**Chapter 1.Writing a Program - Viết một chương trình**

**1. Cho biết yêu cầu phần mềm, yêu cầu chức năng, yêu cầu phi chức năng là gì? Cho ví dụ một yêu cầu chức năng và một yêu cầu phi chức năng.**

- Yêu cầu phần mềm nhằm xác định và đánh giá hệ thống, được xây dựng bởi khách hàng và dưới sự giúp đỡ của kỹ sư phần mềm.

- Yêu cầu chức năng mô tả hệ thống làm được những gì, các chức năng và dịch vụ của hệ thống một cách chi tiết.

- Yêu cầu phi chức nằng: không đề cập trực tiết tới các chưc năng cụ thể của hệ thống mà thường định nghĩa các thuộc tính: hiệu suất, thời gian chạy, khả năng sửa đổi… Một số yêu cầu phi chức năng liên quan đến quy trình xây dựng hệ thống như các chuẩn được sử dụng, các tool, ngôn ngữ lập trình…

**VD:** Đối với hệ thống quản lý bán hàng

- Yêu cầu phần mềm: xây dựng hệ thống nhằm giúp người dùng quản lý các thành phần trong cửa hàng thời trang như khách hàng, sản phẩm, đơn hàng…

- Yêu cầu chức năng: hệ thống phải đáp ứng các chức năng quản lý sản phẩm, khách hàng, hóa đơn của cửa hàng…

- Yêu cầu phi chức năng: khách hàng mong muốn sản phẩm chạy trên nên web aps.net, dùng CSDL SQL server để phù hợp với những hệ thống trước đó của cửa hàng

**2. Trình bày ba loại yêu cầu phi chức năng tiêu biểu**.

- Có 3 loại yêu cầu phi chức năng tiêu biểu:

+ Yêu cầu về sản phẩm: đề cập đến hiệu năng, khả năng sử dụng, độ tin cậy … của sản phẩm.

+ Yêu cầu về tổ chức: lấy từ những chính sách, quy tắc của khách hàng hoặc tổ chức sử dụng hệ thống.

+ Yêu cầu ngoài: được xác định từ các tác nhân bên ngoài hệ thống như phải bảo mật thông tin cá nhân của khách hàng.

**Chapter 2. Building a System – Xây dựng hệ thống**

**1. Độ phức tạp của phần mềm theo chiều rộng và chiều sâu (bread and depth issues) được xác định thông qua những yếu tố nào?**

- Theo chiều rộng:

+ Các chức năng chính

+ Các tính năng bên trong mỗi chức năng chính

+ Giao diện giữa các tính năng và chức năng

+ Giao tiếp với các hệ thống ngoài.

+ Số người dùng đồng thời và tổng số người dùng

- Theo chiều sâu:

+ Số lương các \_\_\_\_ lồng nhau

+ Số lượng quan hệ mắt xích

**2. Cho biết các vấn đề kỹ thuật và phi kỹ thuật trong phát triển các hệ thống lớn.**

- Các vấn đề kỹ thuật:

+ Đơn giản hóa các vấn đề và giải pháp

* Phân tích
* Mô đun hóa
* Sự tách biệt
* Gia tăng trùng lặp

+ Công nghệ và lựa chọn công cụ

* Nền tảng phát triển
* Ngôn ngữ phát triển
* Cơ sở dữ liệu
* Mạng
* Quản lý cấu hình

+ Tiến trình và phương pháp

* Lựa chọn tiến trình
* Lựa chọn phương pháp
* Lựa chọn công cụ cho tiến trình

- Các vấn đề phi kỹ thuật

+ Ước lượng công sức và lập lịch trình cho dự án

* Cần xem xét và ước lượng nhiều mục hơn
* Cần điều phối nhiều mục trong điều khoản của các điều kiện tien quyết và cần thiết
* Cần xem xét đến các khả năng rủi ro và sự thay đổi

+ Nhiệm vụ và sự giao tiêp

* Nhiều thành viên tham gia thì càng cần nhiều các kỹ năng đa dạng
* Sự giao tiếp giữa các thành viên với nhau
* Lỗi và sự thay đổi

**3. Cho biết các yếu tố cần xem xét để xác định số lượng người cần có cho các công việc sau khi chuyển giao phần mềm cho khách hàng.**

- Chuẩn bị và đào tạo trước khi chuyển giao phần mềm

+ Số lượng user kỳ vọng đạt được

+ Số lượng các vấn đề khi triển khai và chất lượng phần mềm được kỳ vọng.

+ Số lượng người dùng và đội ngũ hỗ trợ được đào tạo

+ Số vòng sửa chữa và bảo trì

- Hỗ trợ khách hàng và người dùng sau khi chuyển giao

+ Trung tâm xử lý cuộc gọi và giải pháp cho mỗi vấn đề

+ Sửa lỗi các vấn đề nghiêm trọng và thay đổi chương trình

+ Chỉnh sửa và hoàn thiện các chức năng

**Chapter 3. Engineering of software – Công nghệ phần mềm**

**1. Liệt kê các nguyên nhân chính làm một dự án phần mềm thành công và thất bại.**

- Mục tiêu của dự án không thực tế và rời rạc

- Ước lượng không chính xác về nguồn lực

- Đặc tả yêu cầu không rõ rang

- Báo cáo về trạng thái dự án kém

- Rủi ro trong quản lý

- Giao tiếp kém giữa khách hàng, các nhà phát triển và người dùng

- Sử dụng các công nghệ chưa được kiểm chứng

- Không có khả năng xử lý các vấn đề phức tạp của phần mềm

- Thói quen phát triển phần mềm tùy tiện

- Quản lý kém

- Áp lực thương mại

- Chính sách giữa các bên liên quan

**2. Công nghệ phần mềm là gì? Liêt kê các lĩnh vực có trong công nghệ phần mềm.**

**-** Công nghệ phần mềm là lĩnh vực rộng lớn bao gồm phát triển và hỗ trợ hệ thống phần mềm với các lĩnh vực chính:

+ Quy trình kỹ thuật và nghiệp vụ

+ Phương pháp và kỹ thuật cụ thể

+ Đặc trưng sản phẩm và các thước đo

+ Kỹ năng làm việc nhóm của các thành viên

+ Điều phối, quản lý dự án

**3. Cho biết tám nguyên lý về đạo đức nghề nghiệp trong công nghệ phần mềm của IEEE-CS và ACM.**

- Hoạt động nhất quán với lợi ích cộng đồng

- Hoạt động với lợi ích lớn nhất cho khách hàng, nhân viên và xem xét với các lợi ích cộng đồng

- Đảm bảo các sản phẩm và các sửa đổi gần nhất đáp ứng tiêu chuẩn nghiệp vụ cao nhất có thể

- Duy trì sự toàn vẹn và đánh giá chuyên môn

- Các nhà quản lý và người đứng đầu cần theo dõi sát sao và thúc đẩy sự phát triển quản lý phát triển và bảo trì phần mềm.

- Nâng cao uy tín và danh tiếng nghề nghiệp phù hợp với lợi ích chung.

- Đối xử công bằng và hỗ trợ các đồng nghiệp

- Tham gia học tập liên quan đến nghiệp vụ và thúc đẩy tiêu chuẩn đạo đức trong quá trình làm việc

**4. Cho biết 15 nguyên tắc xây dựng phần mềm của Davis.**

- Chất lượng là số 1

- Phần mềm chất lượng cao là có thể

- Đưa sản phẩm cho khách hàng từ sớm

- Xác định vấn đề trước khi viết yêu cầu phần mềm

- Đánh giá các lựa chọn thiết kế

- Sử dụng mô hình quy trình thích hợp

- Sử dụng ngôn ngữ khác nhau cho các giai đoạn khách nhau

- Giảm thiểu khoảng cách trí tuệ (khoảng cách giữa phần mềm và hệ thống thực tế)

- Đặt kỹ thuật trước công cụ

- Làm phần mềm chạy đúng trước khi chạy nhanh

- Thanh tra code

- Quản lý tốt quan trọng hơn công nghệ tốt

- Con người là nhân tố chính dẫn đến thành công

- Theo sát dự án phần mềm

- Chịu trách nhiệm

**Chapter 4. Software Process Models – Các mô hình tiến trình phần mềm**

**1. Vẽ biểu đồ minh họa và giải thích các hoạt động cho mô hình tiến trình phát triển phần mềm thác nước (waterfall model).**

**waterfallModel**

**-** Các giai đoạn:

+ Xác định yêu cầu

+ Thiết kế

+ Lập trình

+ Kiểm thử

+ Tích hợp và đóng gói

**-** Theo mô hình thác nước, các bước cần được thực hiện theo thứ tự nghiêm ngặt.

+ Trước hết giai đoạn phân tích yêu cầu phải được hoàn tất, kết quả nhận được là danh sách các yêu cầu của phần mềm.

+ Sau khi các yêu cầu được xác định, sẽ chuyến sang giai đoạn thiết kế. Ở giai đoạn này, người ta sẽ tạo ra các tài liệu cho lập trình viên, bao gồm mô tả chi tiết phương pháp và kế hoạch thực hiện các yêu cầu đã làm rõ ở giai đoạn trước.

+ Sau khi giai đoạn thiết kế hoàn tất, lập trình viên bắt đầu lập trình dự án họ nhận được

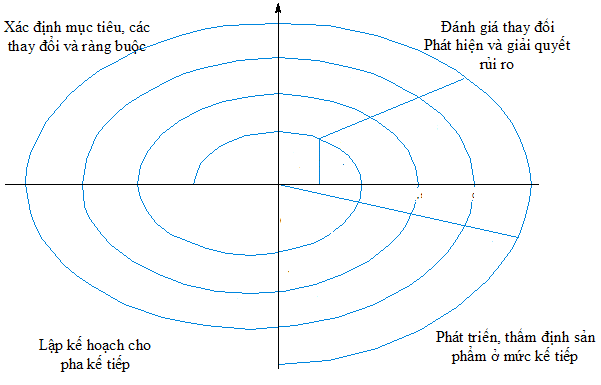
+ Tiếp đó, giai đoạn kiểm thử được bắt đầu, loại bỏ các khiếm khuyết

+ Cuối cùng, khi kiểm thử hoàn tất, dự án chuyển sang giai đoạn tích hợp và đóng gói sản phẩm để chuyển giao

- Với mô hình thác nước, nếu trong bất kỳ giai đoạn nào có sai sót thì lập tức phải quay lại làm từ bước đầu tiên.

**2. Vẽ biểu đồ minh họa và giải thích mô hình tiến trình phát triển phần mềm xoắn ốc (spiral model).**

**-**  Mô hình xoắn ốc là quy trình phát triển định hướng rủi ro cho các dự án phần mềm



- Theo mô hình xoắn ốc, quy trình phát triển phần mềm được biểu diễn như vòng xoắn ôc, bao gồm các pha:

+ Thiết lập mục tiêu: xác định mục tiêu cho từng pha của dự án

+ Đánh giá và giảm thiểu rủi ro: rủi ro được đánh giá và thực hiện các hành động để giảm thiểu rủi ro

+ Phát triển và đánh giá: sau khi đánh giá rủi ro, một mô hình xây dựng hệ thống sẽ được lựa chọn từ những mô hình chung

+ Lập kế hoạch: đánh giá dự án và pha tiếp theo của mô hình xoắn ốc sẽ được lập kế hoạch

**3. Trình bày mô hình trưởng thành về năng lực (Capability Maturity Model - CMM) do SEI (Software Engineering Institute) đề xuất.**

**-** CMM là chuẩn quản lý quy trình chất lượng của các sản phẩm phần mềm được áp dụng cho từng loại hình công ty khác nhau

- Bao gồm 5 cấp độ:

+ Khởi đầu

+ Được quản lý

+ Được xác định

+ Được quản lý định lượng

+ Tối ưu

* Khởi đầu

- Mọi công ty phần mềm, nhóm, cá nhân đều có thể đạt được. Ở cấp độ này, CMM chưa yêu cầu bất kỳ tính năng nào

- Đặc điểm:

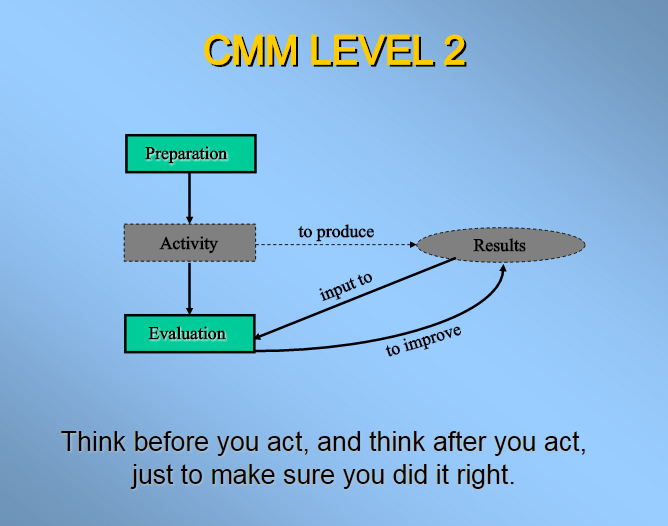
+ Hành chính: Các hoạt động của lực lượng lao động được quan tâm hàng đầu nhưng thực hiện vội vã, hấp tấp

+ Không thống nhất: đào tạo quản lý nhân lực không thống nhất, chủ yếu dựa vào kinh nghiệm cá nhân

+ Người quản lý mong bộ phận nhân sự điều hành và kiểm soát các hoạt động của lực lượng nhân sự

+ Doanh số thường xuyên thay đổi: nhân viên không trung thành với tổ chức.

* Được quản lý



- Bao gồm 6 tiến trình và có thể lặp lại:

+ Quản lý, đặc tả yêu cầu

+ Theo dõi dự án

+ Đảm bảo chất lượng

+ Lập kế hoạch dự án

+ Quản lý các hợp đồng phụ

+ Quản lý cấu hình

* Được xác định

- Bao gồm 7 tiến trình và ảnh hưởng tới cấu trúc phần mềm:

+ Tổ chức các tiến trình

+ Chương trình đào tạo

+ Kỹ thuật sản phẩm

+ Đánh giá ngang hàng

+ Xác định quy trình tổ chức

+ Tích hợp quản lý phần mềm

+ Sự phối hợp giữa các nhóm

* Được quản lý

- Được giới thiệu với nhiều hơn 2 tiến trình đó là xử lý số lượng đo lường và chất lượng

+ Quản lý chất lượng tiến trình

+ Quản lý chất lượng

* Tối ưu

- Vươn tới yêu cầu cấp độ cao nhất, làm chủ cải tiến liên tục với 3 tiến trình:

+ Phòng ngừa các khiếm khuyết

+ Quản lý sự thay đổi công nghệ

+ Quản lý thay đổi tiến trình

**Chapter 5. New and Emerging Process Methodologies - Các mô hình tiến trình phần mềm thịnh hành**

**1. Cho biết các vấn đề tồn tại của các tiến trình phát triển phần mềm truyền thống.**

**-** Thời gian phát triển dài

- Không có sự tham gia của khách hàng

- Không có khả năng xử lý các thay đổi

- Giả thiết các yêu cầu được hiểu ngay từ đầu

- Dựa vào công sức phát triển lớn

- Phương pháp phức tạp

- Lãng phí/ trùng lặp công sức phát triển

**2. Cho biết bốn giá trị chính của lập trình cực độ (Extreme Programing)**

**-** Giao tiếp giữa đội ngũ lập trình và khách hàng

- Đơn giản hóa giữa thiết kế và lập trình

- Phản hổi từ tất cả các cấp độ

- Khuyến khích thực hiện các quyết định khó khăn

**3. Liệt kê các thông lệ của lập trình cực độ**

**-** Phản hồi nhanh: sử dụng lập trình đôi, kiểm thử (unit testing), tích hợp nhanh và đưa ra các bản release

- Đơn giản; đơn giản hóa các tiếp cận nhất có thể

- Gia tăng các thay đổi: thử thêm vào các thay đổi nhỏ, sắp xếp lại code, dùng sửa đổi nhỏ để trau dồi khả năng code

- Chấp nhận thay đổi: giữ lại các lựa chọn cho tương lai, hoãn các quyết định có tính thay đổi lâu nhất có thể

- Làm việc hiệu quả: tạo ra sản phẩm tốt nhất có thể

**4. Giải thích các đặc trưng của các phương pháp/tiến trình Agile**

**-** Một họ các mô hình phát triển phần mềm

- Có các bản release nhanh:

+ Chia thành các phần nhỏ

+ Giao sản phẩm cho khách hàng nhanh nhất có thể

- Thiết kế tăng dần:

+ Không cố gắng có bản thiết kế hoàn thiện ngay từ đầu

+ Trì hoãn các quyết định thiết kế cho tới khi có đầy đủ cơ sở

- Lôi kéo người dùng vào hệ thống phát triển: cố gắng lấy được phản hồi của khách hàng thường xuyên

- Tối thiểu hóa tài liệu: chỉ sử dụng tài liệu khi thực sự cần thiết

- Thực hiện các giao tiếp thông thường: loại bỏ việc giao tiếp qua các văn bản

- Thay đổi: sẵn sang làm việc với thay đổi

- Quy mô: làm việc và giúp người khác làm việc

**5. Cho biết các khía cạnh mà phát triển nguồn mở chia sẻ với các phương pháp Agile.**

- Có các bản release nhỏ

- Giao tiếp thông thường

- Tham gia của khách hàng

- Tích hợp liên tục

- Chung tầm nhìn

**6. So sánh các phương pháp Agile với các phương pháp truyền thống.**

|  |  |  |
| --- | --- | --- |
|  | Agile | PP truyền thống |
| Yêu cầu | - Giả thiết yêu cầu thay đổi  - Xây dựng ít hình thức, không cần đặc tả  - Có tương tác của người dùng | - Yêu cầu không thay đổi  - Yêu cầu xây dựng trên tài liệu chi tiết |
| Thiết kế | - Qua các giao tiếp không có tính hình thức  - Lặp lại | - Hình thức  - Trước khi xây dựng,lập trình |
| Tham gia của người dùng | - Cần thiết, thường xuyên | - Trong 2 giai đoạn: bắt đầu và kết thúc |
| Tài liệu | - Tối thiểu  - Source code | - Tài liệu lớn, có tính hinh thức cao |
| Giao tiếp | - Giao tiếp thông thường, liên tục | - Qua tài liệu, cuộc họp, bản ghi nhớ |
| Độ phức tạp | - Thấp | - Cao |
| Tổng chi phí | - Thấp | - Cao |

**Chapter 6. Requirements Engineering – Xây dựng các yêu cầu phần mềm**

**1. Liệt kê và mô tả các bước cơ bản trong quá trình xây dựng yêu cầu phần mềm.**

**-** Thu thập thông tin: lấy thông tin từ khách hàng hoặc do chính kỹ sư phần mềm đưa ra, hướng tới tìm kiếm thông tin, nghiệp vụ của hệ thống,mục tiêu cảu dự án phần mềm

- Lập tài liệu và xác định yêu cầu

- Đặc tả yêu cầu: viết các yêu cầu theo chuẩn

- Xây dựng chương trình mẫu

- Phân tích yêu cầu

- Rà soát và xác nhận yêu cầu

- Đồng thuận và thống nhất yêu cầu

**2. Cho biết ba công việc chính cần thực hiện trước khi xây dựng yêu cầu phần mềm.**

**-** B1: Lập kế hoạch

- B2: Thống nhất về tài nguyên, quy trình, lịch trình

- B3: Tiến hành các công việc và tổ chức tài nguyên, tổ chức thực hiện tiến trình

**3. Cho biết sáu chiều cơ bản của các yêu cầu phần mềm bạn cần giải quyết khi thu thập yêu cầu phần mềm**.

- Quy trình làm việc

- Dữ liệu và định dạng dữ liệu

- Chức năng riêng biệt

- Giao diện hiện tại và hệ thống

- Các hạn chế khác: hiệu năng, độ tin cậy…

- Giao diện người dùng

**4. Mô tả các mục bạn cần xem xét khi xác định mức độ ưu tiên của các yêu cầu.**

- Yêu cầu, mong muốn hiện tại của khách hàng

- Sự cạnh tranh và điều kiện thị trường tạm thời

- Dự báo tương lai và các yêu cầu mới của khách hàng

- Các lợi thế bán hàng

- Các vấn đề nghiêm trọng của sản phẩm hiện thời

**5. Cho biết 4 loại tạo vết của các yêu cầu.**

Các loại tạo vết yêu cầu chủ yếu mô tả tổng quan về hệ thống, chứ không đi sâu vào chi tiết

**-** Mô tả đơn giản đầu vào, tiến trình, đầu ra bằng tiếng anh

- Dataflow diagrams (DFD) : chủ yếu mô tả luồng dữ liệu chạy như thế nào trong hệ thống, các dữ liệu vào hay ra của một xử lý

- Biểu đồ quan hệ thực thể (ERD): mô tả cấu trúc logic của dữ liệu được xử lý trong hệ thống, mối quan hệ giữa các thực thể trong hệ thống.

- Biểu đồ Use case (UML): mô tả kiểu của các đối tượng trong hệ thống và các loại quan hệ khác nhau tồn tại giữa chúng.

**Chapter 7. Design: Architecture and Methodology - Thiết kế**

**1. Kiến trúc phần mềm là gì? Bản mô tả kiến trúc có thể dùng vào việc gì?**

**-** Kiến trúc phần mềm là cấu trúc các thành phần bên trong hệ thống phần mềm, là bản thiết kế hệ thống ở mức cao nhất.

**-** Kiến trúc phần mềm bao gồm:

+ Các phần tử phần mềm

+ Các thuộc tính có thể truy cập từ bên ngoài

+ Mối quan hệ giữa các phần tử

- Ý nghĩa:

+ Hiểu về hệ thống và thực hiện giao tiếp giữa các bên liên quan

+ Có được sự hiểu biết giữa các bên

+ Để thương lượng và có sự đồng thuận giữa bên phát triển và khách hàng

+ Hiểu được hệ thống hiện thời

+ Tái sử dụng

+ Xây dựng và cải tiến phần mềm

+ Phân tích phần mềm

**2. Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc đường ống và bộ lọc (Pipe and filter)**

+ Có 1 kiểu thành phần là bộ lọc, 1 kiểu kết nối là đường ống. Một bộ lọc thực hiện biến đổi dữ liệu và truyền cho các bộ lọc khác qua đường ống.

+ Bộ lọc không đồng bộ, có thể nhận hoặc gửi dữ liệu

+ Bộ lọc là độc lập, không cần biết ID của bộ lọc gửi và nhận dữ liệu

+ Đường ống là kênh 2 chiều, di chuyển dữ liệu từ bộ lọc này sang bộ lọc khác

+ Bộ lọc phải đẩy dữ liệu, đồng bộ giữa các bộ lọc

+ Đường ống phải kết nối cổng đầu ra của một bộ lọc với cổng đầu vào của bộ lọc khác. Các bộ lọc có thể có luồng điều khiển độc lập.

**3. Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc Khách-Chủ (Client-Server)**

**-** Đặc điểm:

**+** Có 2 kiểu thành phần: Client (khách) và Server (chủ)

+ Client chỉ có thể giao tiếp với server, không thể giao tiếp trực tiếp với client khác

+ Giao tiếp được khởi tạo khi client gửi yêu cầu và server chấp nhận

+ Kết nối bất đối xứng: yêu cầu/chấp nhận

+ Client và server đặt trên các máy khách nhau

- Cấu trúc: 3 tầng

+ Tầng client bao gồm các client

+ Tầng giữa bao gồm các quy tắc nghiệp vụ

+ Tầng CSDL chứa các thông tin

**4. Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc dữ liệu dùng chung (Share-Data)**

**-** Đặc điểm:

+ 2 kiểu thành phần: kho dữ liệu và bộ truy cập dữ liệu

* Kho dữ liệu: nơi cung cấp khả năng lưu trữ dữ liệu bền vững và tin cậy
* Bộ truy cập dữ liệu: truy cập dữ liệu trong kho dữ liệu, thực hiện tính toán và trả về kết quả

+ Bộ truy cập dữ liệu chỉ có thể giao tiếp thông qua kho dữ liệu

+ 2 biến thể:

* Bảng đen: nếu dữ liệu được cập nhật trong kho dữ liệu thì tất cả bộ truy cập dữ liệu sẽ được thông báo
* Kho dữ liệu: được cập nhật thụ động, khi dữ liệu được cập nhật thì phải thông báo cho nó

**5. Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc 3 tầng (Three tier)**

- Các client không được truy cập trực tiếp vào database

- Mềm dẻo và tin cậy

- Bao gồm các tầng:

+ Tầng đại diện: hiển thị dữ liệu trên website, nó giao tiếp với các tầng khác bằng cách gửi dữ liệu cho trình duyệt qua mạng

+ Tầng ứng dụng: còn gọi là tầng logic nghiệp vụ, nó điều khiển chức năng của ứng dụng qua các tiến trình

+ Tầng dữ liệu: lưu trữ dữ liệu, độc lập với ứng dụng và các logic nghiệp vụ

**6. Liêt kê và mô tả các bước trong thiết kế cơ sở dữ liệu**

**-** B1: xác định CSDL cần giải quyết

+ Nhằm định nghĩa các đối tượng

+ Hai yêu cầu: CSDL sẽ được sử dụng như thế nào? Và Những thông tin gì được lưu trong CSDL?

- B2: nghiên cứu hệ thống dữ liệu đã có

+ Nắm bắt nhanh hơn thực tế của hệ thống dữ liệu cần xây dựng qua các hệ thống dữ liệu đã có

- B3: thiết kế các cấu trúc dữ liệu

+ Xây dựng các đối tượng theo suuy nghĩ của mình, từ đó có thể điều chỉnh cho phù hợp

- B4: xây dựng các quan hệ

+ Xây dựng mối liên kết thông tin giữa các đối tượng

- B5: mô tả các niêm luật và ràng buộc

+ CSDL đáp ứng các niêm luật, ràng buộc sẽ có chất lượng cao

+ Bảo đảm tính an toàn, chính xác, hợp lý của CSDL

+ Bảo đảm tính chính xác và đúng đắn của dữ liệu

- B6: tạo lập các bản đặc tả thiết kế và các báo cáo

+ Thể hiện rõ nét những hình ảnh và đặc điểm của dữ liệu

- B7: thực hiện thiết kế CSDL

+ Cần đảm bảo các yêu cầu:

* Tổ chức, sắp xếp các đối tượng và quan hệ đúng với yêu cầu và ràng buộc
* Các đối tượng của CSDL đã được phân tích, chọn lọc cần được thực hiện đầy đủ
* Hình ảnh cúa các đối tượng, quan hệ phải được thể hiện phù hợp, bố trí hợp lý
* Diễn giải, ghi chú những điểm quan trọng để bản thiết kế được rõ ràng hơn

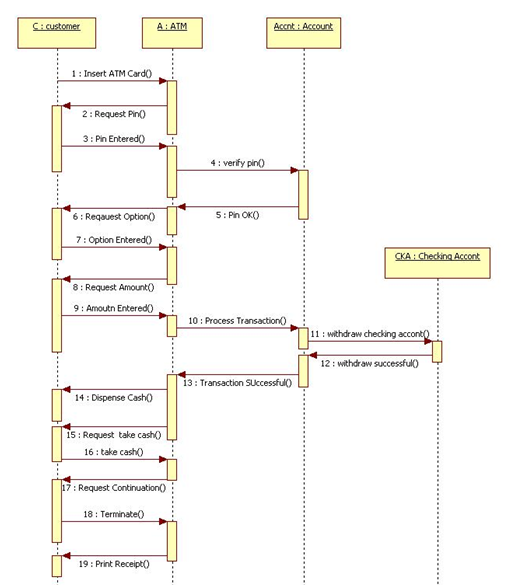
**7. Mô tả ba loại biểu đồ UML trong phân tích và thiết kế hướng đối tượng.**

**-** UML Class diagrams: mô tả các thực thể trong hệ thống, đại diện cho các đối tượng thực trong hệ thống. Mỗi thực thể là một Class, có thuộc tính và các phương thức

- UML state diagrams: mô tả các trạng thái khác nhau của từng thành phần trong hệ thống

Vd: Một sinh viên khi mới vào trương, trạng thái của đối tượng này trong hệ thống lưu trữ là active, nếu có lỗi trong quá trình active, trạng thái của đối tượng này là inactive, nếu đối tượng này tốt nghiệp xong, trạng thái chuyển về Alumni (cựu sinh viên), nếu đối tượng này bị đuổi học, trạng thái sẽ chuyển thành Expelled (bị đuổi).

- UML sequence diagrams: mô tả sự tương tác của các đối tượng tạo thành chức năng của hệ thống



**8. Cho biết các vấn đề trong thiết kế giao diện người dùng**

**-**  Luồng tương tác

- Nhìn và cảm nhận của người dùng

**Chapter 8. Design Characteristics and Metrics – Các đặc trưng và độ đo thiết kế**

**1. Mô tả hai đặc trưng cơ bản của một bản thiết kế tốt.**

**-** Bản thiết kế tốt phải đáp ứng các yêu cầu đặt ra

- 2 đăch trưng:

+ Thống nhất:

* Giao diện người dùng: hình thức và luồng logic
* Xử lý lỗi
* Báo cáo
* Giao diện hệ thống
* Trợ giúp
* Tất cả các phần thiết kế cần thực hiện với mức độ chi tiết như nhau

+ Hoàn chỉnh

* Bản thiết kế đã tính đến hết các yêu cầu của phần mềm
* Tất cả các thiết kế thực hiện cho đến khi hoàn chỉnh và cùng mức độ chi tiết

**2. Trình bày cách tính độ đo dung lượng (Volume metric) của Halstread để phân tích độ phức tạp của chương trình nguồn.**

- Nhằm phân tích độ phức tạp cảu mã nguồn chương trình

- 4 đơn vị cơ bản:

+ n1: số lượng toán tử độc lập

+ n2: số lượng toán hạng độc lập

+ N1: số lần xuất hiện của n1 toán tử

+ N2: số lần xuất hiện của n2 toán hạng

- Từ vựng: n = n1 + n2

- Độ dài chương trình: N = N1 + N2

- 4 độ đo:

+ Dung lượng: V = N \* log2(n)

(20 < V < 1000)

* Dung lượng 1 hàm không rỗng < 20
* V > 1000 🡪 hàm đó làm quá nhiều việc
* V của 1 file: 100 < V< 8000

+ Dung lượng thế: V@ = (2 + n2 @) \* log2(2 + n2 @) 🡪 CT lý tưởng

+ Mức độ cài đặt: L = V@/V

+ Công sức: E = V/L

**3. Trình bày cách tính độ phức tạp module Cyclomatic (Cyclomatic Complexity) của McCabe.**

- 1 cạnh giữa 2 nút là một chuyển điều khiển giữa 2 câu lệnh

- Độ phức tạp:

E = e –n + 2 \*p

+ e: số cạnh

+ n: số nút

+ p: số thành phần kết nối(thương p = 1)

- Độ đo vòng = số quyết định nhị phân (các câu lệnh if) + hoặc số lượng vùng khép kín + 1

- Nhược điểm: nếu số lượng vòng lớn thì rủi ro lớn, số lượng ca kiểm thử nhiều

**4. Trình bày cách tính độ phức tạp liên module của Henry-Kafura.**

**-** Mục đích đo lường tương tác giữa các module : tham số, biến toàn cục, đầu vào, đầu ra

- Fan-in: số lượng luồng thông tin khi vào 1 chương trình

- Fan-out: số lượng luồng thông tin khi ra khỏi chương trình

- Độ phức tạp của module: **Cp = ( fan-in \* fan-out ) 2**

**5. Trình bày các cấp độ ghép nối (coupling) giữa các module.**

**-** Thể hiện sự ghép nối giữa các module

- Nếu cần biết nhiều về module A để hiểu về module B thì A và B kết nối chặt chẽ

- Mục tiêu thiết kế: làm cho các module kết nối càng yếu càng tốt 🡪 các module có thể sửa đổi

- Chi phí giảm

- Ghép nối này không suy giảm trong quá trình lập trình

- Các yếu tố ảnh hưởng: kiểu kết nối giữa các module, độ phức tạp của giao diện, kiểu luồng thông tin giữa các module

- Độ phức tạp của giao diện tăng nếu tăng sự ghép nối, vì vậy cần giảm thiểu số lượng giao diện, giảm thiểu độ phức tạp của mỗi giao diện

- Giảm thiểu sự ghép nối bằng cách: đầu vào xác định của module được dùng bởi module khác, thông tin nên được trao đổi thông qua tham số

- Sự ghép nối tăng nếu: các giao diện không trực tiếp, không rõ ràng, các thành phần bên trong của module được sử dụng trực tiếp, các biến dùng chùng để giao tiếp giữa các module

**6. Trình bày các cấp độ kết dính (cohension) giữa các phần tử trong một module.**

**-** Là mức độ kết dính của các phần tử trong một module

- Mục tiêu thiết kế: thiết kế các module có mối quan hệ giữa các phần tử càng cao càng tốt

- Sự kết dính lớn 🡪 sự kết nối các module yếu

- Các cấp độ:

+ Kết dính ngẫu nhiên (nhỏ nhất)

* Bao gồm các phần không liên quan đến 1 phần khác

+ Kết dính logic

* Hàm có liên quan về logic và các phần tử dữ liệu được đặt trong cùng module

+ Kết dính thời gian:

* Hàm có liên quan về thời gian được đặt trong cùng module

+ Kết dính thủ tục:

* Nhóm hàm trong một module để đảm bảo thủ tục thực thi hợp lý

+ Kết dính giao tiếp

* Nhóm các hàm tạo ra cùng loại dữ liệu và trong cùng module

+ Kết dính trình tự

* Đầu ra của 1 phần tử là đầu vào của phần tử khác trong component

+ Kết dính chức năng(lớn nhất)

* Độ đo lý tưởng: từng tiến trình ảnh hưởng đền hiệu năng của bản thân function đó

**7. Trình bày cách tính độ đo kết dính chức năng yếu và mạnh (weak functional cohesion and strong functional cohesion) của Bieman và Ott**

**-** Các đại lượng:

+ Thẻ dữ liệu: biến,hàm trong chương trình

+ Lát cắt: tập hợp câu lệnh ảnh hưởng tới giá trị một biến quan tâm

+ Lát cắt dữ liệu: tập hợp thẻ dữ liệu

+ Thẻ dính: thẻ dữ liệu trong chương trình nằm ở nhiều hơn một lát cắt dữ liệu 🡪 kết dính các lát cắt khác nhau

+ Thẻ siêu dính: thẻ dữ liệu trong chương trình nằm trong tất cả lát cắt dữ liệu của chương trình

- 2 độ đo:

+ Kết dính chức năng yếu:

+ Kết dính chức năng mạnh:

**8. Trình bày 8 quy tắc thiết kế giao diện người dùng của Shneiderman và Plaisant.**

- Thống nhất giao diện

- Truy cập nhanh cho những người sử dụng thường xuyên

- Phản hồi thông tin

- Các hộ thoại cần kết thúc bởi kết quả nào đó

- Hướng tới các giao diện tránh lỗi, xử lý lỗi đơn giản

- Dễ dàng phục hồi sau khi thực hiện các thao tác

- Hỗ trợ thao tác bên trong

- Giảm tải nhớ

**Chapter 9. Implementation – Xây dựng chương trình**

**1. Cho biết các lỗi lập trình thường gặp**

- Lỗ hổng bộ nhớ: không được giải phóng sau khi sử dụng

- Giải phóng tài nguyên đã giải phóng

- Tham chiếu tới địa chỉ null

- Xác định định danh

- Chỉ số của mảng vượt quá kích thước mảng

- Ngoại lệ về tính toán (vd: phép chia cho 0)

- Mảng bắt đầu với chỉ số 1

- Tràn dữ liệu\

- Sử dụng & và && bất hợp lý

- Lỗi sử dụng chuỗi

- Tràn bộ đệm

**2. Liệt kê 10 thông lệ lập trình (programming practices)**

**-** Điều khiển cấu trúc: chỉ sử dụng một ít cấu trúc có cấu trúc

- Goto: sử dụng ít, chỉ khi các lựa chọn khác tồi tệ hơn

- Che dấu thông tin

- Sử dụng kiểu tự định nghĩa

- Cấu trúc lồng if else nên nhỏ

- Kích thước module: nhỏ, nên có sự kết dính thấp

- Giao diện module đơn giản

- Độ bền: xử lý các tình huống ngoại lệ

- Tác dụng phụ: tránh và nên dùng các tài liệu đặc tả (vd: dùng các biến toàn cục)

- Switch case with default

- Để trống phần catch ngoại lệ: nên sử dụng một vài trương hợp mặc định nếu có ngoại lệ xảy ra

- Để trống if, while: lập trình tồi

- Đọc các trả về: nên kiểm tra chặt chẽ

- Không nên trả về khi ở finally

- Tham số tương quan: cần kiểm tra sự tương thích của chúng với hệ điều hành

**3. Liệt kê các chuẩn lập trình Java**

**-** Các lập trình viên cần dành nhiều thời gian hơn để đọc và viết code

- Cần đọc code của chính mình và của người khác

- Sẽ dễ đọc hơn nếu mọi người trong nhóm sử dụng chung chuẩn lập trình

- Các tiêu chuẩn lập trình sẽ giúp đỡ cho các lập trình viên

- Các tiêu chuẩn lập trình chung thường về cách đặt tên biến, tên file, cách tổ chức, các câu lệnh, khai báo…

- Trong Java, có một số chuẩn lập trình:

+ Cách đặt tên biến, package:

* Tên package phải là chữ thường
* Tên loại dữ liệu nên bắt đầu bằng chữ hoa và là danh từ
* Tên biến nên là chữ thường
* Biến hoạt động với phạm vi rộng có tên dài hơn biến hoạt động trong phạm vi hẹp
* Các biến tăng trong vòng lặp nên là i, j ,k…

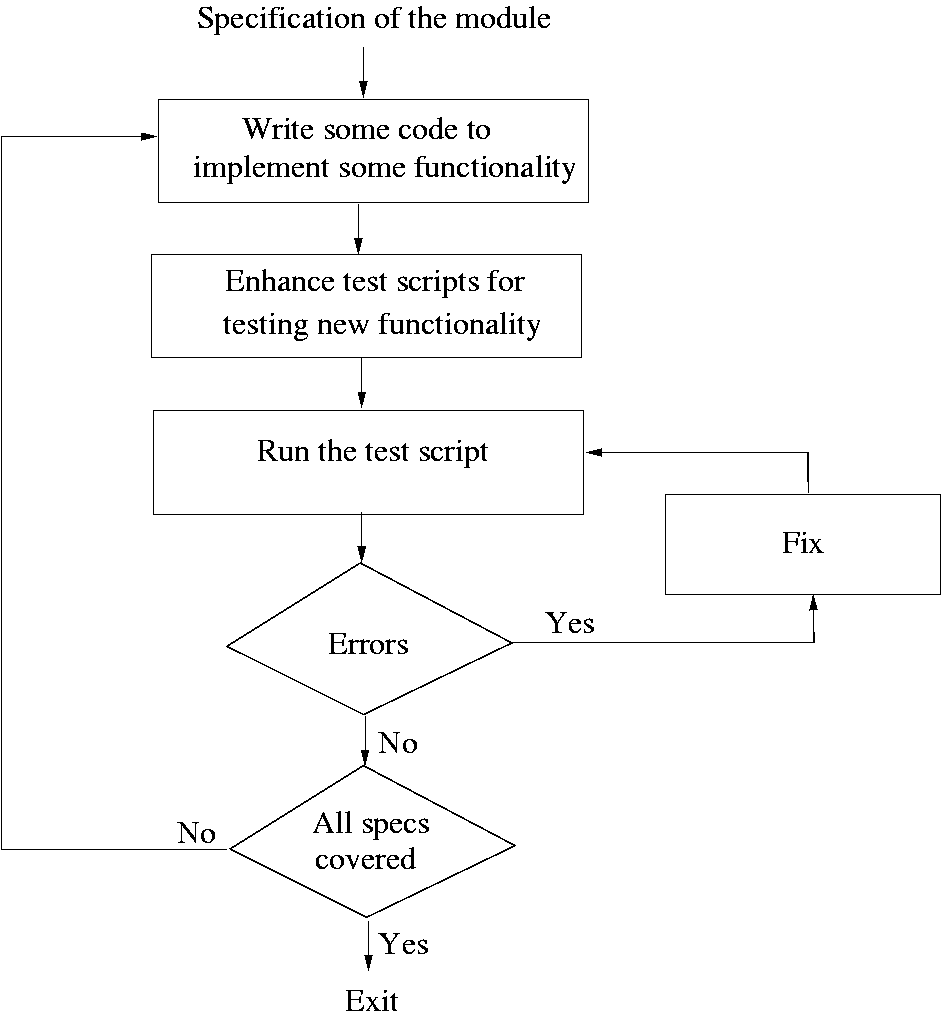
+ File

* Tên file có đuôi .java
* Mỗi file chứa 1 class và có tên giống tên file
* Chiều dài một dòng nên nhỏ hơn 80
* Tránh các ký tự đặc biệt

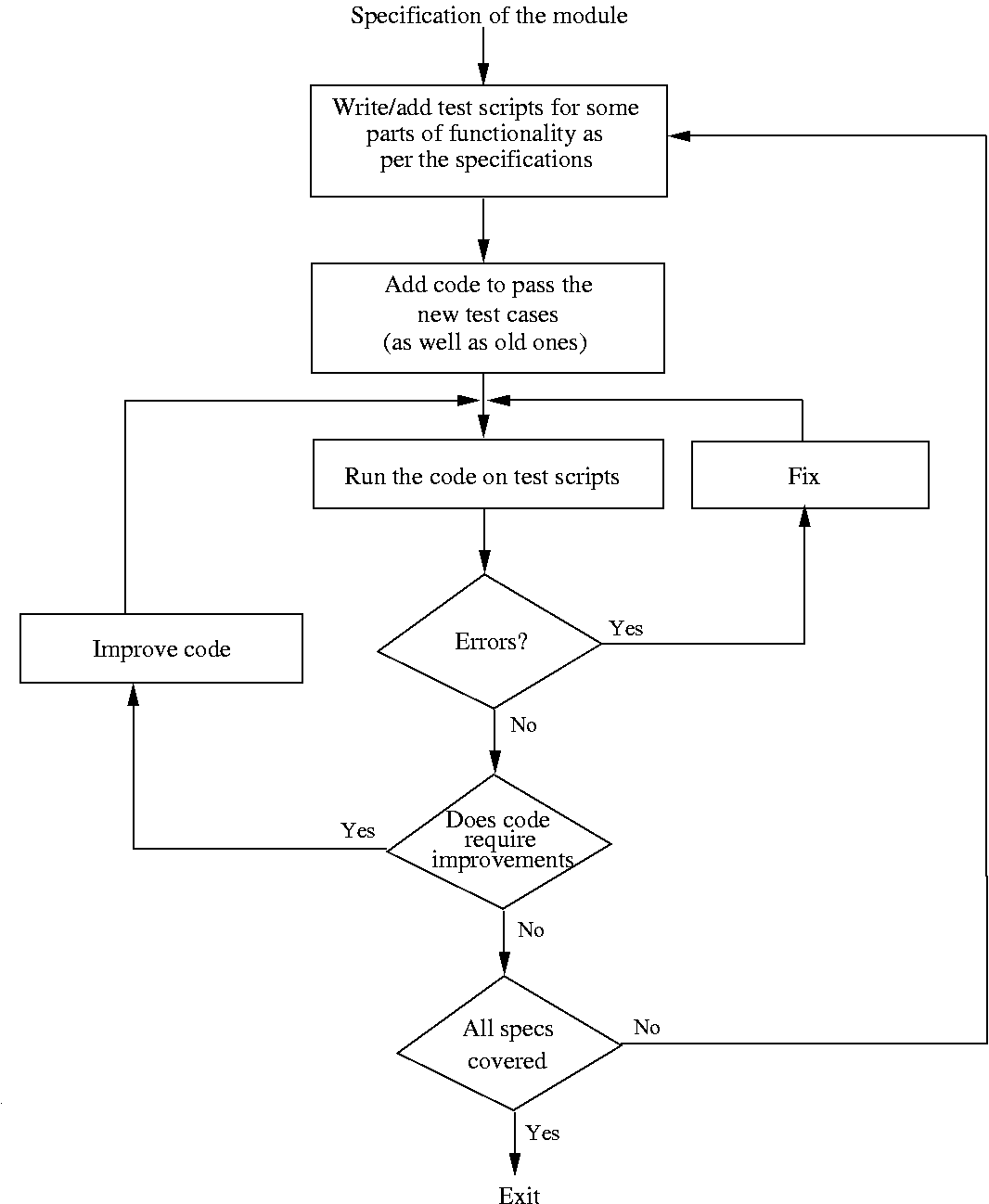
+ Câu lệnh

* Biến được khởi tạo khi khai báo trong phạm vi nhỏ nhất có thể
* Các biến liên quan được khai báo cùng nhau, không liên quan thì khai báo riêng
* Các thuộc tính trong class không được public
* Biến trong vòng lặp nên được khởi tạo trươc khi lặp
* Tránh break và continue trong vòng lặp

**4. Vẽ và mô tả biểu đồ biểu diễn tiến trình lập trình tăng dần (incremental coding process)**

****

**5. Vẽ và mô tả biểu đồ biểu diễn tiến trình lập trình hướng kiểm thử (test driven development)**

****

**6. Trình bày các đặc điểm của lập trình đôi (Pair Programming).**

**-** Đề cao việc thực hành

- Việc rà soát các chương trình được thực hiện liên tục

- Các thiết kế thuật toán logic tốt hơn

- Các điều kiện đặc biệt có khả năng xử lý tốt

- Hiệu quả chưa được đánh giá toàn diện

**7. Liệt kê 10 mục cần kiểm tra khi thanh tra chương trình**

- Định nghĩa dữ liệu phù hợp với khả năng của ngôn ngữ

- Check null trước khi sử dụng

- Tất cả biến và con trỏ phải được khởi tạo

- Chỉ số mảng không được vượt quá kích thước mảng

- Kết thúc vòng lặp: kiểm tra điều kiện, số lần lặp

- Kiểm tra phép chia cho 0

- Kiểm tra đầu ra của các biến

- Vòng lặp không chạy vào các câu lệnh như mong muốn

- Kiểm tra bảo mật

- Kiểm tra dữ liệu đầu vào

**Chapter 10. Testing - Kiểm thử**

**1. Cho biết bốn kỹ thuật phát hiện lỗi phần mềm.**

- Kiểm thử: chạy chương trình trong môi trường có kiểm soát và kiểm chứng đầu ra

- Thanh tra và rà soát: kiểm tra mã nguồn chương trình

- Phương pháp hình thức: chứng minh phần mềm hoạt động đúng

- Phân tích tĩnh: phát hiện các điều kiện dễ bị lỗi

**2. Cho biết các lớp kỹ thuật kiểm thử phần mềm.**

**-** Tiến trình

**-** Phân tích tương đương

- Phân tích giá trị biên

- Phân tích đường dẫn

**3. Cho biết 4 cấp độ kiểm thử phần mềm.**

**-** Unit test

- Functional test

- Conponent test

- System/Regression test

**4. Trình bày kỹ thuật kiểm thử phần mềm phân lớp tương đương (Equivalence Class Partitioning)**

**-** Kiểm thử hộp đen

**-** Chia dữ liệu đầu vào thành các lớp tương đương là các mục tiêu tìm ra lỗi

- Chọn phần tử đại diện cho mỗi lớp và tiến hành kiểm thử

- Xác định lớp tương đương dựa vào trực giác, kinh nghiệm và đặc tả

**5. Trình bày kỹ thuật kiểm thử phần mềm phân tích giá trị biên (Boundary Value Analysis)**

**-** Giá trị biên dễ gây lỗi

- Phân chia dữ liệu đầu vào thành các lớp tương đương

- Các ca kiểm thử bao gồm: biên, biên + 1, biên -1

**6. Trình bày kỹ thuật kiểm thử phần mềm phân tích đường dẫn (Path Analysis)**

**-** Kiểm thử hộp trắng

- Các bước:

+ Tạo đồ thị luồng điều khiển của chương trình: mỗi nút ứng với 1 câu lệnh, 1 cạnh giữa 2 nút là chuyển điều khiển giữa 2 câu lệnh

+ Xác định đường dẫn cần kiểm thử: đường đi từ bước xuất phát đến bước kết thúc

- Mục tiêu:

+ Thử tất cả các đường dẫn logic: tập các đường đi từ điểm xuất phát đến điểm kết thúc

+ Phủ tất cả các đường dẫn độc lập, các đường dẫn logic cắt nhau ít nhất 1 cạnh

+ Phủ được các nhánh, các cạnh đi ra từ nút điều kiện

+ Phủ được các câu lệnh

**Chapter 11. Configuation Management, Integration, and Builds - Quản lý cấu hình, tích hợp và xây dựng chương trình chạy**

**1. Liệt kê 10 sản phẩm (artifacts) được quản lý cấu hình**.

- Đặc tả yêu cầu

- Tài liệu thiết kế

- Source Code

- Các kịch bản kiểm thử

- Code đã được thực thi

- Các bảng trong CSDL

- Dữ liệu khởi tạo

- Các cuộc gọi về các vấn đề của sản phẩm với khách hàng

- Sửa chữa khiếm khuyết sản phẩm

- Tài liệu sử dụng cho người dùng.

**2. Cho biết vai trò của quản lý cấu hình và các chức năng cơ bản của quản lý cấu hình.**

**-** Vai trò:

+ Đáp ứng các mục tiêu cơ bản của dự án khi chuyển giao sản phẩm cho khách hàng

* Danh sách các file trong sản phẩm
* Phiên bản của các file
* Kết hợp trong sản phẩm như thế nào

**-** Chức năng:

+ Cung cấp trạng thái của chương trình

+ Phiên bản cuối cùng của chương trình

+ Khôi phục thay đổi, trở về phiên bản nào đó

+ Tránh thay đổi không được phép

+ Tập hợp tài nguyên, thông tin khác trong hệ thống

**3. Cho biết mô hình lưu trữ và truy cập các thành phần (Parts Storage and Access Model) trong quản lý cấu hình và các tính năng chính của mô hình này.**

***Parts***

***Control***

***Parts Database***

**-** Chức năng cơ bản: tạo mới và xóa 1 phần

- Chức năng truy cập: xem, sửa, quay lại 1 phần

- Chức năng điều khiển và dịch vụ:

+ Nhập, xuất 1 phần

+ Danh sách 1 phần

+ Thay đổi tên, phiên bản… của 1 phần

+ Gộp vào 1 phần

**Chapter 12. Software Support and Maintenance - Hỗ trợ và bảo trì phần mềm**

**1. Cho biết các công việc cần thực hiện trong hỗ trợ khắc phục khiếm khuyết phần mềm (defect support) và cho biết xu hướng thay đổi của số lượng khiếm khuyết theo thời gian sau khi chuyển giao phần mềm.**

- Các công việc cần thực hiện:

+ Dự báo về số lượng vấn đề

+ Ước lượng, lập kế hoạch cho các nguồn lực hỗ trợ

+ Đào tạo xây dựng đội ngũ hỗ trợ

+ Báo cáo, theo dõi khiếm khuyết

+ Xác định, sửa lỗi và công bố

- Xu hướng thay đổi:

+ Trong giai đoạn mới công bố sản phẩm, nhiều vấn đề được phát hiện và báo cáo

+ Khi số lượng vấn đề cuối cùng được giảm đi thì việc phát hiện bản chất của chúng lại trở nên khó khăn hơn

**2. Trình bày hai đặc điểm của công việc hỗ trợ khách hàng (Customer Support)**

**-** Hỗ trợ khách hàng là không miễn phí, có thể chiếm đến 18% chi phí

- Hỗ trợ khách hàng không mãi mãi

+ Nhiều sản phẩm chỉ qua 1 vài bản release

+ Mỗi sản phẩm release chỉ hỗ trợ trong 1 khoảng thời gian nhất định

+ Người dùng cần cập nhật phiên bản mới ngay khi có thể

**3. Cho biết các công việc thực hiện trong giai đoạn trước khi kết thúc việc hỗ trợ phần mềm, hay còn gọi là giai đoạn mặt trời lặn (product sunset period).**

**-** Dừng bổ sung các tính năng và hoàn thiện

- Chỉ sửa lỗi các vấn đề thực sự nghiêm trọng

- Thông báo sản phẩm thay thế mới

- Khuyến khích khách hàng mới và khách hàng đã có sử dụng sản phẩm mới

- Thông báo cho người dùng phiên bản cũ về ngày kết thúc hỗ trợ

- Cung cấp thông tin về nhà cung cấp sẵn sang hỗ trợ sản phẩm cũ nếu khách hàng tiếp tục sử dụng sản phẩm cũ

- Kết thúc hỗ trợ sản phẩm cho khách hàng và rút sản phẩm khỏi thị trường

**4. Trình bày các mục cần có trong một mẫu yêu cầu thay đổi trong giai đoạn bảo trì phần mềm.**

**-** Yêu cầu thay đổi

- Tên người yêu cầu

- Sự ưu tiên của yêu cầu

- Ngày yêu cầu

- Ngày nhận/ Từ chối

- Ngày thực thi / hoàn thành

- Mô tả ngắn gọn yêu cầu

- Vùng bị ảnh hưởng bởi yêu cầu thay đổi

- Ước tính công sức

- Chi phí bảo trì

**Chapter 13. Software Project Management – Quản lý dự án phần mềm**

**1. Cho biết các giai đoạn và các mục tiêu của tiến trình quản lý dự án phần mềm.**

- Các giai đoạn:

+ Lập kế hoạch

+ Tổ chức

+ Giám sát

+ Điều chỉnh

- Mục tiêu:

+ Kết quả cuối cùng của dự án làm hài long người dùng/ khách hàng

+ Tất cả mong muốn và các thành phần cần thiết của dự án được quản lý

+ Mục tiêu cuối cùng và lịch trình của dự án được quản lý

+ Tất cả thành viên trong nhóm được điều hành và luôn trong trạng thái làm việc tốt nhất

+ Yêu cầu công cụ và các nguồn lực khác được tạo sẵn và sử dụng có hiệu quả

**2. Cho biết các bước của giai đoạn lập kế hoạch trong tiến trình quản lý dự án phần mềm.**

- B1: Ước lượng

+ Công sức, lịch trình và các nguồn lực cần thiết

- B2: Xác định và thiết lập các mục tiêu cho dự án

- B3: Xác định phân bổ nguồn lực: con người, tiến trình, công cụ và có sở vật chất

- B3: Nhận định và phân tích rủi ro của dự án

**3. Cho biết các công việc được thực hiện trong giai đoạn tổ chức trong tiến trình quản lý dự án phần mềm.**

**-** Cấu trúc tổ chức dự án cần phải được thiết kế

- Việc tuyển dụng nhân lực cần bắt đầu và hoàn thành cùng với việc thu thập các nguồn lực khác

- Bất kỳ yêu cầu về giáo dục và đào tạo nài cũng cần hoàn thiện

- Thiết lập cơ chế theo dõi

**4. Trình bày chương trình khung để xây dựng cấu trúc phân rã công việc (Work breakdown Structure) trong việc ước lượng công sức hoàn thành dự án.**

**-** Ước lượng công sức của một dự án hoàn thành bởi các sản phẩm, các nhiệm vụ cần thiết của dự án và nguồn lực yêu cầu để thực hiện các nhiệm vụ

- Kiểm tra và quyết định các sản phẩm chuyển giao bên ngoài của dự án

- Xác định các bước và nhiệm vụ cần thiết để sản xuất ra từng sản phẩm, bao gồm các nhiệm vụ được yêu cầu để sản xuất bất kỳ sản phẩm nội bộ trung gian nào

- Trình tự các nhiệm vụ để chọn ra các nhiệm vụ có thể thực hiện song song

- Ước lượng hoàn thành của mỗi nhiệm vụ

- Ước lượng hoàn thành sản phẩm của mỗi cá nhân

- Tính toán thời gian yêu cầu hoàn thành của mối nhiệm vụ

- Đối với mỗi sản phẩm, cần đưa ra thời gian hoàn thành, nguồn lực và tài nguyên cho mỗi nhiệm vụ.