**Câu 1: trình bày các thuộc tính của mô hình chất lượng phần mềm theo chuẩn ISO/IEC 9126**

KN

ISO/IEC 9126 là tiêu chuẩn quốc tế đánh giá phần mềm. Trên thực tế, tiêu chuẩn ISO/IEC 9126 được mô tả là một phương pháp phân loại và chia nhỏ những thuộc tính chất lượng, nhằm tạo nên những đại lượng đo đếm được dùng để kiểm định chất lượng của sản phẩm phần mềm. Các cấp độ của mô hình trên bao gồm:

* ISO/IEC 9126-1 trình bày về mô hình chất lượng sản phẩm phần mềm.
* ISO/IEC 9126-2 trình bày về các phép đánh giá ngoài.
* ISO/IEC 9126-3 trình bày về các phép đánh giá trong.
* ISO/IEC 9126-4 các phép đánh giá cho chất lượng sản phẩm phần mềm trong quá trình sử dụng

Trả lời câu 1:

**1. Tính chức năng**

Khả năng của phần mềm cung cấp các chức năng đáp ứng được nhu cầu sử dụng khi phần mềm làm việc trong điều kiện cụ thể.

* Tính phù hợp: là khả năng của một phần mềm có thể cung cấp một tập các chức năng thích hợp cho công việc cụ thể phục vụ mục đích của người sử dụng.
* Tính chính xác: là khả năng của phần mềm có thể cung cấp các kết quả hay hiệu quả đúng đắn hoặc chấp nhận được với độ chính xác cần thiết.
* Khả năng làm việc hợp tác: khả năng tương tác với một hoặc một vài hệ thống cụ thể của phần mềm.
* Tính an toàn: khả năng bảo vệ thông tin và dữ liệu của sản phẩm phần mềm, sao cho những người, hệ thống không được phép thì không thể truy cập, đọc hay chỉnh sửa chúng.
* Tính tuân thủ chức năng: các phần mềm theo các chuẩn, quy ước, quy định.

**2. Tính tin cậy**

Là khả năng của phần mềm có thể hoạt động ổn định trong những điều kiện cụ thể.

* Tính chính xác: khả năng tránh các kết quả sai.
* Khả năng chịu lỗi: khả năng của phần mềm hoạt động ổn định tại một mức độ cả trong trường hợp có lỗi xảy ra ở phần mềm hoặc có những vi phạm trong giao diện.
* Khả năng phục hồi: khả năng của phần mềm có thể tái thiết lại hoạt động tại một mức xác định và khôi phục lại những dữ liệu có liên quan trực tiếp đến lỗi.
* Tính tuân thủ tin cậy: phần mềm thoả mãn các chuẩn, quy ước, quy định

**3. Tính khả dụng**

Là khả năng của phần mềm có thể hiểu được, học được, sử dụng được và hấp dẫn người sử dụng trong từng trường hợp sử dụng cụ thể.

* Có thể hiểu được: người sử dụng có thể hiểu được xem phần mềm có hợp với họ không và và sử dụng chúng thế nào cho những công việc cụ thể.
* Có thể học được: người sử dụng có thể học các ứng dụng của phần mềm
* Có thể sử dụng được: khả năng của phần mềm cho phép người sử dụng sử dụng và điều khiển nó.
* Tính hấp dẫn: khả năng hấp dẫn người sử dụng của phần mềm.
* Tính tuân thủ khả dụng: phần mềm thoả mãn các chuẩn, quy ước, quy định.

**4. Tính hiệu quả**

Khả năng của phần mềm có thể hoạt động một cách hợp lý, tương ứng với lượng tài nguyên nó sử dụng, trong điều kiện cụ thể.

* Đáp ứng thời gian: khả năng của phần mềm có thể đưa ra trả lời, thời gian xử lý và một tốc độ thông lượng hợp lý khi thực hiện công việc của mình, dưới điều kiện làm việc xác định.
* Tận dụng tài nguyên: khả năng của phần mềm có thể sử dụng một số lượng, một loại tài nguyên hợp lý để thực hiện công việc trong những điều kiện cụ thể.
* Tính tuân thủ hiệu quả: thoả mãn các chuẩn, quy ước, quy định.

**5. Khả năng bảo hành, bảo trì**

Khả năng của phần mềm có thể chỉnh sửa.

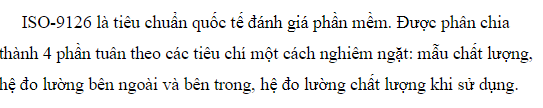
* Việc chỉnh sửa bao gồm: sửa lại cho đúng, cải tiến và làm phần mềm thích nghi được với những thay đổi của môi trường, của yêu cầu và của chức năng xác định.
* Có thể phân tích được: phần mềm có thể được chẩn đoán để tìm những thiếu sót hay những nguyên nhân gây lỗi hoặc để xác định những phần cần sửa.
* Có thể thay đổi được: phần mềm có thể chấp nhận một số thay đổi cụ thể trong quá trình triển khai.
* Tính bền vững: khả năng tránh những tác động không mong muốn khi chỉnh sửa phần mềm.
* Có thể kiểm tra được: khả năng cho phép phần mềm chỉnh sửa có thể được đánh giá.
* Khả năng tuân thủ bảo trì: thoả mãn các chuẩn, quy ước, quy định.

**6. Tính khả chuyển**

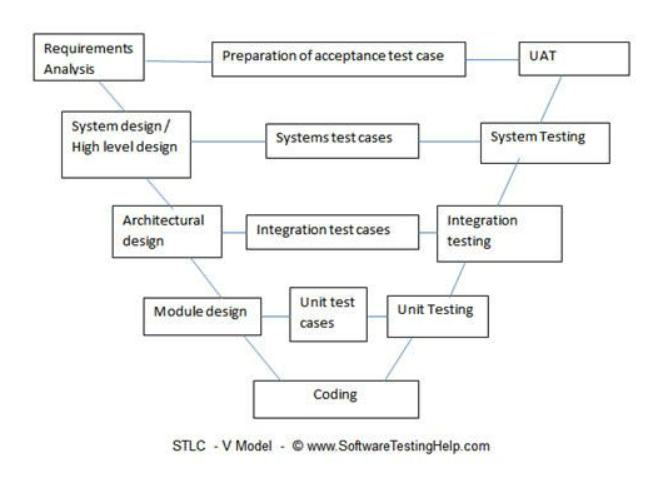
Là khả năng của phần mềm cho phép nó có thể được chuyển từ môi trường này sang môi trường khác.

* Khả năng thích nghi: khả năng của phần mềm có thể thích nghi với nhiều môi trường khác nhau mà không cần phải thay đổi.
* Có thể cài đặt được: phần mềm có thể cài đặt được trên những môi trường cụ thể.
* Khả năng cùng tồn tại: phần mềm có thể cùng tồn tại với những phần mềm độc lập khác trong một môi trường chung, cùng chia sẻ những tài nguyên chung.
* Khả năng thay thế: phần mềm có thể dùng thay thế cho một phần mềm khác, với cùng mục đích và trong cùng môi trường.
* Tính tuân thủ khả chuyển: thoả mãn các chuẩn, quy ước, quy định.

**Câu 2 :trình bày các thành phần của mô hình chất lượng theo chuẩn ISO/IEC 9126**



**Câu 3.trình bày các loại kiểm thử trong mô hình chữ V**



**1. Phân tích yêu cầu:**

Trong giai đoạn này các yêu cầu được thu thập, phân tích và nghiên cứu. Ở giai đoạn này việc hệ thống chạy như thế nào không quan trọng, quan trọng là hệ thống có những chức năng gì. Brain storming/walkthrough, interviews cần được thực hiện để có mục tiêu rõ ràng.

* Hoạt động xác minh:Đánh giá yêu cầu (Requirements review).
* Hoạt động xác nhận: Tạo test case UAT (User acceptance test- kiểm thử chấp nhận) = Đầu ra cần có: Tài liệu hiểu về yêu cầu, UAT test case.

**2. Yêu cầu hệ thống / High level design:**

Trong giai đoạn này, high level design của phần mềm được xây dựng. Nhóm sẽ nghiên cứu và điều tra về các yêu cầu có thể được thực hiện như thế nào. Tính khả thi về mặt kỹ thuật của yêu cầu cũng được tìm hiểu. Nhóm cũng tìm hiểu về các mô-đun sẽ được tạo / phụ thuộc, nhu cầu phần cứng / phần mềm

* Hoạt động xác minh: Đánh giá thiết kế (Design reviews)
* Hoạt động xác nhận: Tạo test plan và test case, tạo ma trân truy vết (traceability metrics)
* Đầu ra cần có: System test cases, Feasibility reports, System test plan, tài liệu về yêu cầu phần cứng và các mô đun, …

**3. Thiết kế kiến trúc:**

Trong giai đoạn này, dựa trên thiết kế mức cao, kiến trúc phần mềm được tạo ra. Các mô-đun, mối quan hệ và sự phụ thuộc của họ, sơ đồ kiến trúc, bảng cơ sở dữ liệu, chi tiết về công nghệ đều được hoàn tất trong giai đoạn này.

* Hoạt động xác minh: Đánh giá thiết kế
* Hoạt động xác nhận: Kế hoạch thử nghiệm tích hợp và các trường hợp thử nghiệm.
* Đầu ra cần có: Tài liệu thiết kế, Kế hoạch kiểm thử tích hợp và các trường hợp thử nghiệm, Thiết kế bảng cơ sở dữ liệu,…

**4. Thiết kế mô đun / Thiết kế cấp thấp:**

Trong giai đoạn này mỗi mô-đun hoặc các thành phần phần mềm đều được thiết kế riêng. Các method, class, giao diện, các kiểu dữ liệu vv đều được hoàn tất trong giai đoạn này.

* Hoạt động xác minh: Đánh giá thiết kế
* Hoạt động xác nhận: Tạo và xem xét các trường hợp kiểm tra đơn vị.
* Đầu ra cần có: Các đơn vị kiểm tra đơn vị

**5. Code:**

Trong giai đoạn này, code được thực hiện.

* Hoạt động xác minh: Xem xét mã, kiểm tra các trường hợp kiểm tra
* Hoạt động xác nhận: Tạo các trường hợp kiểm tra chức năng.
* Đầu ra cần có: các trường hợp thử nghiệm, danh sách kiểm tra xem lại.

**1. Kiểm tra đơn vị (unit test):**

(Trong giai đoạn này tất cả unit test case đã được tạo ra trong giai đoạn thiết kế cấp thấp sẽ được thực hiện.

* Unit test là một kỹ thuật kiểm tra hộp trắng, nơi một đoạn code được viết, nó sẽ tích hợp một phương pháp (hoặc một đoạn code khác) để kiểm tra xem đoạn code có cho kết quả mong muốn hay không. Thử nghiệm này về cơ bản được thực hiện bởi nhóm phát triển. Trong trường hợp có bất thường, bug được ghi lại và theo dõi.
* Đầu ra cần có: Kết quả thực hiện unit test)

Kiểm thử đơn vị (Unit Testing) là mức kiểm thử cơ bản, trong đó các đơn vị / thành phần riêng lẻ của một phần mềm được kiểm tra. Mục đích là để xác nhận rằng mỗi đơn vị của phần mềm thực hiện như thiết kế. Một đơn vị là thành phần nhỏ nhất có thể kiểm tra, mỗi đơn vị phải có đầu vào và đầu ra để có thể kiểm tra được (loại kiểm thử White Box).

♣Trong lập trình thủ tục:đơn vị là một chương trình riêng lẻ như hàm, Module

♣Trong lập trình hướng đối tượng: đơn vị là một phương thức, có thể thuộc về một lớp cơ sở, lớp cha, lớp trừu tượng hoặc lớp dẫn xuất / lớp con.

Chi phí sửa chữa một khiếm khuyết được phát hiện trong quá trình kiểm thử ở mức đơn vị là ít hơn so với các lỗi được phát hiện ở các cấp cao hơn. Các công cụ kiểm thử đơn vị:

♣MS Test trong Visual Studio

♣NUtnit

**2. Kiểm tra tích hợp:**

(Trong giai đoạn này các trường hợp kiểm thử tích hợp được thực hiện đã được tạo ra trong giai đoạn thiết kế kiến trúc. Trong trường hợp có bất kỳ dị thường, bug được ghi lại và theo dõi.

* Kiểm thử tích hợp (Integration Testing): Kiểm thử tích hợp là một kỹ thuật mà đơn vị kiểm tra là mô-đun được tích hợp và kiểm tra xem các mô-đun tích hợp vào có cho kết quả mong đợi không. Nói một cách đơn giản, nó xác nhận xem các thành phần của ứng dụng có hoạt động với nhau như mong đợi hay không.
* Đầu ra cần có: Các kết quả kiểm tra tích hợp.)

Kiểm thử tích hợp (Integration Testing) là một cấp độ kiểm thử phần mềm trong đó các đơn vị riêng lẻ được kết hợp và kiểm tra theo một nhóm. Mục đích của kiểm thử tích hợp này là để lộ các lỗi trong tương tác giữa các đơn vị tích hợp.

Kiểm thử tích hợp xảy ra sau kiểm thử đơn vị và trước kiểm thử hệ thống.

Trình điều khiển kiểm thử và các tình huống kiểm thử được sử dụng để hỗ trợ Kiểm thử tích hợp

Cách tiếp cận, phương pháp, chiến lược của kiểm thử tích hợp:

♣Có nhiều cách tiếp cận khác nhau để kiểm thử tích hợp. Các phương pháp phổ biến và được sử dụng thường xuyên nhất là Kiểm thử Big Bang, Kiểm thử Top\_Down, Kiểm thử Bottom\_Up.

♣Sự lựa chọn của phương pháp phụ thuộc vào các yếu tố khác nhau như chi phí, độ phức tạp, mức độ quan trọng của ứng dụng, v.v. Ngoài ra, có nhiều loại kiểm thử tích hợp ít được biết đến như tích hợp dịch vụ phân tán, tích hợp Sandwich, tích hợp đường trục, tích hợp tần số cao, tích hợp lớp, v.v.

**3. Kiểm tra hệ thống (System testing):**

(Trong giai đoạn này, kiểm thử chức năng và phi chức năng được thực hiện. Nói cách khác, việc kiểm tra thực tế hoạt động của ứng dụng diễn ra ở đây. Lỗi được phát hiện và theo dõi để sửa. Báo cáo tiến độ cũng là một phần quan trọng trong giai đoạn này. Ma trận truy vết (traceability metrics) được cập nhật để kiểm tra mức độ bao phủ và rủi ro được giảm bớt.

* Đầu ra cần có: Kết quả kiểm tra, Các bản ghi kiểm tra, báo cáo lỗi, báo cáo tóm tắt kiểm tra và các ma trận truy xuất cập nhật.)

Kiểm thử hệ thống (System Testing) là kiểm thử một sản phẩm PM hoàn chỉnh và tích hợp đầy đủ. Thông thường, phần mềm chỉ là một thành phần của hệ thống trên máy tính. Phần mềm được tương tác với các hệ thống phần mềm hay phần cứng khác. Kiểm thử hệ thống thực hiện một loạt các kiểm thử khác nhau với mục đích kiểm tra phần mềm có tương thích với toàn bộ hệ thống trên máy tính hay không?.

Kiểm thử hệ thống thuộc loại kiểm thử hộp đen (Black Box Testing), dựa trên đầu vào và đầu ra của chương trình để kiểm thử, bao gồm kiểm tra các chức năng và phi chức năng của sản phẩm PM.

Sau đây là danh sách các loại kiểm thử hệ thống mà các công ty phát triển phần mềm thường sử dụng:

♣Kiểm thử khả năng sử dụng - Usability Testing: Kiểm thử khả năng sử dụng chủ yếu tập trung vào việc người dùng dễ dàng sử dụng ứng dụng, linh hoạt trong việc kiểm soát xử lý và khả năng của hệ thống để đáp ứng các mục tiêu.

♣Kiểm thử tải - Load Testing: Kiểm thử tải là cần thiết để biết rằng một phần mềm sẽ thực hiện theo tải thực tế.

♣Kiểm thử phục hồi - Recovery Testing: Kiểm thử phục hồi được thực hiện để chứng minh một giải pháp phần mềm là đáng tin cậy và có thể phục hồi thành công khi các sự cố xảy ra.

♣Kiểm thử di chuyển - Migration Testing: Kiểm thử di chuyển được thực hiện để đảm bảo rằng phần mềm có thể được chuyển từ cơ sở hạ tầng từ hệ thống cũ sang cơ sở hạ tầng hệ thống mới mà không gặp sự cố nào.

♣Kiểm thử chức năng - Functional Testing: Còn được gọi là kiểm thử tính đầy đủ của chức năng. Tester có thể lập danh sách các chức năng bổ sung mà sản phẩm có thể phải cải thiện trong quá trình kiểm thử chức năng.

♣Kiểm thử phần cứng / phần mềm - Hardware/Software Testing: IBM gọi kiểm thử phần cứng / phần mềm là Kiểm thử CTNH / SW, là khi tester tập trung sự chú ý của mình vào các tương tác giữa phần cứng và phần mềm trong quá trình kiểm thử hệ thống

**4. Acceptance Testing**

**Acceptance Testing** (Kiểm thử chấp nhận) là một kiểm thử nhằm xác định hệ thống phần mềm có đạt yêu cầu kỹ thuật hay không. Bằng việc kiểm tra các hành vi của hệ thống qua dữ liệu thực tế, kiểm thử chấp nhận sẽ xác định có hay không việc hệ thống đáp ứng được các tiêu chí lẫn yêu cầu của khách hàng. Một số kỹ thuật được sử dụng trong Acceptance Testing đó là phân tích giá trị biên giới, phân vùng tương đương và sử dụng bảng quyết định.

**Câu 4: Interface testing là gì? Các thành phần của Interface testing ?**

Kiểm thử giao tiếp (Interface testing) là kiểm tra giao tiếp phía máy chủ (Server). Điều này được thực hiện bằng cách xác minh rằng truyền thông / giao tiếp ở máy chủ được thực hiện đúng ?. Khả năng tương thích của máy chủ với phần mềm, phần cứng, mạng và cơ sở dữ liệu cần được kiểm tra.

Kiểm tra xem tất cả các tương tác giữa các máy chủ được thực thi và lỗi được xử lý đúng cách?. Các lỗi được trả về được xử lý và thông báo thích hợp cho người dùng.

2 thành phần phổ biến của Interface testing bao gồm:

* Giao tiếp giữa web server và application server
* Giao tiếp giữa database server và application server

**Câu 5: Trình bày các yêu cầu cơ bản của GUI Testing**

* Kiểm tra tất cả các yếu tố GUI cho kích thước, vị trí, chiều rộng, chiều dài và chấp nhận các ký tự hoặc số. Ví dụ, bạn phải có khả năng cung cấp đầu vào cho các trường đầu vào.
* Kiểm tra bạn có thể thực hiện các chức năng dự định của ứng dụng bằng cách sử dụng GUI
* Kiểm tra các thông điệp lỗi được hiển thị chính xác
* Kiểm tra Xoá phân cách các phần khác nhau trên màn hình
* Kiểm tra Font được sử dụng trong ứng dụng có thể đọc được
* Kiểm tra sự liên kết của văn bản là thích hợp
* Kiểm tra màu sắc của phông chữ và các thông điệp cảnh báo
* Kiểm tra xem những hình ảnh rõ ràng hay không
* Kiểm tra xem các hình ảnh đã được căn chỉnh đúng cách
* Kiểm tra vị trí của các phần tử GUI trong các màn hình có độ phân giải khác nhau

**Câu 6 :Trình bày Unit Testing, các kỹ thuật thực hiện**

**Unit Test** là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử. Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Mục tiêu của Kiểm thử đơn vị là cô lập một phần code và xác minh tính chính xác của đơn vị đó.

## Kiểm thử đơn vị có hai loại:

* Manual (thủ công)
* Automated (Tự động)

Kiểm thử đơn vị thường là tự động nhưng vẫn có thể được thực hiện thủ công. Thường thì việc tự động hóa Kiểm thử đơn vị được ưa thích hơn. Khi kiểm thử đơn vị thủ công, có thể sử dụng những tài liệu hướng dẫn mà trong đó đã ghi rõ các bước thực hiện.

UT có 3 trạng thái cơ bản:

* Fail (trạng thái lỗi)
* Ignore (tạm ngừng thực hiện)
* Pass (trạng thái làm việc)
* Toàn bộ UT được vận hành trong một hệ thống tách biệt. Có rất nhiều PM hỗ trợ thực thi UT với giao diện trực quan. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail)

UT chỉ thực sự đem lại hiệu quả khi:

* Được vận hành lặp lại nhiều lần
* Tự động hoàn toàn
* Độc lập với các UT khác.

Mỗi UT đều được tiết kế theo trình tự sau:

* Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả…
* Triệu gọi các phương thức cần kiểm tra.
* Kiểm tra sự hoạt động đúng đắn của các phương thức.
* Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.

## **Ứng dụng Unit test**

* Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.
* Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.
* Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc(workflow – tập hợp của nhiều quy trình)

lợi ích sau:

* Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn code nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.
* Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
* Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
* Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
* Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.

Trong môi trường làm việc Unit Test còn có tác dụng rất lớn đến năng suất làm việc:

* Giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.
* Tăng sự tự tin khi hoàn thành một công việc. Chúng ta thường có cảm giác không chắc chắn về các đoạn mã của mình như liệu các lỗi có quay lại không, hoạt động của module hiện hành có bị tác động không, hoặc liệu công việc hiệu chỉnh mã có gây hư hỏng đâu đó…
* Là công cụ đánh giá năng lực của bạn. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái “pass” sẽ thể hiện tốc độ làm việc, năng suất của bạn.

Các kỹ thuật bao gồm kiểm thử các đoạn mã code được tổng hợp ở dưới đây:

* Statement Coverage (Bao phủ dòng lệnh)
* Decision Coverage (Bao phủ quyết định)
* Branch Coverage (Bao phủ các nhánh)
* Condition Coverage (Bao phủ điều kiện)
* Finite State Machine Coverage (Bao phủ trạng thái hữu hạn)



