1. **Định nghĩa về kiểm thử phần mềm**

Kiểm thử phần mềm là quá trình đánh giá chất lượng của một phần mềm bằng cách thực hiện các ca kiểm thử, các kịch bản và quy trình để phát hiện các lỗi, thiếu sót, hoặc khuyết điểm trong phần mềm. Mục đích của kiểm thử phần mềm là đảm bảo rằng phần mềm hoạt động đúng và đáp ứng các yêu cầu của người sử dụng, đồng thời giảm thiểu rủi ro và chi phí sửa chữa trong tương lai.

Kiểm thử phần mềm bao gồm nhiều phương pháp và kỹ thuật khác nhau, bao gồm kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống và kiểm thử chấp nhận. Các kỹ thuật này thường được sử dụng để kiểm tra tính toàn vẹn, tính khả dụng, tính đáp ứng, tính bảo mật, tính ổn định và tính tương thích của phần mềm. Kiểm thử phần mềm là một phần quan trọng trong quá trình phát triển phần mềm và đảm bảo rằng phần mềm được phát hành với chất lượng cao nhất có thể.

1. **Trình bày các bước cơ bản trong quá trình kiểm thử phần mềm và giải thích ý nghĩa của từng bước**

* **Các bước cơ bản trong quá trình kiểm thử phần mềm**

1. Xác định yêu cầu kiểm thử: Bước này bao gồm việc xác định các yêu cầu về chức năng, hiệu suất và độ tin cậy của phần mềm cần được kiểm thử.

2. Lập kế hoạch kiểm thử: Bước này bao gồm thiết kế kế hoạch kiểm thử và xác định các tài nguyên cần thiết để thực hiện kiểm thử.

3. Thiết kế kiểm thử: Trong bước này, các ca kiểm thử cụ thể sẽ được thiết kế bao gồm các bước kiểm thử và các kết quả mong đợi.

4. Thực hiện kiểm thử: Bước này bao gồm thực hiện các ca kiểm thử đã thiết kế và ghi lại kết quả.

5. Đánh giá kết quả kiểm thử: Bước này bao gồm đánh giá các kết quả kiểm thử và so sánh chúng với các tiêu chuẩn được đặt ra trước đó.

6. Báo cáo kết quả kiểm thử: Bước này bao gồm việc báo cáo kết quả kiểm thử cho các bên liên quan và đề xuất các biện pháp khắc phục những lỗi được phát hiện.

* **Giải thích ý nghĩa các bước**

Ý nghĩa của các bước này là giúp đảm bảo rằng phần mềm được kiểm thử đầy đủ và chính xác, từ đó giảm thiểu rủi ro của lỗi phần mềm khi đưa sản phẩm vào hoạt động và đảm bảo chất lượng sản phẩm. Ngoài ra, các bước này còn giúp tăng tính hiệu quả và độ tin cậy của phần mềm, đồng thời giảm thiểu thời gian và chi phí cho quá trình phát triển phần mềm.

1. **Phát biểu định nghĩa về kiểm thử đơn vị và kiểm thử thích hợp. Nêu sự khác nhau giữa hai hình thức kiểm thử này.**

* **Định nghĩa kiểm thử đơn vị và kiểm thử thích hợp**
* Kiểm thử đơn vị (Unit testing) là quá trình kiểm tra các thành phần của phần mềm, như hàm, module, class, phương thức, để đảm bảo chúng hoạt động đúng như mong đợi. Kiểm thử đơn vị được thực hiện bởi các nhà phát triển và được tích hợp vào quá trình phát triển phần mềm.
* Kiểm thử thích hợp (Integration testing) là quá trình kiểm tra các thành phần của phần mềm sau khi đã được kết hợp với nhau để đảm bảo chúng hoạt động đúng trong môi trường hệ thống thực tế. Kiểm thử thích hợp thường được thực hiện sau khi kiểm thử đơn vị và trước khi kiểm thử hệ thống.
* **Sự khác nhau giữa hai hình thức kiểm thử này là:**
* Phạm vi kiểm thử: Kiểm thử đơn vị tập trung vào kiểm tra từng thành phần riêng lẻ của phần mềm, trong khi kiểm thử thích hợp tập trung vào kiểm tra tính tương tác giữa các thành phần.
* Đối tượng kiểm thử: Kiểm thử đơn vị kiểm tra các đơn vị cấp thấp như hàm, module, class, phương thức, trong khi kiểm thử thích hợp kiểm tra các đơn vị cấp cao hơn như các tính năng của phần mềm hoặc toàn bộ hệ thống.
* Thời gian kiểm thử: Kiểm thử đơn vị được thực hiện trong quá trình phát triển phần mềm, trong khi kiểm thử thích hợp thường được thực hiện sau khi phát triển xong và trước khi triển khai hệ thống.
* Người thực hiện kiểm thử: Kiểm thử đơn vị thường được thực hiện bởi các nhà phát triển, trong khi kiểm thử thích hợp thường được thực hiện bởi các chuyên gia kiểm thử hoặc nhóm kiểm thử độc lập.

1. **Trình bày các kỹ thuật kiểm thử phần mềm và giải thích ý nghĩa của từng kỹ thuật.**

* **Kiểm thử hộp đen (Black box testing):**
* Kỹ thuật kiểm thử này tập trung vào kiểm tra các đầu vào và đầu ra của phần mềm mà không quan tâm đến cách thức hoạt động bên trong của phần mềm.
* Ý nghĩa của kỹ thuật kiểm thử hộp đen là đảm bảo tính đầy đủ và chính xác của các chức năng của phần mềm.
* **Kiểm thử tích lũy (Incremental testing):**
* Kỹ thuật kiểm thử này tập trung vào kiểm tra từng phần của phần mềm một cách độc lập và kết hợp chúng lại để kiểm tra tính tương tác giữa các phần.
* Ý nghĩa của kỹ thuật kiểm thử tích lũy là đảm bảo tính liên kết và tính tương tác giữa các phần của phần mềm.
* **Kiểm thử hiệu năng (Performance testing):**
* Kỹ thuật kiểm thử này tập trung vào kiểm tra khả năng của phần mềm trong việc xử lý các tải trọng khác nhau.
* Ý nghĩa của kỹ thuật kiểm thử hiệu năng là đảm bảo tính ổn định và hiệu suất của phần mềm trong môi trường thực tế.
* **Kiểm thử bảo mật (Security testing):**
* Kỹ thuật kiểm thử này tập trung vào kiểm tra tính bảo mật của phần mềm bằng cách tìm kiếm các lỗ hổng bảo mật và các vấn đề liên quan đến quản lý quyền truy cập.
* Ý nghĩa của kỹ thuật kiểm thử bảo mật là đảm bảo tính bảo mật và độ an toàn của phần mềm trong môi trường thực tế.
* **Kiểm thử chấp nhận (Acceptance testing):**
* Kỹ thuật kiểm thử này tập trung vào kiểm tra tính đúng đắn và đầy đủ của các yêu cầu của khách hàng.
* Ý nghĩa của kỹ thuật kiểm thử chấp nhận là đảm bảo tính thỏa mãn của khách hàng đối với phần mềm và tính đúng đắn của các yêu cầu.

1. **Nêu lợi ích của việc sử dụng phương pháp kiểm thử tự động trong quá trình phát triển phần mềm**

- Tiết kiệm thời gian và chi phí: Kiểm thử tự động cho phép tự động hóa quá trình kiểm thử, giúp tiết kiệm thời gian và chi phí so với việc kiểm thử thủ công.

- Tăng tính chính xác: Kiểm thử tự động giúp loại bỏ các sai sót do con người gây ra trong quá trình kiểm thử thủ công, giúp đảm bảo tính chính xác của kết quả kiểm thử.

- Tăng độ phủ kiểm thử: Kiểm thử tự động cho phép kiểm thử các chức năng của phần mềm một cách đầy đủ và liên tục, giúp tăng độ phủ kiểm thử.

- Dễ dàng tái sử dụng: Kiểm thử tự động cho phép tái sử dụng các ca kiểm thử đã được thiết lập, giúp tiết kiệm thời gian và chi phí trong quá trình phát triển phần mềm.

- Tăng độ tin cậy: Kiểm thử tự động giúp đảm bảo tính tin cậy của phần mềm, giúp giảm thiểu rủi ro liên quan đến các lỗi phát sinh trong quá trình sử dụng phần mềm.

- Giúp đáp ứng nhanh hơn với các thay đổi: Kiểm thử tự động giúp giảm thời gian phát triển phần mềm và giúp đáp ứng nhanh hơn với các thay đổi trong yêu cầu của khách hàng hoặc thị trường.

1. **Các công cụ kiểm thử phần mềm**

* Selenium
* Appium
* TestComplete
* JMeter
* Cucumber
* Postman

1. **Flowchart kiểm thử**

Flowchart kiểm thử là một biểu đồ được sử dụng để mô tả quá trình kiểm thử phần mềm. Nó cho phép người dùng hình dung được các bước kiểm thử cần thực hiện và các điều kiện cần thiết để hoàn thành một bài kiểm thử. Các bước trong flowchart kiểm thử thường bao gồm:

1. Chuẩn bị kiểm thử: Xác định các tài nguyên cần thiết cho kiểm thử, bao gồm phần mềm kiểm thử, tài liệu hướng dẫn, môi trường kiểm thử, v.v.

2. Xác định yêu cầu kiểm thử: Xác định các yêu cầu kiểm thử dựa trên các tài liệu hướng dẫn và yêu cầu của khách hàng.

3. Thiết kế kiểm thử: Thiết kế các ca kiểm thử dựa trên các yêu cầu kiểm thử đã xác định.

4. Thực hiện kiểm thử: Thực hiện các ca kiểm thử đã thiết kế và ghi lại kết quả kiểm thử.

5. Phân tích kết quả kiểm thử: Phân tích kết quả kiểm thử để tìm ra các lỗi và khuyết điểm của phần mềm.

6. Báo cáo kết quả kiểm thử: Tổng hợp kết quả kiểm thử và báo cáo cho các bên liên quan.

Flowchart kiểm thử có thể được sử dụng để mô tả quá trình kiểm thử từ đầu đến cuối hoặc chỉ mô tả một phần trong quá trình kiểm thử. Nó cũng có thể được sử dụng để tạo các kế hoạch kiểm thử và phân bổ công việc cho các thành viên trong nhóm kiểm thử.

**Kỹ thuật xác định vùng tương đương:**

1)Xác định vùng tươnng đương:

Vùng tương đương->tạo ra phân hoạch->xác định bởi mqh tương đương

Xác định dựa vào

+Phân tích mã nguồn chương trình

+Hiểu biết về chức năng của chương trình

2)-Kiểm thử tương đương yếu:

+chỉ yêu cầu mỗi không gian tương đương có ít nhất 1 phần tử xuất hiện

+)lập qui tắc để tự đông xây dựng dễ dàng và dễ thấy số ca kiểm thử tối thiểu là số lớp tương đương lớn nhất của các miền

-kiểm thử tương đương mạnh:

+kết hợp cá thể của các miền tương đương

-kiểm thử tương đương đơn giản:

-chỉ phân chia 1lớp gồm các giá trị hợp lệ và các miền giá trị không hợp lệ

3)Kinh nghiệm áp dụng:

+sd khi ngôn ngữ có kiểm tra kiểu mạnh

=nếu cần kiểm tra ngoại lê cta nên mở rộng kthử vs gtrị ngoài miền xđ

-dữ liệu đầu vào có miền giá trị là khoảng và hữu hạn

-kết hợp vs kiểm thử biên sẽ tốt hơn

-nên dúng vs các chương trình phức tạp

-mạnh khi các biến là độc lập.khi các biến phụ thuộc dễ tạo ra kiểm thử vô lí

-cần số lần thử để tìm xác đinh các lớp tương đương đúng

**Phân tích gtrị biên**

1)+Là kỹ thuật thiết kế testcase và hoàn thành phân vùng tương đương

+Tập trung vào biên của miền xác định vì lỗi thường hay xảy ra ở đây

+ctr viết bằng ngôn ngữ không có kiểm tra kiểu mạnh càng cần

2)giả sử P có 2 biến a<y1<b,c<y2<d ctr sẽ nhận n đầu vào trong kgian n chiều

3)-trường hợp 1 biến:

+ giá trị nhỏ nhất +ngay trên gia trị nhỏ nhất

+một giá trị bth +ngay dưới gtrị lớn nhất +giá trị lớn nhất

-trường hợp 2 biến: a<x1<b;c<x2<d

+trục x1x2 lấy các grtị theo 4 cạnh giao cắt

-trường hợp nhiều biến

+điểm trên biên điểm cực biên điểm ngoài biên điểm trong biên

3)cách tạo các ca kiểm thử:

-phân tích gtrị biên dựa trên nguyên lý giả định khuyếm khuyết đơn:”hỏng hóc xảy ra hiếm khi do 2(hoặc hơn) khuyếm khuyết cùng lúc”

-tạo bằng việc lấy các gtrị bth của các chiều biên rồi thay mỗi chiều = các gtrị như trên

4)các loại

-kiểm thử biên mạnh:mở rộng đơn giản BVA bổ sung thêm 2 gtrị ngoài biên

-kiểm thử trường hợp xấu nhất:

+khi các biến có tương tác vs nhau cần kiểm tra bộ kết hợp

+có thể kết hợp vs kiểm thử mạnh

-Kiểm thử gtrị đặc biệt:

+được thực hiện nhiều nhất trên thực tế và cx trực quan nhất ko có dạng cố định

+sd kỹ nghệ và kiến thức miền ứng dụng để phán đoán đưa ra ca kiểm thử

+mang tính chủ quan cao nhưng vẫn là pp hiệu quả để phát hiện khuyếm khuyết

6)Ưu và nhược:

Ưu: thay vì phải test hết toàn bộ gtr thì tập trung vào kiểm thử các gtrị biên của miền gtrị inputs để thiết kế TC.Tiết kiệm thời gian thiết kế và thực hiện TC

Nhược:chỉ hiệu quả trong trường hợp các đối số đầu vào là độc lập và mỗi đố số đều có một miền gtrị hữu hạn

**Bảng quyết định**

1)+Là kỹ thuật đc sd để kiểm thử hđ hệ thống khi kết hợp các đầu vào khác nhau

+là 1 cách tiếp cần có hệ thống trong đó các kết hợp đầu vào khác nhau và hành vi hệ thống tương ứng của chúng(đầu ra) đc ghi lại dưới dạng bảng

+đc gọi là bảng nguyên nhân -ảnh hưởng

3)cách tạo TC sd bảng quyết định

+hđ tốt khi đc kết hợp vs phân vùng tương đương

+Các bược:

1Liệt kê các Conditions/Inputs 2Tính slg kết hợp có thể(Rules)

3dặt các kết hợp trong bảng 4)giảm thiểu TC kết hợp và quyết định TC

+cách giảm TCkết hợp và quyết định TC:

1nhận biết TC có chung action và chỉ khác nhau ở 1 condition duy nhất

2chuyển gtrị condition ở 1 thành – và loại bỏ bớt case

4)mức độ qtr của bqd khi viết TC

-giúp test các business logic phức tạp 1 cách dễ dàng hơn

-giá trị biên và phân vùng tương đương sd nếu hệ thống hiển thị cùng 1 kq đầu ra của 1 tập hợp các input-đầu vào

-mỗi 1 bộ gtrị đàu vào khác nhau,kq đầu ra khác nhau thì 2 kĩ thuật trên ko hquả

-đảm bảo đc độ bao phủ của TC vs cách trình bày đơn giản dễ sd

-sd làm tài liệu tham khỏa, tlại ghi lại yc của dự án hoặc dùng để ptr các chức năng

5) Ưu và Nhược

Ưu:

+dễ dàng xây dựng và chuyển dổi thành 1 bộ quy tắc

+phát hiện ra 1số TC mà theo casch thưognf dễ bị thiếu

+đc dùng làm tliêu khi làm việc vs stakeholders vì bảng giúp mng dễ hiểu

Nhược:

Khi slg input tăng thì bảng quyết định sẽ trờ nên phức tạp

-Không có các bước chi tiết step by step để thực hiện test

**Kiểm thử tổ hợp**

1)có những lỗi chỉ xuất hiện do két hợp 1 số điều kiện

+đẻ ktra kỹ thì phải ktra tất các tổ hợp của các đk có thể

2)hướng tiếp cẩn:kiểm thử đôi 1

+kq nghiên cứu cho thấy kthử đổi 1 có knăg phát hiện 80%lỗi

+là 1 trường hợp đặc biệt của kiểm thử tất cả các tổ hợp

**Đề kiểm tra cuối kì**

**Câu 1: Hãy so sánh đặc điểm, ưu nhược điểm của dạng testcase thông thường và dạng checklist**

Đặc điểm và ưu nhược điểm của dạng test case thông thường và dạng checklist như sau:

* Dạng test case thông thường:

- Đặc điểm: Bao gồm một danh sách các bước thực hiện để kiểm tra tính năng hoặc chức năng của sản phẩm. Mỗi bước kiểm tra được mô tả chi tiết, bao gồm các giá trị đầu vào cần nhập và kết quả mong đợi.

- Ưu điểm:

+ Đảm bảo kiểm tra toàn diện các tính năng hoặc chức năng của sản phẩm.

+ Dễ dàng tái sử dụng các test case đã được thiết kế.

+ Giúp định vị và khắc phục các lỗi một cách chính xác và nhanh chóng.

- Nhược điểm:

+ Yêu cầu nhiều thời gian và công sức để thiết kế các test case.

+ Không phù hợp với các tính năng hoặc chức năng phức tạp và đa dạng.

* Dạng checklist:

- Đặc điểm: Bao gồm một danh sách các mục cần kiểm tra để đảm bảo tính đầy đủ và chính xác của sản phẩm.

- Ưu điểm:

+ Dễ dàng thiết kế và sử dụng checklist.

+ Phù hợp với các tính năng hoặc chức năng đơn giản và ít phức tạp.

+ Giúp đảm bảo tính đầy đủ và chính xác của sản phẩm.

- Nhược điểm:

+ Không đảm bảo kiểm tra toàn diện và chi tiết các tính năng hoặc chức năng của sản phẩm.

+ Không giúp định vị và khắc phục các lỗi một cách chính xác và nhanh chóng.

Tóm lại, cả hai dạng kiểm thử đều có ưu nhược điểm riêng. Việc lựa chọn sử dụng loại nào phụ thuộc vào tính phù hợp với sản phẩm cần kiểm thử và mức độ phức tạp của tính năng hoặc chức năng cần kiểm thử.

Ảnh có chứa biểu đồ

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động

Ảnh có chứa bàn

Mô tả được tạo tự động