**Software Testing**

**Kiểm tra chương trình**

Thử nghiệm được thiết kế để cho thấy rằng một chương trình thực hiện những gì nó được dự định để làm và để khám phá các khuyết tật của chương trình trước khi nó được đưa vào sử dụng.

Khi bạn kiểm tra phần mềm, bạn thực thi một chương trình bằng cách sử dụng dữ liệu nhân tạo.

Bạn kiểm tra kết quả chạy thử để tìm lỗi, bất thường hoặc thông tin về các thuộc tính không hoạt động của chương trình.

Có thể tiết lộ sự hiện diện của các lỗi **KHÔNG** vắng mặt của họ.

Kiểm tra là một phần của quá trình xác minh và xác thực tổng quát hơn, bao gồm các kỹ thuật xác thực tĩnh.

**Mục tiêu kiểm tra chương trình**

Để minh họa cho nhà phát triển và khách hàng rằng phần mềm đáp ứng các yêu cầu của họ.

*Đối với phần mềm tùy chỉnh, điều này có nghĩa là phải có ít nhất một kiểm tra cho mọi yêu cầu trong tài liệu yêu cầu. Đối với các sản phẩm phần mềm chung, có nghĩa là phải có các thử nghiệm cho tất cả các tính năng của hệ thống, cộng với sự kết hợp các tính năng này, sẽ được đưa vào bản phát hành sản phẩm.*

Để khám phá các tình huống trong đó hành vi của phần mềm không chính xác, không mong muốn hoặc không phù hợp với đặc điểm kỹ thuật của phần mềm.

*Kiểm tra khiếm khuyết có liên quan đến việc tạo ra các hành vi hệ thống không mong muốn như sự cố hệ thống, các tương tác không mong muốn với các hệ thống khác, tính toán sai và tham nhũng dữ liệu.*

**Kiểm tra xác thực và lỗi**

Mục tiêu đầu tiên dẫn đến kiểm tra xác thực

*Bạn mong đợi hệ thống hoạt động chính xác bằng cách sử dụng một tập hợp các trường hợp thử nghiệm nhất định phản ánh việc sử dụng dự kiến của hệ thống.*

Mục tiêu thứ hai dẫn đến kiểm tra lỗi

*Các trường hợp thử nghiệm được thiết kế để lộ các khuyết tật. Các trường hợp thử nghiệm trong kiểm tra lỗi có thể được cố ý che khuất và không cần phải phản ánh cách hệ thống được sử dụng bình thường.*

**Mục tiêu quy trình thử nghiệm**

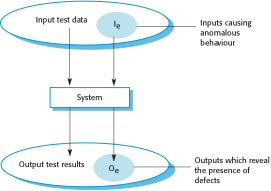
**Kiểm tra xác thực**

* Để minh họa cho nhà phát triển và khách hàng hệ thống rằng phần mềm đáp ứng các yêu cầu của họ
* Một thử nghiệm thành công cho thấy rằng hệ thống hoạt động như dự định.

**Kiểm tra lỗi**

* Để phát hiện lỗi hoặc lỗi trong phần mềm mà hành vi của nó không chính xác hoặc không phù hợp với đặc điểm kỹ thuật của nó
* Một thử nghiệm thành công là một thử nghiệm mà làm cho hệ thống hoạt động không chính xác và để lộ ra một khiếm khuyết trong hệ thống.

**Một mô hình đầu vào-đầu ra của kiểm thử chương trình**



**Xác minh và xác thực**

Xác minh: "Chúng tôi đang xây dựng sản phẩm đúng".

* Phần mềm phải phù hợp với đặc điểm kỹ thuật của nó.

Xác nhận : "Chúng ta đang xây dựng đúng sản phẩm".

* Phần mềm phải làm những gì người dùng thực sự yêu cầu.

**Tự tin V & V**

Mục tiêu của V & V là thiết lập sự tự tin rằng hệ thống là 'phù hợp với mục đích'.

Phụ thuộc vào mục đích của hệ thống, kỳ vọng của người dùng và môi trường tiếp thị

**Mục đích phần mềm**

* Mức độ tin cậy phụ thuộc vào mức độ quan trọng của phần mềm đối với một tổ chức.

**Kỳ vọng của người dùng**

* Người dùng có thể có kỳ vọng thấp của một số loại phần mềm nhất định.

**Môi trường tiếp thị**

* Đưa sản phẩm ra thị trường sớm có thể quan trọng hơn việc tìm ra khuyết tật trong chương trình.

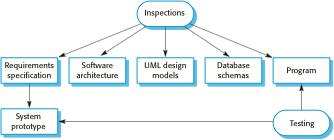
**Inspections & Testing**

Software inspections: Quan tâm đến việc phân tích biểu diễn hệ thống tĩnh để phát hiện sự cố (xác minh tĩnh)

* Có thể được bổ sung bằng phân tích mã và tài liệu dựa trên công cụ.

Software testing: Quan tâm đến việc thực hiện và quan sát hành vi của sản phẩm (xác minh động)

* Hệ thống được thực hiện với dữ liệu thử nghiệm và hành vi hoạt động của nó được quan sát.



**Software Inspections**

Những người này liên quan đến việc kiểm tra sự biểu diễn nguồn với mục đích phát hiện dị thường và khuyết tật.

Việc kiểm tra không yêu cầu thực thi hệ thống để có thể được sử dụng trước khi triển khai.

Chúng có thể được áp dụng cho bất kỳ đại diện nào của hệ thống (yêu cầu, thiết kế, dữ liệu cấu hình, dữ liệu thử nghiệm, v.v.).

Chúng đã được chứng minh là một kỹ thuật hiệu quả để phát hiện lỗi chương trình.

**Ưu điểm của kiểm tra**

Trong quá trình thử nghiệm, lỗi có thể che giấu (ẩn) các lỗi khác. Vì kiểm tra là một quá trình tĩnh nên bạn không phải lo lắng về tương tác giữa các lỗi.

Các phiên bản không đầy đủ của một hệ thống có thể được kiểm tra mà không phải trả thêm chi phí. Nếu một chương trình không đầy đủ, thì bạn cần phát triển các bộ kiểm tra chuyên dụng để kiểm tra các bộ phận có sẵn.

Cũng như tìm kiếm các khiếm khuyết của chương trình, việc kiểm tra cũng có thể xem xét các thuộc tính chất lượng rộng hơn của một chương trình, chẳng hạn như tuân thủ các tiêu chuẩn, tính di động và khả năng bảo trì.

**Inspections and Testing**

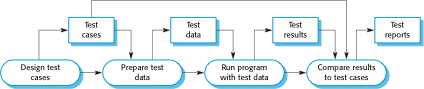
Kiểm tra và thử nghiệm là bổ sung và không phản đối các kỹ thuật xác minh.

Cả hai nên được sử dụng trong quá trình V & V.

Kiểm tra có thể kiểm tra sự phù hợp với đặc điểm kỹ thuật nhưng không tuân thủ các yêu cầu thực tế của khách hàng.

Kiểm tra không thể kiểm tra các đặc điểm phi chức năng như hiệu suất, khả năng sử dụng, v.v.

**A model of the software testing process**

****

**Các giai đoạn thử nghiệm**

Thử nghiệm phát triển, nơi hệ thống được kiểm tra trong quá trình phát triển để phát hiện lỗi và lỗi.

Phát hành thử nghiệm, trong đó một nhóm thử nghiệm riêng biệt kiểm tra một phiên bản hoàn chỉnh của hệ thống trước khi nó được phát hành cho người dùng.

Kiểm tra người dùng, nơi người dùng hoặc người dùng tiềm năng của hệ thống kiểm tra hệ thống trong môi trường của riêng họ.

**Thử nghiệm phát triển**

Thử nghiệm phát triển bao gồm tất cả các hoạt động thử nghiệm được thực hiện bởi nhóm phát triển hệ thống.

* Kiểm tra đơn vị, nơi các đơn vị chương trình riêng lẻ hoặc các lớp đối tượng được kiểm tra. Thử nghiệm đơn vị nên tập trung vào việc kiểm tra chức năng của các đối tượng hoặc phương pháp.
* Kiểm tra thành phần, trong đó một số đơn vị riêng lẻ được tích hợp để tạo thành phần hỗn hợp. Kiểm thử thành phần nên tập trung vào việc kiểm tra các giao diện thành phần.
* Kiểm tra hệ thống, trong đó một số hoặc tất cả các thành phần trong hệ thống được tích hợp và hệ thống được kiểm tra toàn bộ. Kiểm tra hệ thống nên tập trung vào thử nghiệm các tương tác thành phần.

**Unit testing**

Thử nghiệm đơn vị là quá trình thử nghiệm từng thành phần riêng biệt.

Đó là một quá trình kiểm tra lỗi.

Các đơn vị có thể là:

* Các hàm hoặc phương thức riêng lẻ trong một đối tượng
* Các lớp đối tượng với một số thuộc tính và phương thức
* Các thành phần tổng hợp với các giao diện được xác định được sử dụng để truy cập chức năng của chúng.

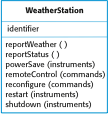
**Kiểm tra lớp đối tượng**

Phạm vi kiểm tra hoàn chỉnh của một lớp học liên quan đến

* Kiểm tra tất cả các hoạt động liên kết với một đối tượng
* Thiết lập và thẩm vấn tất cả các thuộc tính đối tượng
* Tập thể dục đối tượng ở tất cả các trạng thái có thể.

Thừa kế làm cho việc kiểm tra lớp đối tượng khó khăn hơn vì thông tin được kiểm tra không được bản địa hóa.

**Giao diện đối tượng trạm thời tiết**

****

**Kiểm tra trạm thời tiết**

Cần xác định các trường hợp kiểm thử cho reportWeather, hiệu chỉnh, kiểm tra, khởi động và tắt máy.

Sử dụng một mô hình nhà nước, xác định trình tự của các quá trình chuyển đổi trạng thái được kiểm tra và các chuỗi sự kiện gây ra các quá trình chuyển đổi này

Ví dụ:

* Shutdown -> Running-> Shutdown
* Configuring-> Running-> Testing -> Transmitting -> Running
* Running-> Collecting-> Running-> Summarizing -> Transmitting -> Running

**Automated Testing**

Bất cứ khi nào có thể, kiểm tra đơn vị phải được tự động hóa để kiểm tra được chạy và kiểm tra mà không cần sự can thiệp thủ công.

Trong thử nghiệm đơn vị tự động, bạn sử dụng một khung công tác tự động hóa thử nghiệm (như JUnit) để viết và chạy các thử nghiệm chương trình của bạn.

Các khung kiểm thử đơn vị cung cấp các lớp thử nghiệm chung mà bạn mở rộng để tạo các trường hợp thử nghiệm cụ thể. Sau đó, họ có thể chạy tất cả các thử nghiệm mà bạn đã triển khai và báo cáo, thường xuyên thông qua một số GUI, về sự thành công của các thử nghiệm khác.

**Automated Test Components**

Một phần thiết lập, nơi bạn khởi tạo hệ thống với trường hợp thử nghiệm, cụ thể là các đầu vào và đầu ra dự kiến.

Một phần cuộc gọi, nơi bạn gọi đối tượng hoặc phương pháp được kiểm tra.

Một phần khẳng định nơi bạn so sánh kết quả của cuộc gọi với kết quả mong đợi. Nếu xác nhận đánh giá là đúng, thử nghiệm đã thành công nếu sai, sau đó nó đã thất bại.

**Đơn vị kiểm tra hiệu quả**

Các trường hợp thử nghiệm sẽ cho thấy rằng, khi được sử dụng như mong đợi, thành phần mà bạn đang thử nghiệm thực hiện những gì nó được cho là phải làm.

Nếu có lỗi trong thành phần, chúng sẽ được tiết lộ bởi các trường hợp thử nghiệm.

Điều này dẫn đến 2 loại trường hợp kiểm thử đơn vị:

* Việc đầu tiên trong số này nên phản ánh hoạt động bình thường của một chương trình và nên cho thấy rằng các thành phần hoạt động như mong đợi.
* Loại trường hợp thử nghiệm khác phải dựa trên trải nghiệm thử nghiệm về những vấn đề phổ biến phát sinh. Nó nên sử dụng đầu vào bất thường để kiểm tra xem chúng được xử lý đúng cách và không làm hỏng thành phần.

**Chiến lược thử nghiệm**

Thử nghiệm phân vùng, nơi bạn xác định các nhóm đầu vào có đặc điểm chung và cần được xử lý theo cùng một cách.

* Bạn nên chọn các bài kiểm tra từ trong mỗi nhóm này.

Kiểm tra dựa trên hướng dẫn, nơi bạn sử dụng nguyên tắc kiểm tra để chọn các trường hợp kiểm tra.

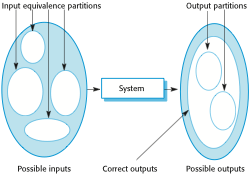
* Những hướng dẫn này phản ánh kinh nghiệm trước đây về các loại lỗi mà các lập trình viên thường thực hiện khi phát triển các thành phần.

**Kiểm tra phân vùng**

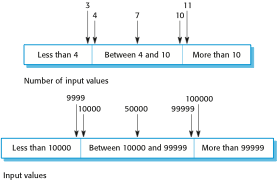
Dữ liệu đầu vào và kết quả đầu ra thường rơi vào các lớp khác nhau, nơi tất cả các thành viên của một lớp có liên quan.

Mỗi lớp này là một ***phân vùng tương đương*** hoặc miền nơi chương trình hoạt động theo cách tương đương cho mỗi thành viên của lớp.

Các trường hợp kiểm tra nên được chọn từ mỗi phân vùng.

**Equivalence Partitioning** 

**Equivalence Partitions**

****

**Nguyên tắc kiểm tra (chuỗi)**

* Kiểm tra phần mềm với các chuỗi chỉ có một giá trị duy nhất.
* Sử dụng chuỗi các kích thước khác nhau trong các thử nghiệm khác nhau.
* Kiểm tra nguồn gốc để các phần tử đầu tiên, trung bình và cuối cùng của chuỗi được truy cập.
* Thử nghiệm với các chuỗi có độ dài bằng không.

**Nguyên tắc kiểm tra chung**

* Chọn đầu vào buộc hệ thống tạo tất cả thông báo lỗi
* Thiết kế đầu vào gây ra bộ đệm đầu vào tràn
* Lặp lại cùng một đầu vào hoặc một loạt các đầu vào nhiều lần
* Buộc tạo ra các đầu ra không hợp lệ
* Lực lượng kết quả tính toán quá lớn hoặc quá nhỏ.

**Key points**

Thử nghiệm chỉ có thể hiển thị sự hiện diện của các lỗi trong một chương trình. Nó không thể chứng minh rằng không có lỗi còn lại.

Kiểm thử phát triển là trách nhiệm của nhóm phát triển phần mềm. Một nhóm riêng biệt phải chịu trách nhiệm thử nghiệm một hệ thống trước khi nó được phát hành cho khách hàng.

Thử nghiệm phát triển bao gồm kiểm tra đơn vị, trong đó bạn kiểm tra từng đối tượng và phương pháp thử nghiệm thành phần trong đó bạn kiểm tra các nhóm đối tượng liên quan và kiểm tra hệ thống, trong đó bạn thử nghiệm một phần hoặc toàn bộ hệ thống.

**Kiểm tra thành phần**

Các thành phần phần mềm thường là các thành phần tổng hợp được tạo thành từ một số đối tượng tương tác.

* Ví dụ, trong hệ thống trạm thời tiết, thành phần cấu hình lại bao gồm các đối tượng xử lý từng khía cạnh của cấu hình lại.

Bạn truy cập chức năng của các đối tượng này thông qua giao diện thành phần được xác định.

Do đó, thử nghiệm các thành phần tổng hợp nên tập trung vào việc hiển thị rằng giao diện thành phần hoạt động theo đặc điểm kỹ thuật của nó.

* Bạn có thể giả định rằng các bài kiểm tra đơn vị trên các đối tượng riêng lẻ trong thành phần đã được hoàn thành.

**Kiểm tra giao diện**

Mục tiêu là để phát hiện lỗi do lỗi giao diện hoặc giả định không hợp lệ về giao diện.

Loại giao diện

* ***Giao diện tham số*** Dữ liệu truyền từ một phương thức hoặc thủ tục này sang phương thức khác.
* ***Giao diện bộ nhớ*** chia sẻ Khối bộ nhớ được chia sẻ giữa các thủ tục hoặc chức năng.
* ***Các giao diện thủ tục*** Hệ thống con gói gọn một tập hợp các thủ tục được gọi bởi các hệ thống con khác.
* ***Các giao diện truyền tin nhắn*** Các hệ thống con yêu cầu các dịch vụ từ các hệ thống con khác

**Lỗi giao diện**

**Giao diện sử dụng sai**

* Thành phần gọi điện gọi một thành phần khác và gây ra lỗi khi sử dụng giao diện của nó, ví dụ: thông số theo thứ tự sai.

**Giao diện hiểu lầm**

* Một thành phần gọi điện thoại nhúng các giả định về hành vi của thành phần được gọi là không chính xác.

**Lỗi thời gian**

* Thành phần gọi và gọi hoạt động ở các tốc độ khác nhau và thông tin lỗi thời được truy cập.

**Nguyên tắc kiểm tra giao diện**

Thiết kế kiểm tra để các tham số cho một thủ tục được gọi là ở các đầu cực của phạm vi của chúng.

Luôn kiểm tra các tham số con trỏ với con trỏ null.

Kiểm tra thiết kế khiến cho thành phần thất bại.

Sử dụng thử nghiệm ứng suất trong các hệ thống truyền tin nhắn.

Trong các hệ thống bộ nhớ dùng chung, thay đổi thứ tự các thành phần được kích hoạt.

**Thử nghiệm hệ thống**

Kiểm tra hệ thống trong quá trình phát triển bao gồm tích hợp các thành phần để tạo ra một phiên bản của hệ thống và sau đó kiểm tra hệ thống tích hợp.

Trọng tâm trong thử nghiệm hệ thống là kiểm tra tương tác giữa các thành phần.

Kiểm tra hệ thống kiểm tra xem các thành phần có tương thích hay không, tương tác một cách chính xác và chuyển đúng dữ liệu vào đúng thời điểm trên các giao diện của chúng.

Kiểm tra hệ thống kiểm tra hành vi nổi lên của một hệ thống.

**Kiểm tra hệ thống và thành phần**

Trong quá trình kiểm tra hệ thống, các thành phần có thể tái sử dụng đã được phát triển riêng biệt và các hệ thống có thể được tích hợp với các thành phần mới được phát triển. Hệ thống hoàn chỉnh sau đó được kiểm tra.

Các thành phần được phát triển bởi các thành viên nhóm khác nhau hoặc các nhóm phụ có thể được tích hợp ở giai đoạn này. Kiểm tra hệ thống là một tập thể chứ không phải là một quy trình riêng lẻ.

* *Ở một số công ty, việc kiểm tra hệ thống có thể bao gồm một nhóm thử nghiệm riêng biệt mà không có sự tham gia của các nhà thiết kế và lập trình viên.*

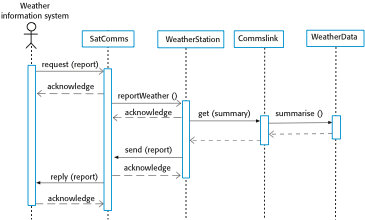
**Thử nghiệm sử dụng**

Các trường hợp sử dụng được phát triển để xác định các tương tác hệ thống có thể được sử dụng làm cơ sở cho việc kiểm tra hệ thống.

Mỗi trường hợp sử dụng thường liên quan đến một số thành phần hệ thống để kiểm tra trường hợp sử dụng buộc các tương tác này xảy ra.

Các biểu đồ trình tự liên kết với ca sử dụng ghi lại các thành phần và các tương tác đang được thử nghiệm.

**Thu thập biểu đồ trình tự dữ liệu thời tiết**



**Chính sách thử nghiệm**

Thử nghiệm hệ thống thông gió là không thể vì vậy các chính sách thử nghiệm xác định phạm vi kiểm tra hệ thống yêu cầu có thể được phát triển.

Ví dụ về chính sách thử nghiệm:

* Tất cả các chức năng hệ thống được truy cập thông qua các menu sẽ được kiểm tra.
* Việc kết hợp các hàm (ví dụ: định dạng văn bản) được truy cập thông qua cùng một menu phải được kiểm tra.
* Trong trường hợp đầu vào của người dùng được cung cấp, tất cả các hàm phải được kiểm tra với cả đầu vào chính xác và không chính xác.

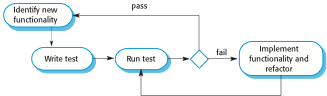
**Hướng phát triển thử nghiệm**

Phát triển theo hướng thử nghiệm (TDD) là một cách tiếp cận để phát triển chương trình, trong đó bạn liên tục kiểm tra và phát triển mã.

Các bài kiểm tra được viết trước khi viết mã và ‘qua’ các bài kiểm tra là trình điều khiển phát triển quan trọng.

Bạn phát triển mã gia tăng, cùng với một thử nghiệm cho số gia tăng đó. Bạn không chuyển sang bước tăng tiếp theo cho đến khi mã mà bạn đã phát triển vượt qua thử nghiệm của nó.

TDD được giới thiệu như một phần của các phương thức nhanh như Lập trình cực đoan. Tuy nhiên, nó cũng có thể được sử dụng trong các quy trình phát triển theo kế hoạch.



**TDD Process Các hoạt động**

Bắt đầu bằng cách xác định sự gia tăng của chức năng được yêu cầu. Điều này thường sẽ nhỏ và có thể triển khai trong một vài dòng mã.

Viết một bài kiểm tra cho chức năng này và thực hiện điều này như một bài kiểm tra tự động.

Chạy thử nghiệm, cùng với tất cả các thử nghiệm khác đã được triển khai. Ban đầu, bạn chưa triển khai chức năng để thử nghiệm mới sẽ thất bại.

Thực hiện chức năng và chạy lại thử nghiệm.

Khi tất cả các thử nghiệm chạy thành công, bạn chuyển sang triển khai phần tiếp theo của chức năng.

**Lợi ích của phát triển theo hướng thử nghiệm**

**Mã số bảo hiểm**

* Mỗi phân đoạn mã mà bạn viết có ít nhất một thử nghiệm được liên kết để tất cả các mã được viết có ít nhất một thử nghiệm.

**Kiểm tra hồi quy**

* Một bộ kiểm thử hồi quy được phát triển dần dần khi một chương trình được phát triển.

**Gỡ lỗi đơn giản**

* Khi một thử nghiệm thất bại, nó sẽ được rõ ràng, nơi vấn đề nằm. Mã mới được viết cần được kiểm tra và sửa đổi.

**Tài liệu hệ thống**

* Bản thân các bài kiểm tra là một dạng tài liệu mô tả mã nên làm gì.

**Kiểm tra hồi quy**

Kiểm tra hồi quy đang kiểm tra hệ thống để kiểm tra xem các thay đổi có không phải là 'làm hỏng' mã hoạt động trước đó hay không.

Trong quy trình kiểm tra thủ công, kiểm thử hồi quy là tốn kém nhưng, với thử nghiệm tự động, nó đơn giản và dễ hiểu. Tất cả các bài kiểm tra đều chạy lại mỗi lần thay đổi được thực hiện cho chương trình.

Các thử nghiệm phải chạy ‘thành công’ trước khi thay đổi được thực hiện.

**Kiểm tra phát hành**

Kiểm tra phát hành là quá trình thử nghiệm một bản phát hành cụ thể của một hệ thống được thiết kế để sử dụng bên ngoài nhóm phát triển.

Mục tiêu chính của quá trình thử nghiệm phát hành là thuyết phục nhà cung cấp của hệ thống rằng nó đủ tốt để sử dụng.

* *Do đó, việc thử nghiệm phát hành phải cho thấy rằng hệ thống cung cấp chức năng, hiệu suất và độ tin cậy được chỉ định của nó và rằng nó không bị lỗi trong quá trình sử dụng bình thường.*

Phát hành thử nghiệm thường là một quá trình thử nghiệm hộp đen, nơi các thử nghiệm chỉ được lấy từ đặc tả hệ thống.

**Phát hành thử nghiệm và kiểm tra hệ thống**

Kiểm tra phát hành là một dạng thử nghiệm hệ thống.

Sự khác biệt quan trọng:

* *Một nhóm riêng biệt không tham gia vào việc phát triển hệ thống, phải chịu trách nhiệm kiểm tra phát hành.*
* *Kiểm tra hệ thống của nhóm phát triển nên tập trung vào việc phát hiện lỗi trong hệ thống (kiểm tra lỗi). Mục tiêu của thử nghiệm phát hành là kiểm tra xem hệ thống có đáp ứng các yêu cầu của nó và đủ tốt để sử dụng bên ngoài (kiểm tra xác thực) hay không.*

**Yêu cầu dựa trên thử nghiệm**

Kiểm tra dựa trên yêu cầu bao gồm kiểm tra từng yêu cầu và phát triển một bài kiểm tra hoặc kiểm tra cho nó.

Yêu cầu MHC-PMS:

* *Nếu bệnh nhân được biết là bị dị ứng với bất kỳ loại thuốc cụ thể nào, thì việc kê toa thuốc đó sẽ dẫn đến một thông báo cảnh báo được đưa ra cho người sử dụng hệ thống.*
* *Nếu người kê đơn chọn bỏ qua cảnh báo dị ứng, họ sẽ cung cấp lý do tại sao điều này bị bỏ qua.*

**Yêu cầu kiểm tra**

Thiết lập một hồ sơ bệnh nhân không có dị ứng được biết đến. Kê đơn thuốc cho dị ứng được biết là tồn tại. Kiểm tra xem thông báo cảnh báo có được hệ thống phát hành không.

Thiết lập một hồ sơ bệnh nhân với một dị ứng được biết đến. Kê đơn thuốc cho bệnh nhân bị dị ứng, và kiểm tra xem hệ thống có phát ra cảnh báo hay không.

Thiết lập một hồ sơ bệnh nhân trong đó dị ứng với hai hoặc nhiều loại thuốc được ghi lại. Kê toa cả hai loại thuốc này một cách riêng biệt và kiểm tra xem có cảnh báo chính xác cho từng loại thuốc không.

Kê toa hai loại thuốc mà bệnh nhân bị dị ứng. Kiểm tra xem hai cảnh báo có được cấp đúng hay không.

Quy định một loại thuốc phát hành cảnh báo và ghi đè cảnh báo đó. Kiểm tra xem hệ thống có yêu cầu người dùng cung cấp thông tin giải thích tại sao cảnh báo bị loại bỏ hay không.

**Tính năng thử nghiệm theo kịch bản(Features Tested By Scenario)**

Xác thực bằng cách đăng nhập vào hệ thống.

Tải xuống và tải lên hồ sơ bệnh nhân được chỉ định cho máy tính xách tay.

Trang chủ lịch trình truy cập.

Mã hóa và giải mã hồ sơ bệnh nhân trên thiết bị di động.

Ghi lại và sửa đổi bản ghi.

Liên kết với cơ sở dữ liệu thuốc duy trì thông tin tác dụng phụ.

Hệ thống nhắc cuộc gọi.

**Kiểm tra năng suất**

Một phần của thử nghiệm phát hành có thể bao gồm kiểm tra các thuộc tính nổi bật của một hệ thống, chẳng hạn như hiệu suất và độ tin cậy.

Các xét nghiệm phải phản ánh hồ sơ sử dụng của hệ thống.

Các thử nghiệm hiệu suất thường liên quan đến việc lên kế hoạch cho một loạt các thử nghiệm mà tải được tăng đều đặn cho đến khi hiệu năng hệ thống trở nên không thể chấp nhận được.

Thử nghiệm ứng suất là một hình thức kiểm tra hiệu suất trong đó hệ thống được cố tình quá tải để kiểm tra hành vi lỗi của nó.

**Kiểm tra người dùng**

Thử nghiệm của người dùng hoặc khách hàng là một giai đoạn trong quá trình thử nghiệm trong đó người dùng hoặc khách hàng cung cấp đầu vào và tư vấn về kiểm tra hệ thống.

Kiểm tra người dùng là cần thiết, ngay cả khi hệ thống toàn diện và thử nghiệm phát hành đã được thực hiện.

* *Lý do cho điều này là ảnh hưởng từ môi trường làm việc của người dùng có ảnh hưởng lớn đến độ tin cậy, hiệu suất, khả năng sử dụng và tính chắc chắn của một hệ thống. Đây không thể được nhân rộng trong một môi trường thử nghiệm.*

**Các loại thử nghiệm người dùng**

**Alpha Testing**

* *Người dùng phần mềm làm việc với nhóm phát triển để kiểm tra phần mềm tại trang web của nhà phát triển.*

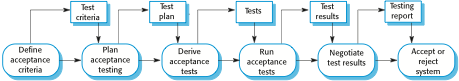
**Beta Testing**

* *Việc phát hành phần mềm được cung cấp cho người dùng để cho phép họ thử nghiệm và nâng cao các vấn đề mà họ phát hiện với các nhà phát triển hệ thống.*

**Kiểm tra chấp nhận**

* *Khách hàng thử nghiệm một hệ thống để quyết định xem nó có sẵn sàng để được chấp nhận từ các nhà phát triển hệ thống và được triển khai trong môi trường khách hàng hay không. Chủ yếu cho các hệ thống tùy chỉnh.*

**Quá trình kiểm tra nghiệm thu**

****

**Các giai đoạn trong quá trình kiểm tra nghiệm thu**

Xác định tiêu chí chấp nhận

Kiểm tra chấp nhận kế hoạch

Thử nghiệm chấp nhận có nguồn gốc

Chạy thử nghiệm chấp nhận

Thương lượng kết quả kiểm tra

Từ chối / chấp nhận hệ thống

**Phương thức nhanh và thử nghiệm chấp nhận**

Trong các phương thức nhanh, người dùng / khách hàng là một phần của nhóm phát triển và chịu trách nhiệm đưa ra các quyết định về khả năng chấp nhận của hệ thống.

Kiểm tra được xác định bởi người dùng / khách hàng và được tích hợp với các thử nghiệm khác ở chỗ chúng được chạy tự động khi thay đổi được thực hiện.

Không có quy trình thử nghiệm chấp nhận riêng biệt.

Vấn đề chính ở đây là liệu người dùng được nhúng có 'điển hình' hay không và có thể đại diện cho quyền lợi của tất cả các bên liên quan của hệ thống.

**Những điểm chính**

Khi kiểm tra phần mềm, bạn nên cố gắng 'phá vỡ' phần mềm bằng cách sử dụng kinh nghiệm và hướng dẫn để chọn loại trường hợp thử nghiệm đã có hiệu quả trong việc phát hiện khuyết tật trong các hệ thống khác.

Bất cứ nơi nào có thể, bạn nên viết các bài kiểm tra tự động. Các bài kiểm tra được nhúng trong một chương trình có thể chạy mỗi lần thay đổi được thực hiện cho một hệ thống.

Phát triển thử nghiệm đầu tiên là một cách tiếp cận để phát triển, nơi các bài kiểm tra được viết trước khi mã được kiểm tra.

Thử nghiệm kịch bản bao gồm việc phát minh ra một kịch bản sử dụng điển hình và sử dụng điều này để lấy ra các trường hợp thử nghiệm.

Thử nghiệm chấp nhận là một quá trình thử nghiệm người dùng, nơi mục đích là để quyết định xem phần mềm có đủ tốt để được triển khai và sử dụng trong môi trường hoạt động của nó hay không.