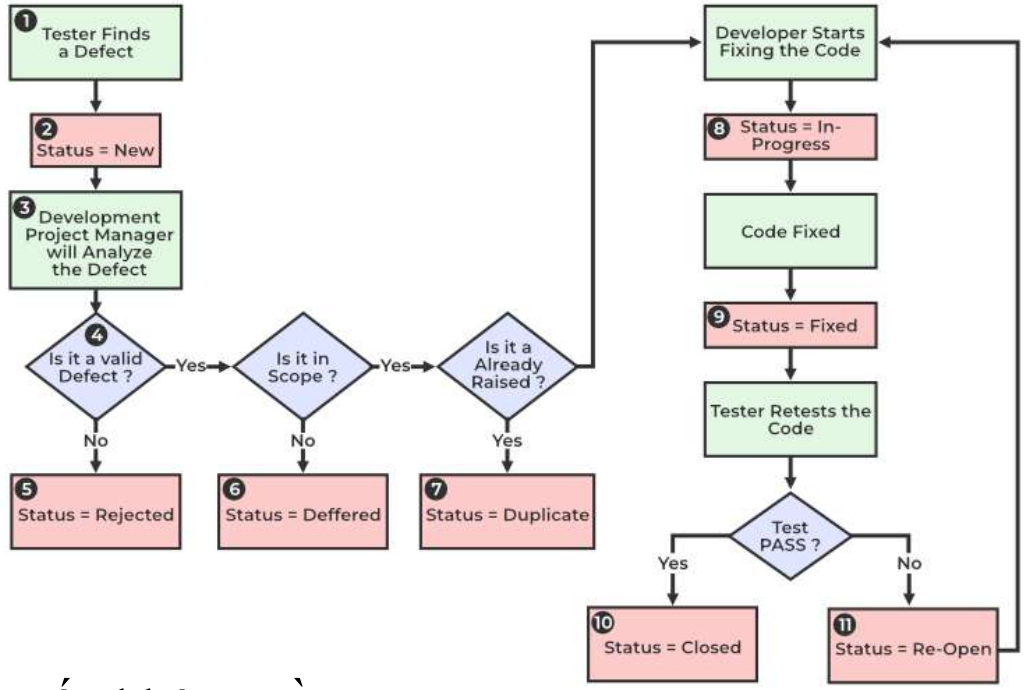
2. So sánh Lỗi-Khuyết điểm-Thất bại (Error-Defect-Failure):

Tên khiếm khuyết	Mô tả	Vai trò
Lỗi – Error	Là lỗi trong quá trình thiết kế hoặc mã lập trình.	Xuất phát từ quá trình phát triển phần mềm, thường là do con người.
Khuyết điểm – Defect	Là một sai sót hoặc lỗi trong phần mềm, dẫn đến hoạt động không đúng hoặc không thỏa mãn yêu cầu.	Xuất phát từ Error và được phát hiện trong quá trình kiểm thử phần mềm.
Thất bại – Failure	Là việc phần mềm không hoạt động như kỳ vọng, dẫn đến sự cố hoặc không thể sử dụng.	Xuất phát từ Defect và được phát hiện khi phần mềm đã triển khai và sử dụng.

2. Khiếm khuyết phần mềm (Defects/Bugs)

3. Defect/Bug Life Cycle:

Vòng đời Defect/Bug:



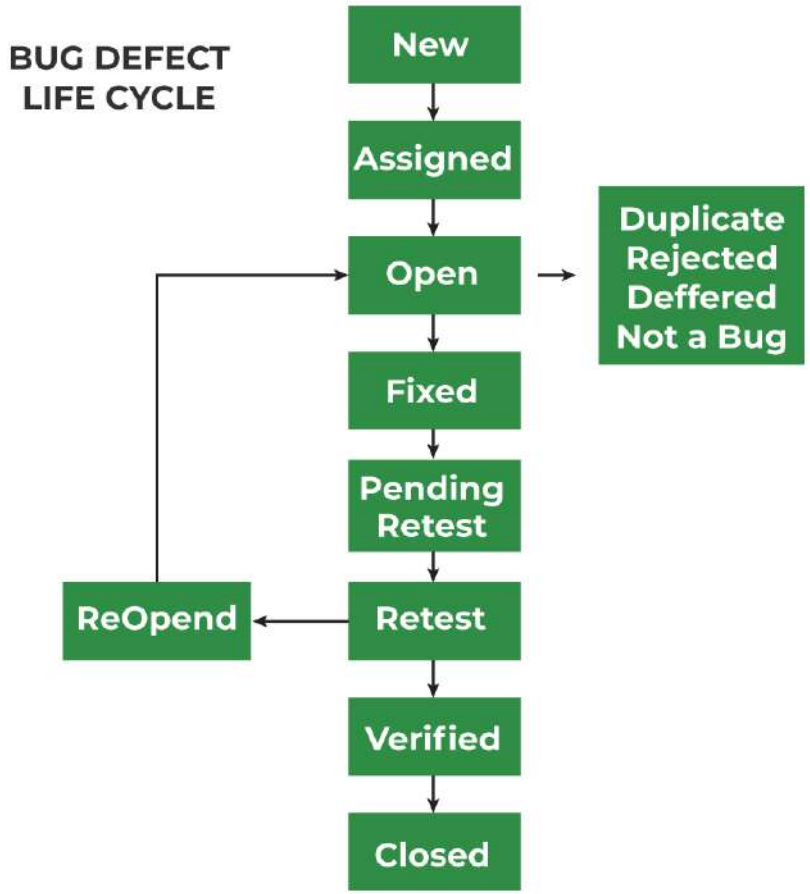
1. Nhà kiểm thử tìm ra một Defect.
2. Trạng thái Defect chuyển thành "Mới".
3. Quản lý dự án phát triển sẽ phân tích Defect.
4. Defect có hợp lệ không?
5. Nếu không hợp lệ thì trạng thái chuyển "Bị từ chối".
6. Nếu hợp lệ thì nó có nằm trong Scope không? Nếu không nằm trong Scope thì trạng thái chuyển "Hoãn lại".
7. Nếu nằm trong Scope thì nó có sẵn sàng thực hiện không? Nếu có thì trạng thái chuyển "".
9. Nếu không thì

2. Khiếm khuyết phần mềm (Defects/Bugs)

3. Defect/Bug Life Cycle:

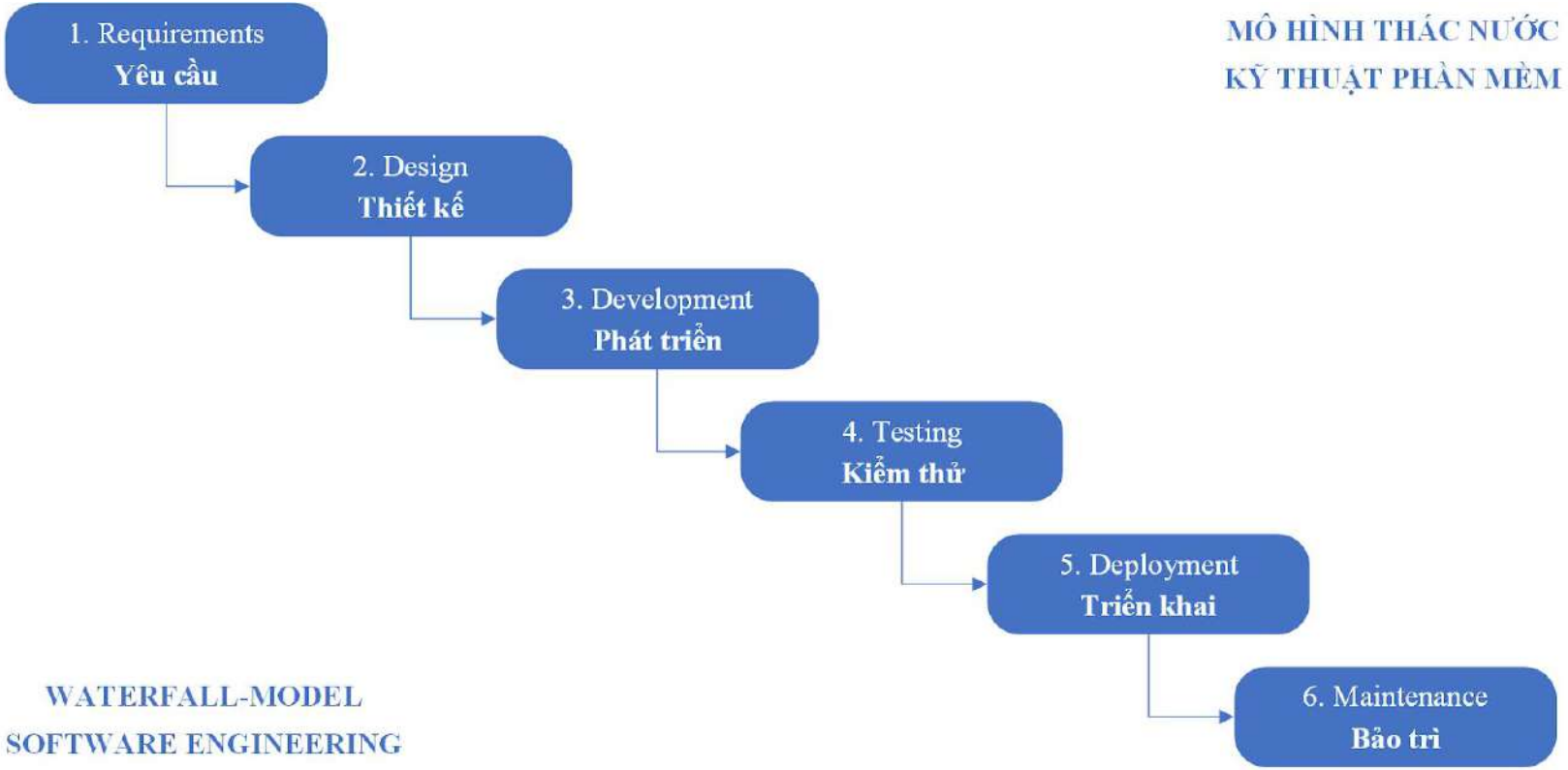
Defect/Bus Status:

1. New
2. Assigned
3. Open, Duplicate Rejected Deffered Not a Bug
4. Fixed
5. Pending Retest
6. Retest
7. ReOpend
8. Verified.
9. Closed



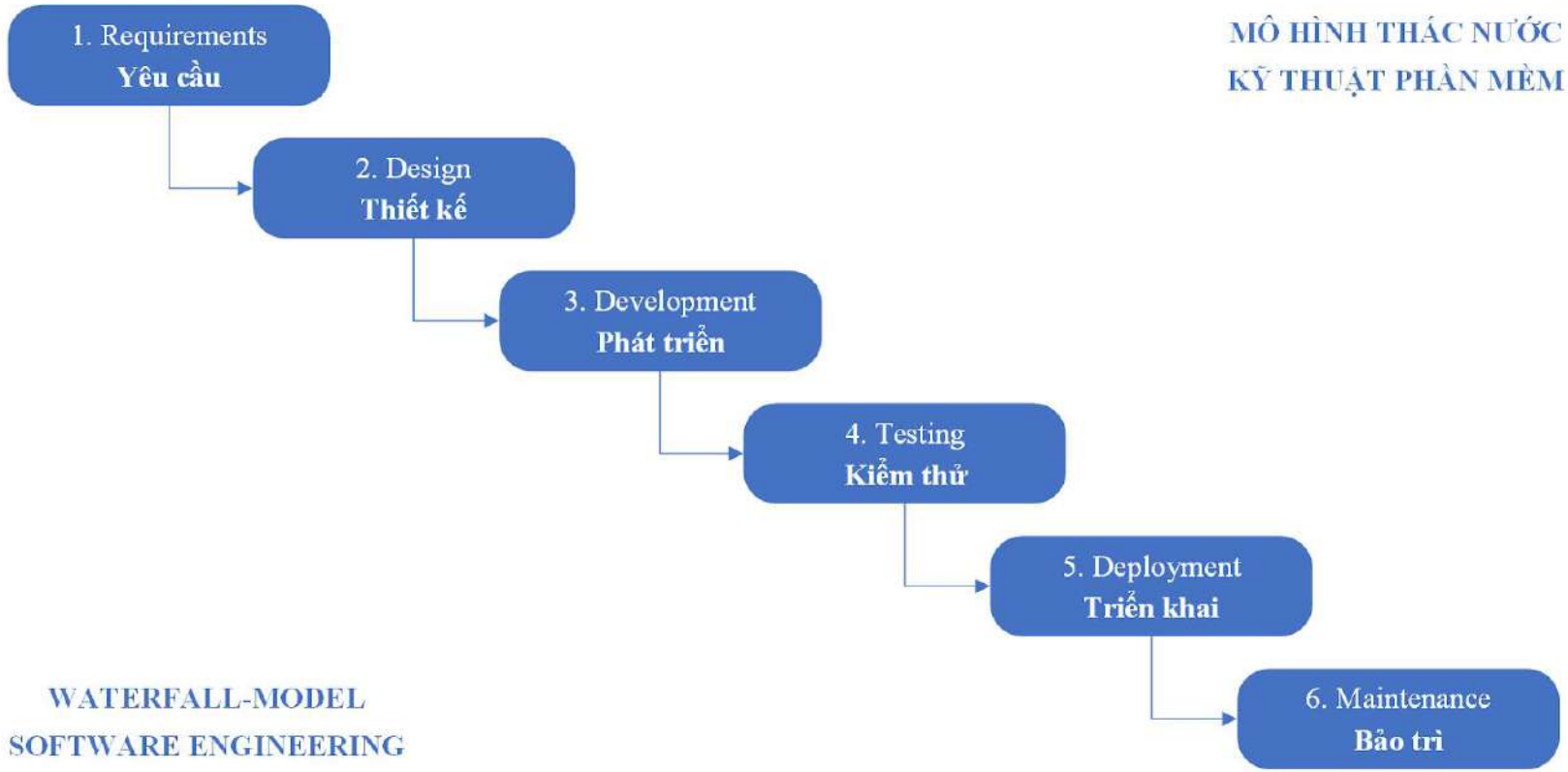
3. Số liệu đo lường (Testing Metrics)

Khái niệm: Mô hình thác nước (Waterfall model) là một trong những mô hình phát triển phần mềm cổ điển nhất và đơn giản nhất trong các mô hình vòng đời phát triển phần mềm. Mô hình này được tổ chức theo kiểu tuyến tính, trong đó các giai đoạn phát triển liên tục chảy từ trên xuống dưới như một dãy thác nước.



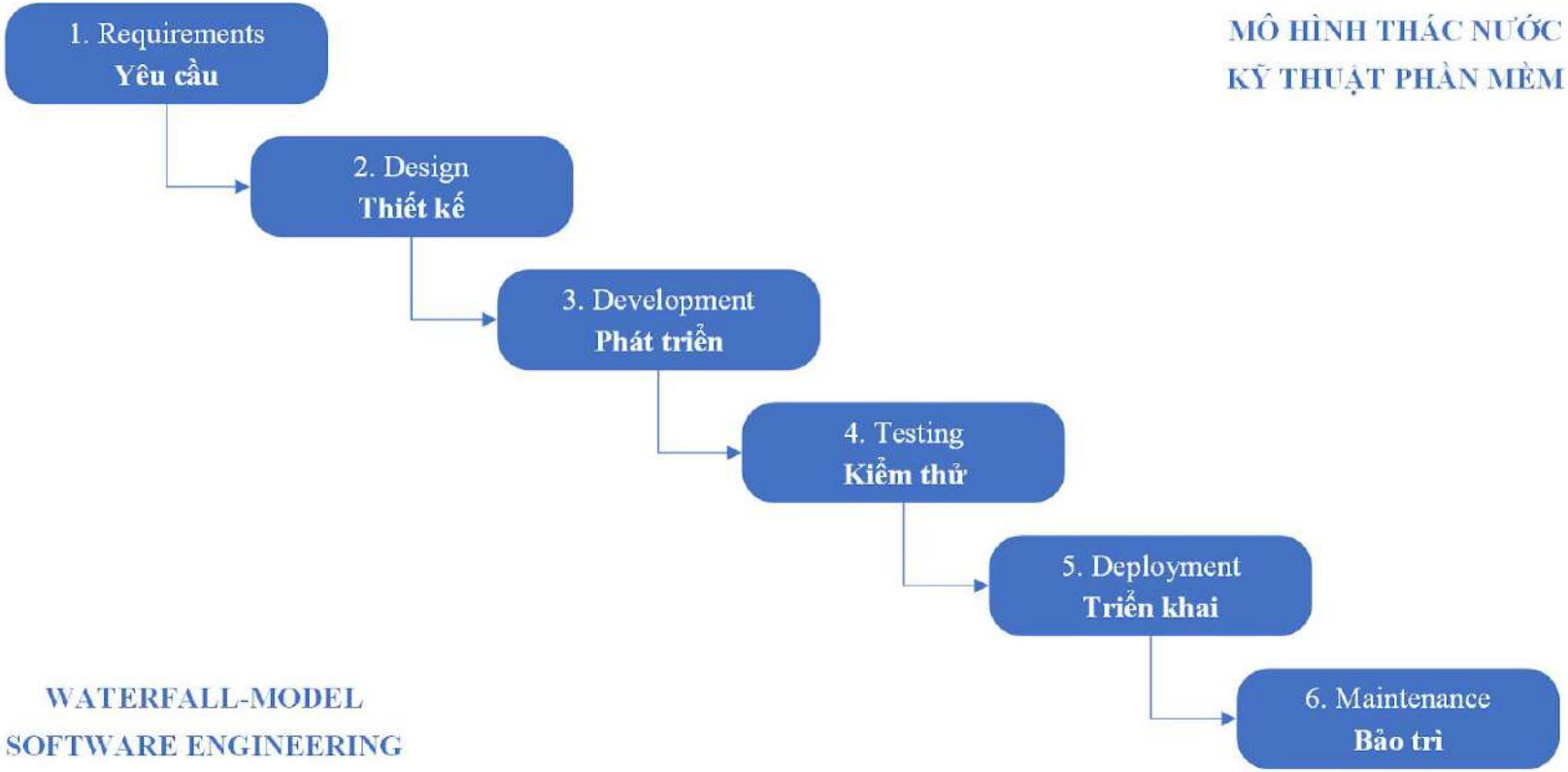
4. Tiêu chuẩn kiểm thử (Testing Standard)

Khái niệm: Mô hình thác nước (Waterfall model) là một trong những mô hình phát triển phần mềm cổ điển nhất và đơn giản nhất trong các mô hình vòng đời phát triển phần mềm. Mô hình này được tổ chức theo kiểu tuyến tính, trong đó các giai đoạn phát triển liên tục chảy từ trên xuống dưới như một dãy thác nước.



5. Công cụ kiểm thử (Testing Tools)

Khái niệm: Mô hình thác nước (Waterfall model) là một trong những mô hình phát triển phần mềm cổ điển nhất và đơn giản nhất trong các mô hình vòng đời phát triển phần mềm. Mô hình này được tổ chức theo kiểu tuyến tính, trong đó các giai đoạn phát triển liên tục chảy từ trên xuống dưới như một dãy thác nước.



BATIZENS

Trân trọng cảm ơn!