NỘI DUNG ÔN TẬP

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

**A. LÝ THUYẾT PHẦN**

Phần 1: GIỚI THIỆU CNPM, QUY TRÌNH, PHÂN TÍCH VÀ ĐẶC TẢ YÊU CẦU

***1. Quy trình phát triển phần mềm có những hoạt động chính nào? Hãy trình bày những hoạt động này.***

* Quy trình phát triển phần mềm là một tập các hoạt động có liên quan với nhau và được thực hiện theo một trật tự nhất định, nhằm tạo ra sản phẩm phần mềm chất lượng cao.
* Quy trình phát triển phần mềm có những hoạt động chính : 4 hoạt động chính.

1. Đặc tả yêu cầu phần mềm: Định nghĩa các chức năng chính của phần mềm và các ràng buộc xung quanh nó.
2. Phát triển phần mềm: Phần mềm được thiết kế và xây dựng.
3. Xác minh và thẩm định phần mềm: Phần mềm được kiểm tra xemcó đúng theo đặc tả và có đáp ứng theo nhu cầu của người dùng không.
4. Sự tiến hóa của phần mềm: Phần mềm được cập nhật/chỉnh sửa đáp ứng theo những thay đổi yêu cầu của khách hàng.

***2. Mô hình phát triển phần mềm Thác nước trải qua những giai đoạn nào? Hãy trình bày về những giai đoạn này.***

Mô hình phát triển phần mềm Thác nước trải qua những giai đoạn: 5 giai đoạn

1. **Phân tích các yêu cầu và định nghĩa:** Các kỹ sư IT sẽ phải thu thập tấtcả các yêu cầu cần thiết, sau đó sẽ hội ý để hiểu đúng những yêu cầu. Cuối cùng, các kỹ sư sẽ cần phải tìm hiểu xem những yêu cầu này có phù hợp để kiểm thử ở các bước tiếp theo hay không.
2. **Thiết kế phần mềm và hệ thống:** Từ những yêu cầu được xác định trong bước 1, nhóm dự án tạo ra thiết kế cho sản phẩm để đáp ứng tất cảcác yêu cầu đó, bao gồm cả thiết kế phần cứng, thiết kế phần mềm, ngôn ngữ lập trình, lưu trữ dữ liệu. Đây đồng thời cũng là phần giúp bạnxác định dự án sẽ hữu ích thế nào đối với người dùng. Nếu bước này gặp vấn đề thì rất có thể phải quay lại bước 1 để thực hiện lại.
3. **Hiện thực và kiểm thử các đơn vị:** Khi hệ thống đã được thiết kế đầy đủ và cụ thể, các module chức năng của sản phẩm sẽ được thực hiện trong giai đoạn này để đáp ứng các tiêu chuẩn đã thực hiện ở bước trước. Đây là giai đoạn mà các nhiệm vụ công việc được thảo luận ở bước 2 được tiến hành và cũng là giai đoạn mà đội ngũ lập trình sẽ là nguồn lực chủ yếu được sử dụng. Ở giai đoạn kiểm thử, thường sẽ là công việc của đội ngũ QA và tester nhằm tìm kiếm và báo cáo các lỗi trong hệ thống cần được xử lý. Việc này bao gồm tất cả các hoạt động kiểm thử tính năng và phi tính năng. Đây là giai đoạn cực kỳ quan trọngmà nhóm không được phép mắc sai lầm nhằm đảm bảo hệ thống được kiểm tra đầy đủ, các mục tiêu thiết kế và chức năng người dùng yêu cầu được đáp ứng và các nhu cầu kinh doanh được giải quyết.
4. **Tích hợp và kiểm thử hệ thống:** Đây là giai đoạn mà sản phẩm được triển khai vào môi trường mà người dùng có thể bắt đầu sử dụng được. Hay nói cách khác là giai đoạn mà sản phẩm thực sự đi vào hoạt động. Trong giai đoạn này, nhóm dự án cần đảm bảo các yếu tố như: môi trường đang hoạt động, không có lỗi trên server, các tiêu chí test đã được đáp ứng hoặc kiểm tra lại môi trường sau khi ứng dụng được triển khai để đảm bảo sản phẩm không gặp vấn đề….
5. **Vận hành và bảo trì:** Đây là giai đoạn cuối cùng của quá trình, trong đó nhóm dự án tập trung giải quyết các vấn đề của khách hàng. Trong các dự án phần mềm, đây thường là giai đoạn các bản được phát hành để cập nhật và sửa lỗi.

***3. Trình bày ý tưởng chung của mô hình phát triển phần mềm Tiến hóa (Evolutionary development).***

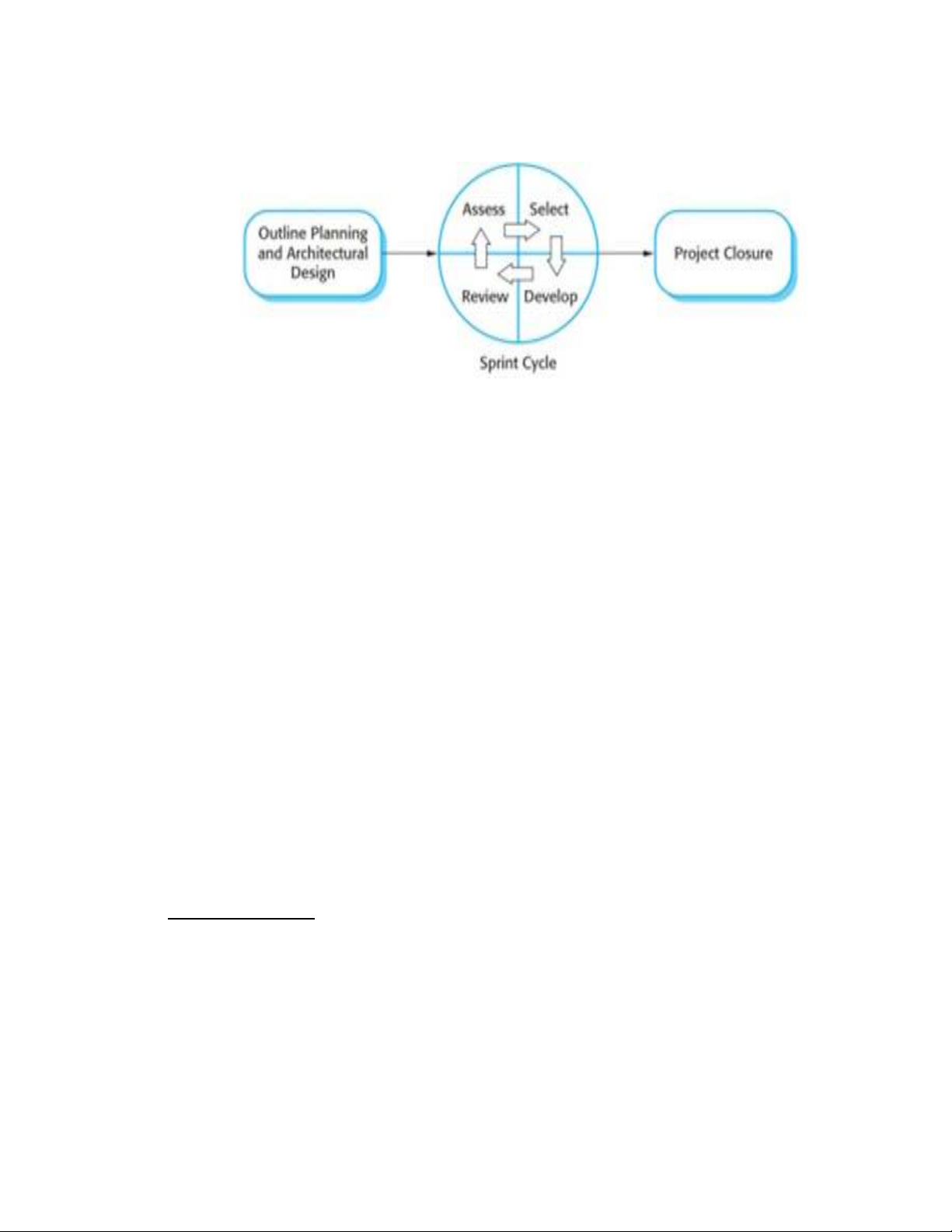
* Còn gọi là mô hình phát triển Tăng dần (Incremental development).
* Các mô hình phát triển phần mềm Tiến hóa dựa trên ý tưởng chung:
* Phát triển một phiên bản đầu tiên (initial version)
* Lấy phản hồi từ người dùng.
* Nâng cấp (tiến hóa) phần mềm thông qua các phiên bản cho đến khi hệ thống được chấp thuận.
* Các hoạt động trong quy trình (Đặc tả yêu cầu, phát triển, thẩm định) một cách đan xen nhau để nhận thông tin phản hồi một cách nhanh chóng (thay vì phân tách thành các giai đoạn riêng biệt).

***4. Mô hình phát triển phần mềm Dựa trên thành phần (Component-based software engineering) trải qua những giai đoạn nào? Hãy trình bày về những giai đoạn này.***

* Ý tưởng: trên cơ sở sử dụng lại các thiết kế, mã nguồn đã có sẵn (tương tựvới cái cần để xây dựng hệ thống), chỉnh sửa, tích hợp vào hệ thống mới.
* Giai đoạn Đặc tả yêu cầu và Thẩm định hệ thống tương tự như các quy trình khác.
* Các giai đoạn trung gian trong mô hình này thì đặc biệt hơn so với các quy trình khác:
* Phân tích thành phần (Component analysis)
* Với bản đặc tả yêu cầu, nhà phát triển sẽ tìm kiếm các thành phần có sẵn và phân tích xem chúng có thể đáp ứng được thế nàotrong việc xây dựng hệ thống mới.
* Thông thường, các thành phần tìm thấy sẽ không đáp ứng 100%yêu cầu.
* Chỉnh sửa yêu cầu (Requirement modification)
* Ở giai đoạn này, các yêu cầu sẽ được phân tích dựa trên thông tin về các thành phần (component) đã tìm thấy.
* Các yêu cầu có thể được chỉnh sửa lại cho phù hợp với các thànhphần tìm thấy.
* Nếu việc chỉnh sửa yêu cầu không thể thực hiện được, trở lại bước tìm kiếm các thành phần -> tìm giải pháp khác.
* Thiết kế hệ thống với các thành phần tái sử dụng (System design with reuse) – Thiết kế kiến trúc, nền tảng (framework) cho hệ thống với các thành phần tái sử dụng.
* Phát triển và tích hợp (Development and Integration).
* Các thành phần tái sử dụng sẽ được tích hợp vào hệ thống mới.
* Phát triển các thành phần mới.
* Thông thường, có 3 kiểu thành phần có thể tái sử dụng (theo tổnghợp của Ian Sommerville).
* Web servers: Cho phép triệu gọi từ xa thông qua Web.
* Dịch vụ vé máy bay, dự báo thời tiết, giá vàng, kiểm lỗi chính tả…
* Các thư viện/plugins:oThư viện xuất ra file Excel, PDF…
* Plugin hiển thị sliders (wordpress), comment facebook trên website,…
* Hệ thống phần mềm hoàn chỉnh:
* Hệ thống CT-scan xuất ra file ảnh, dùng import vào hệthống quản lý y khoa

***5. Khung quy trình phát triển phần mềm Scrum gồm những giai đoạn nào? gồm những sự kiện nào? Hãy trình bày về các giai đoạn và sự kiện này.***

* **Khung quy trình Scrum (Scrum process framework)**



* Giai đoạn 1: Lên kế hoạch sơ bộ, xác định các mục tiêu của dự án và thiếtkế kiến trúc hệ thống.
* Giai đoạn 2: Các vòng Sprint, mỗi Sprint sẽ tạo ra một phần tăng trưởng.
* Giai đoạn 3: Đóng dự án, tổng hợp, hoàn tất các tài liệu dự án, rút bài họckinh nghiệm...

**Các sự kiện trong Scrum:** [**https://hocvienagile.com/agipedia/tong-quan-ve-scrum/**](https://hocvienagile.com/agipedia/tong-quan-ve-scrum/)

* Sprint (một giai đoạn, 2-4 tuần).
* Sprint planning (cuộc họp lên kế hoạch cho Sprint).
* Daily Scrum (họp hằng ngày, Stand-up meeting).
* Sprint Review (họp demo sản phẩm).
* Sprint Retrospective (họp rút kinh nghiệm).

***6. Scrum master và Product owner đóng vai trò như thế nào trong khung quy trình phát triển phần mềm Scrum?***

Scrum master:là một vai trò then chốt giúp nhóm Scrum làm việc hiệu quả bằng cách tuân thủ nguyên lý, các kỹ thuật và quy tắc của Scrum.

Scrum Master không phải là người quản lý của Nhóm mà là một lãnh đạo theo phong cách phục vụ (Servant Leader). Scrum Master làm tất cả những gì trong thẩm quyền phục vụ Product Owner, Nhóm Phát triển, và Tổ chức đi đến thành công.

**Công việc cụ thể**

Công việc đầu tiên trong ngày thường làm là xem lại danh sách công việc của mình, những cái nào đã hoàn thành, cái nào cần được xử lý trong ngày, sau đó kiểm tra email xem có vấn đề hay yêu cầu nào cần xử lý, rồi đưa vào danh sách công việc.Tiếp theo, xem qua các dự án mà mình đang quản lý để biết tiến độ của từng bạn như thế nào.

Nếu bạn nào gặp vướng mắc và cần hỗ trợ, sẽ tìm người có khả năng supportbạn đó nhanh nhất.

Sau khi mọi thứ đã được giải quyết ổn thỏa, anh sẽ bắt đầu xử lý các công việc khác trong danh sách công việc của anh. Công việc ở đây có nhiều loại, ví dụ như thảo luận với khách hàng về các vấn đề phát sinh trên hệ thống của họ, meeting để cập nhật cho khách hàng về tiến độ của dự án hoặc thảo luận với khách hàng về Sprint tiếp theo…

Product Owner:là một trong ba vai trò trong nhóm Scrum. Vai trò này chịu trách nhiệm tối ưu hóa lợi nhuận trên đầu tư (ROI – Return On Investment) thông qua việc quyết định các tính năng của sản phẩm, đánh giá và sắp xếp độ ưu tiên của từng hạng mục, những hạng mục có độ ưu tiên cao thì sẽ được đưa vào phát triển trước, những hạng mục có độ ưu tiên thấp hơn thì sẽ được phát triển sau. Product Owner thường khác với một Giám đốc Sản phẩm truyền thống ở chỗ đó là Product Owner tham gia tích cực vào quá trình phát triển sản phẩm, thay vì chỉ quản lý và ủy quyền cho những người khác thực hiện các quyết định liên quan đến sản phẩm.

* Tìm hiểu, phân tích và đưa ra các tính năng mong muốn trong Product Backlog
* Đánh giá các hạng mục trong Product Backlog để điều chỉnh tiến độ dự án phù hợp
* Tối ưu hóa lợi nhuận trên vốn đầu tư (ROI)
* Đảm bảo tính minh bạch của Product Backlog
* Đưa đầy đủ thông tin đến Nhóm Phát triển
* Theo dõi tiến độ của sản phẩm

***7. Yêu cầu phần mềm là gì? Hãy phân biệt và cho ví dụ thể hiện sự khác nhau giữa Yêu cầu chức năng và Yêu cầu phi chức năng.***

* Yêu cầu phần mềm (Software requirements) là những mô tả về những cái mà hệ thống cần phải làm:
* Những chức năng (dịch vụ) mà hệ thống phải cung cấp,–
* Những ràng buộc đối với hệ thống.
* Yêu cầu phần mềm phản ánh nhu cầu (mong muốn) của khách hàng với hệthống.
* Quy trình thu thập, phân tích, viết tài liệu đặc tả và thẩm định các yêu cầu phần mềm được gọi là quy trình kỹ nghệ yêu cầu phần mềm (Software requirements engineering - RE).

Yêu cầu chức năng và phi chức năng (Functional & non-Functional req)

* **Yêu cầu chức năng** (Functional requirements) là những phát biểu về những chứcnăng (dịch vụ) mà hệ thống cần phải cung cấp.
* Trong một số trường hợp, nó cũng phát biểu cụ thể những cái hệ thống sẽkhông cung cấp.
* Ví dụ: – Người dùng có thể tìm kiếm sách theo mã sách, tiêu đề hoặc tên tácgiả. – Hệ thống có thể xuất báo cáo thống kê số lượt mượn sách trong mộttháng.
* **Yêu cầu phi chức năng** (non-functional requirements) là những ràng buộc(constraints) đối với các chức năng (dịch vụ) mà hệ thống cung cấp.
* Các ràng buộc như:
* về thời gian
* về hiệu năng
* về quy trình phát triển phần mềm…
* Yêu cầu phi chức năng thường áp dụng lên toàn hệ thống hơn là lên từng chứcnăng riêng lẻ.
* Ví dụ:
* Hệ thống phải dễ sử dụng –
* Hệ thống phải hoạt động tốt trên Firefox x.x, Chrome x.x
* Hệ thống phải phản hồi trong vòng < 1s đối với tất cả các yêu cầu tìm kiếmsách.
* **Yêu cầu về sản phẩm (Product requirements)**
* Gồm những ràng buộc về chức năng của sản phẩm phần mềm.

**Ví dụ:**

* Yêu cầu về hiệu năng (Performance requirements) • Hệ thống chạy nhanh thế nào? Yêu cầu tiêu tốn RAM bao nhiêu?-
* Yêu cầu về độ tin cậy (Reliability requirements) • Tỉ lệ lỗi? Thời gian hệ thống ngừng hoạt động?-
* Yêu cầu về bảo mật (Security requirements)
* Yêu cầu về tính khả dụng (Usability requirements).
* **Yêu cầu về tổ chức (Organizational requirements)**
* Liên quan đến các chính sách, thủ tục, các yếu tố thuộc về tổ chức phíakhách hàng và phía nhà phát triển.

**Ví dụ:**

* Yêu cầu về quy trình phát triển phần mềm (Development processrequirements): chỉ ra ngôn ngữ lập trình, mô hình phát triển phầnmềm sẽ sử dụng…
* Nơi làm việc phải được bảo mật (không dùng USB, laptop...)
* **Các yêu cầu từ các yếu tố bên ngoài (External requirements)**
* Liên quan đến các yếu tố bên ngoài hệ thống, ngoài quy trình phát triểnphần mềm.
* **Ví dụ:**
* Yêu cầu về luật pháp (Legislative requirements) cần phải tuân thủ đểhệ thống hoạt động đúng pháp luật.
* Yêu cầu về đạo đức (Ethical requirements) để đảm bảo hệ thống được chấp nhận bởi người dùng và công chúng.
* Yêu cầu liên quan đến các điều khoản, quy định (ví dụ quy định của ngân hàng trung tâm…)

***8. Việc mô tả yêu cầu phần mềm có thể chia thành hai mức độ: Yêu cầu ngườidùng (User requirements) và Yêu cầu hệ thống (System requirements). Hãy phân biệt và cho ví dụ thể hiện sự khác nhau giữa hai mức độ này.***

* Việc mô tả các yêu cầu có nhiều cấp độ, có thể chia thành 2 mức:
* Yêu cầu người dùng (User requirements): những mô tả yêu cầu mức cao, trừu tượng (high-level abstract requirements).
* Yêu cầu hệ thống ( System requirements ): những mô tả yêu cầu mức chi tiết (detailed description of what the system should do).

**User requirements:** Là những phát biểu tổng quan, bằng ngôn ngữ tự nhiên (hoặc được bổ sung bởi một số biểu đồ) về những dịch vụ (chức năng) mà hệ thống cần cung cấp cho người dung và các ràng buộc đi kèm.

**System requirements:**

* Là những mô tả chi tiết hơn về các chức năng và ràng buộc màhệ thống cần cung cấp.
* Nó mô tả chính xác những hạng mục nào cần phải làm.
* Nó có thể là một phần trong bản hợp đồng giữa khách hàng và nhà phát triển.
* **Ví dụ yêu cầu người dùng**: Hệ thống cho phép thêm một sản phẩmmới.
* **Ví dụ yêu cầu hệ thống:**
* Hệ thống cho phép thêm sản phẩm mới gồm các trường thông tin: tên sản phẩm, giá, kích thước...
* Hệ thống tự động tạo mã sản phẩm theo mã danh mục sảnphẩm).

***9. Quy trình kỹ nghệ yêu cầu (Requirements engineering process) bao gồm những hoạt động chính nào? Hãy trình bày về những hoạt động này.***

* Quy trình kỹ nghệ yêu cầu bao gồm các hoạt động nhằm tạo ra bản đặc tả yêu cầu chất lượng.
* Quy trình thường gồm 3 hoạt động:
* Thu thập & Phân tích yêu cầu.
* Đặc tả yêu cầu.
* Thẩm định yêu cầu.
* **Hoạt động 1:** Thu thập và Phân tích yêu cầu
* Kết thúc giai đoạn này, người phân tích (analyst) sẽ cơ bản hiểu được hệ thống cần phải như thế nào:
* Có những chức năng gì, các ràng buộc, các dữ liệu vào và kết quả đầu ra của hệ thống.
* Quá trình này thường liên quan tới rất nhiều người, gọi là các bên liên quan (stackholder).
  + - Khách hàng, người dùng cuối, người phân tích, người quảnlý...
* Trong quá trình phân tích yêu cầu, nhà phát triển (analyst) và phíakhách hàng, người dùng cuối (clients, users) sẽ có một chuỗi các cuộc họp:
* Các cuộc họp ban đầu, khách hàng và người dùng cuối sẽ môtả về công việc của họ, môi trường của họ, các quy trình...
* Người phân tích sẽ lắng nghe, ghi chép, thu thập các thôngtin, biểu mẫu... Một khi đã hiểu được ở mức độ cơ bản, người phân tích sẽ viết các tài liệu yêu cầu người dùng, một số mô hình, một số phác thảo 🡪để làm rõ những điểm chưa hiểu, hoặc để xác nhận lại với khách hàng xem có đúng ý như vậy không.
* **Hoạt động 2:** Đặc tả yêu cầu
* Sau khi phân tích yêu cầu, người phân tích đã hiểu được về hệ thốngcần phải làm gì.
* Đặc tả yêu cầu là việc ghi vào tài liệu những mô tả về hệ thống (tàiliệu đặc tả yêu cầu – SRS – Software Requirement Specification)
* Tại hoạt động này, những vấn đề như: giao diện, ngôn ngữ sử dụng,công cụ... cũng được bàn tới.
* **Hoạt động 3:** Thẩm định yêu cầu
  + Thẩm định yêu cầu tập trung vào việc đảm bảo bản đặc tả yêu cầutrên là đúng với những yêu cầu/mong muốn/mục tiêu của kháchhàng về hệ thống cần xây dựng.
  + Kết quả của hoạt động này là một tài liệu đặc tả yêu cầu đã đượcthẩm định (validated SRS), một bản đặc tả yêu cầu chất lượng tốt.
  + Quy trình kỹ nghệ yêu cầu thường không diễn ra theo một hướngtuyến tính (linear sequence) mà thường:
* Chồng lấn lên nhau
* Có thông tin phản hồi và lặp lại.

***10. Trong quy trình kỹ nghệ yêu cầu thường gặp phải những vấn đề trở ngại nào?***

* Các bên liên quan không thực sự biết mình muốn gì.
* Các bên liên quan diễn đạt yêu cầu theo thuật ngữ trong lĩnh vực của họ.
* Giữa chính các bên liên quan cũng có xung đột về mục tiêu, mong muốn vềhệ thống cần xây dựng.
* Những yếu tố về tổ chức, pháp luật, đạo đức...ảnh hưởng đến yêu cầu hệthống.
* Thay đổi yêu cầu
* Môi trường kinh doanh thay đổi, nghiệp vụ thay đổi, công nghệ thayđổi...
* Tại thời điểm thu thập yêu cầu, khách hàng chưa nghĩ hết những điều họ muốn.
* Phải sắp xếp các cuộc gặp để trao đổi -> không phải lúc nào các bên cũngsẵn sàng.

PHẦN 2: THIẾT KẾ PHẦN MỀM, LẬP TRÌNH

***11. Kiến trúc phần mềm là gì? Kiến trúc phần mềm có ảnh hưởng gì đến chất lượng của sản phẩm phần mềm? Nêu 3 thuộc tính chất lượng bị ảnh hưởng bởi kiến trúc phần mềm và giải thích vì sao.***

* Kiến trúc phần mềm: Kiến trúc phần mềm của một chương trình máy tính hay một hệ thống tính toán là cấu trúc của các thành phần trong hệ thống đó.Kiến trúc phần mềm bao gồm các phần tử phần mềm, cácthuộc tính và mối quan hệ giữa chúng
* Kiến trúc phần mềm cung cấp một cái nhìn tổng quan về các thành phần con cấu thành nên một hệ thống và cách chúng tương tác, phối hợp với nhau.

Kiến trúc phần mềm cung cấp các góc nhìn khác nhau về hệ thống cần xây dựng:

+ Góc nhìn thể hiện các thành phần (component) và mối liênhệ giữa chúng (về khía cạnh logic)

+ Góc nhìn về sự phân bố các thành phần lên các tài nguyên hệ thống (về khía cạnh vật lý)

+ Góc nhìn chi tiết về các mô-dun và mối liên hệ giữa chúng.

* Performance: Kiến trúc phần mềm nên được thiết kế để thực thi các tác vụ quan trọng với số lượng nhỏ các component mà chúng được đặt trong cùng một máy tính thay vì phân tán trên mạng . Sử dụng lượng lớn các component trong kiến trúc phần mềm có thể làm giảm số các giao tiếp giữa các component, và làm tăng performance
* Security: Với yêu cầu bảo mật, một kiến trúc nhiều lớp nên được sử dụng. Tài nguyên quan trọng cần được bảo vệ sẽ được đặt trong những lớp trong cùng được bảo mật ở cấp độ cao.
* Safety: kiến trúc nên được thiết kế đê các chức năng liên quan đến safety nằm trong cùng 1 component hay trong lượng nhỏ component. Điều này giúp giảm chi phí và vấn đề khi thực hiện những việc như tắt hệ thống khi có lỗi xảy ra
* Availibility: kiến trúc nên được thiết kế bao gồm những component dự phòng để có thể thay thế và cập nhật component mà không làm ngưng hệ thống
* Maintainability: kiến trúc nên được thiết kế để có thể sẵn sàng thay đổi được

***12. Vì sao nên sử dụng các kiến trúc có sẵn khi thiết kế phần mềm? Hãy trình bày ý tưởng và giải thích các thành phần cũng như quan hệ giữa chúng của một kiểu kiến trúc mà bạn biết. Lợi ích khi sử dụng kiểu kiếntrúc này là gì?***

* Vì những kiểu kiến trúc có sẵn đã được kiểm định, nghiên cứu, ứng dụngthành công qua các thế hệ đi trước. Việc áp dụng các kiểu kiến trúc có sẵn giúp ta đánh giá được ưu nhược điểm của các kiểu kiến trúc và từ nhu cầu của mình mà lựa chọn phù hợp để quá trình sản xuất phần mềm được tốt nhất.
* VD: Mô hình MVC

Mô tả: Tách phần presentation và interaction từ hệ thống data. Hệ thống được tổ chức thành 3 phần tương tác qua lại lẫn nhau. Model: Đây có thể là cơ sở dữ liệu, file hay một đối tượng đơn giản. View: View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác.Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được. Controller: Một controller bao gồm cả Model lẫn View. Nó nhận input và thực hiện các tương tác.

Được sử dụng khi: có nhiều cách để xem và tương tác với dữ liệu và cũng được sử dụng khi không rõ về những yêu cầu về tương tác và trình bày dữ liệu trong tương lai.

Ưu điểm:

* Hỗ trợ quá trình phát triển nhanh chóng: Với đặc điểm hoạt động độc lập của từng thành phần, các lập trình viên có thể làm việc đồng thời trên từng bộ phận khác nhau của mô hình này. MVC giúp bạn tiết kiệm rất nhiều thời gian.
* Khả năng cung cấp đồng thời nhiều khung View: Với mô hình MVC, bạn có thể tạo rađồng thời nhiều khung View cho Model.
* Hỗ trợ các kỹ thuật không đồng bộ: MVC có thể hoạt động trên nền tảng JavaScript. Điều này có nghĩa là các ứng dụng MVC có thể hoạt động với các file PDF, các trình duyệt web cụ thể, và cả các widget máy tính.
* Dễ dàng thao tác chỉnh sửa: Bộ phận Model hoạt động tách biệt với View đồng nghĩa với việc bạn có thể đưa ra các thay đổi, chỉnh sửa hoặc cập nhật dễ dàng ở từng bộ phận.

Nhược điểm:

* Khó khăn trong quá trình điều hướng code: Điều hướng khung có thể phức tạp vì mô hình này bao gồm nhiều lớp và yêu cầu người dùng thích ứng với các tiêuchí phân tách của MVC.
* Không thích hợp việc phát triển các ứng dụng nhỏ vì mô hình này yêu cầu bạn lưu trữ một số lượng lớn các file.
* Nhiều khung hoạt động đồng thời: Việc phân tách một tính năng thành ba bộ phận khác nhau dễ dẫn đến hiện tượng phân tán. Do đó, đòi hỏi các nhà phát triển phải duy trì tính nhất quán của nhiều bộ phận cùng một lúc.

***13. Độ đo coupling phản ánh điều gì trong một thiết kế phần mềm? Nó ảnhhưởng thế nào đến chất lượng phần mềm? Người thiết kế mong muốn độ đo coupling là thấp hay cao trong một thiết kế phần mềm? Giải phápkhắc phục là gì?***

Độ đo coupling phản ánh sự liên kết của các thành phần trong kiến trúc phần mềm.Coupling thể hiện sự phụ thuộc, sự kết nối giữa các module với nhau.

* Trong thiết kế, người ta hướng tới mục tiêu Low coupling.
* Low coupling (Loose coupling):
  + - 2 modules được gọi là low coupling nếu chúng ít phụ thuộc vào nhau,sự thay đổi trong module này ít làm ảnh hưởng hoặc không làm ảnh hưởng đến module kia.
* High coupling (Tight coupling):
  + - 2 modules được gọi là high coupling nếu chúng phụ thuộc chặt chẽ vào nhau, sự thay đổi trong module này dẫn đến phải thay đổi ở module kia.Tùy theo nhu cầu của phần mềm cần sản xuất mà lựa chọn độ đo cao hay thấp.
* Giải pháp: Đảo ngược sự phụ thuộc

***14.Độ đo cohesion phản ánh điều gì trong một thiết kế phần mềm? Nó ảnh hưởng thế nào đến chất lượng phần mềm? Người thiết kế mong muốn độ đo này thấp hay cao trong một thiết kế phần mềm? Giải pháp khắc phục là gì?***

* Cohesion thể hiện sức mạnh liên kết giữa các thành phần trong cùng mộtmodule.
  + Khi thiết kế module, cần hướng tới High cohesion.
  + High cohesion:
* Một module được gọi là high cohesion nếu các thành phần bên trongnó có liên kết chặt chẽ với nhau.
* Nếu xét một module là class, thì các thành phần bên trong nó sẽ làcác trường dữ liệu, các phương thức.
* Low cohesion:
* Nếu các thành phần bên trong nó rời rạc, không liên kết với nhau.
* Một module mà thực hiện nhiều chức năng khác nhau, sẽ dễ dẫn tớilow cohesion.
* Ví dụ lớp Utility: định dạng ngày giờ, định dạng đường dẫn...

Có nhiều cách để chia hệ thống thành các module nhỏ khác nhau.

* Việc phân chia tốt sẽ làm cho hệ thống đạt được tính chất high cohesion trong mỗi module và tính chất low coupling giữa các module.

***15.Phân biệt giữa 2 cách tiếp cận trong thiết kế: hướng đối tượng và hướngchức năng. Lấy ví dụ minh họa sự khác nhau thông qua việc thiết kế hệ thống quản lý đào tạo theo tín chỉ của trường Đại học Quy Nhơn (sinh viên có thể đề xuất hệ thống khác).***

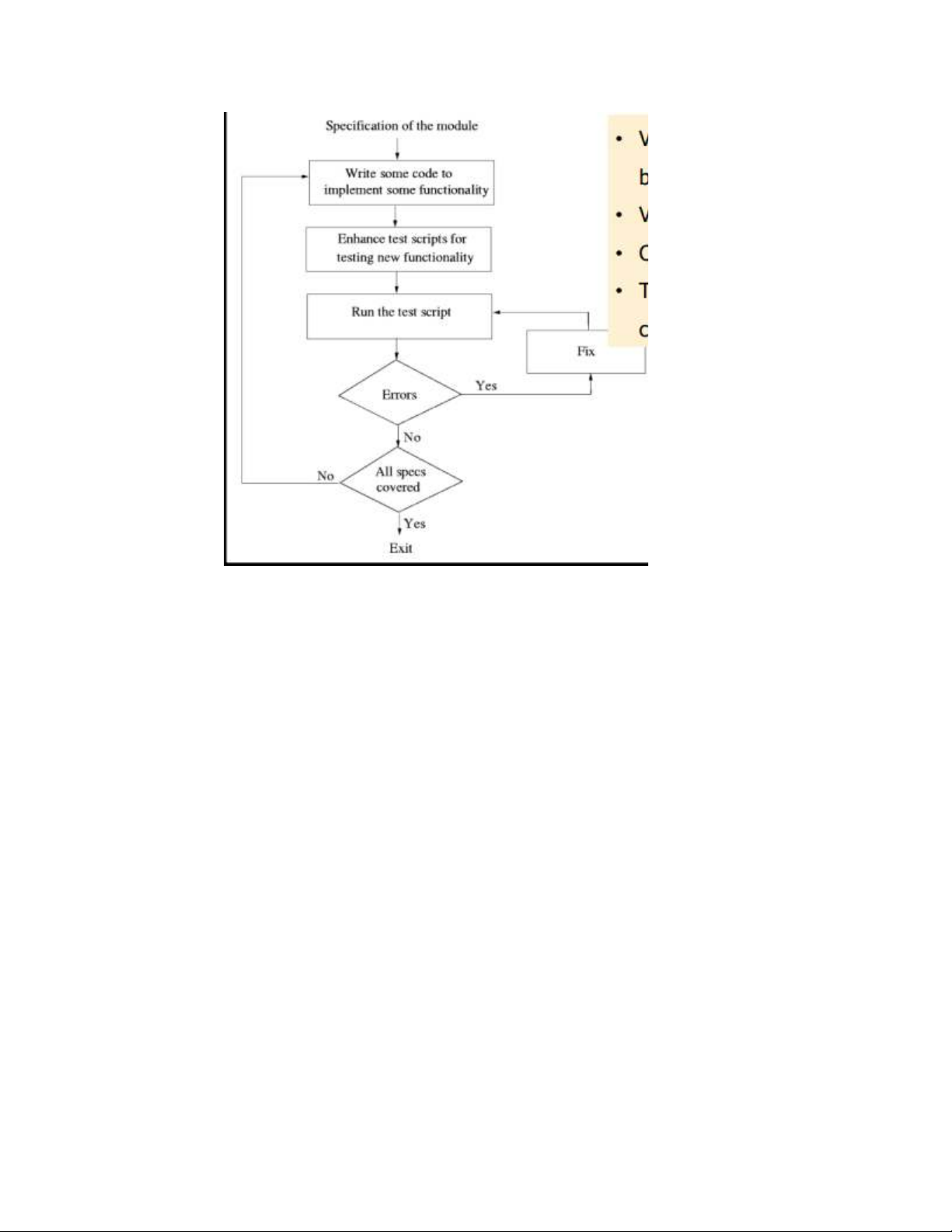
|  |  |  |
| --- | --- | --- |
| Phương pháp | Phân tích thiết kế hướng chức năng | Phân tích thiết kế hướng đối tượng |
| Cách tiếp cận | Đặc trưng của phương pháp hướng chức năng là phân chia chương trình chính thành nhiều chương trình con, mỗi chương trình con nhằm đến thực hiện một công việc xác định.Cách thức thực hiện của phương pháp hướng chức năng làphương pháp thiết kế từ trên xuống (top-down). Phương pháp này tiến hành phân rã bài toán thành các bài toán nhỏ hơn, rồi tiếp tục phân rã các bài toán con cho đến khi nhận được các bài toán có thể cài đặtđược ngay sử dụng các hàm của ngôn ngữ lập trình hướng cấu trúc. | - Khác với phương pháp hướng cấu trúc chỉ tập trung vào dữ liệu hoặc vào hành động , phương pháp hướng đối tượng tậptrung vào cả hai khía cạnh của hệ thống là dữ liệu và hành động  - Cách tiếp cận hướng đối tượng là một lối tư duy theo cách ánh xạ các thành phần trong bài bài toán vào các đối tượng ngoài đời thực. Với cách tiếp cận này, một hệ thống được chia tương ứng thành các phần nhỏ gọi là đối tượng. Mỗi đối tượng bao gồm đầy đủ cả dữ liệu và hành động liên quan đến đối tượng đó. Các đối tượng trong một hệ thống tương đối độc lập với nhau và phần mềm sẽ được xây dựng bằng cách kết hợp các đối tượng đó lại với nhau thông qua các mối quan hệ và tương tác giữa chúng.- - Phương pháp thiết kế từ dưới lên (bottom-up) . Bắt đầu từ những thuộc tính cụ thể của từng đối tượng sau đó tiến hành trừu tượng hóa thành các lớp (Đối tượng) |

***16.Nêu và giải thích các hoạt động trong 2 quy trình viết mã nguồn: Quy trình viết mã tăng dần (Incremental Coding Process) và Quy trình viết mã hướng kiểm thử (Test-Driven Development). So sánh sự giống và khác nhau giữa 2 quy trình này.***

* Quy trình viết mã tăng dần Incremental Coding Process

Viết mã cho một số chức năng ban đầu.

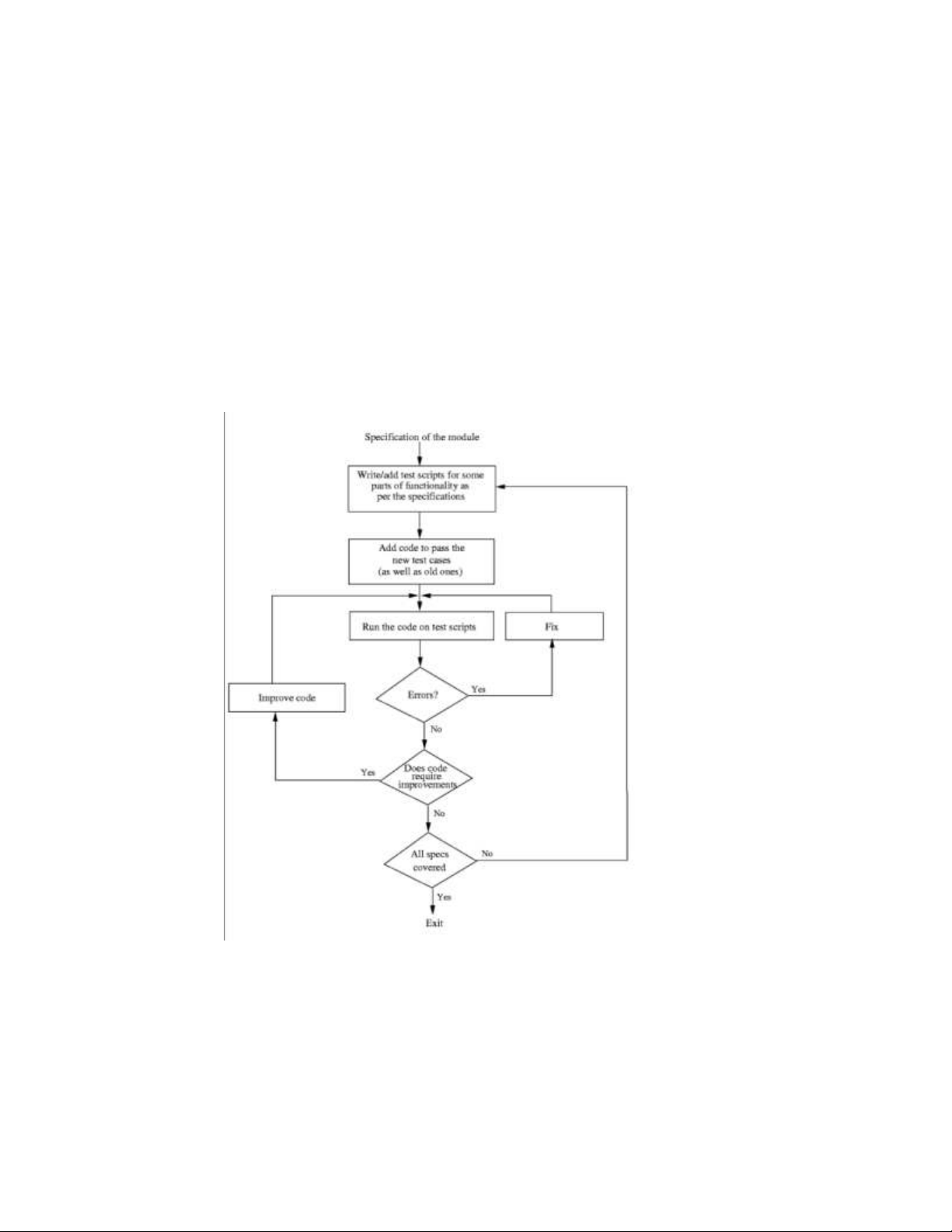
* Viết test script để kiểm thử
* Chạy test script, sửa lỗi nếu có.
* Tiếp tục bổ sung viết mã cho các chức năng khác.



* Điểm mạnh:
* Dễ debug (dễ phát hiện lỗi): vì lập trình viên viết vàchạy test script sau mỗi lần viết mã cho chức năngmới, nên nếu chạy test script phát hiện lỗi, thì lỗi nàythuộc về phần mã mới bổ sung vào gần nhất.
* Việc viết test script có ý nghĩa rất lớn khi mã lệnh càngngày càng nhiều: test script giúp kiểm tra nhanh, chínhxác và thường xuyên.
* Test script là cơ sở để thực hiện kiểm thử đơn vị (unittesting) trước khi đưa module lên server (repository)
* Hạn chế:
* Giữa mã nguồn và test scripts thường không đồng bộ:Viết mã rất nhiều, nhưng test scripts lại ít, không đủ độphủ kiểm tra hết các đoạn mã đã viết ra.ngày càng nhiều: test script giúp kiểm tra nhanh, chínhxác và thường xuyên.
* Test script là cơ sở để thực hiện kiểm thử đơn vị (unittesting) trước khi đưa module lên server (repository)
* Quy trình viết mã hướng kiểm thử Test-Driven Development - TDD
* TDD là quy trình viết mã ngược với hướng tiếp cận thông thường khi viết mã:
* Đầu tiên, viết các test cases để kiểm tra mã trước.
* Sau đó mới viết mã để thỏa các test cases này.

Viết test scripts cho một số chức năng ban đầu dựa trên các đặc tả.

* Viết mã để thỏa các test cases.
* Chạy test script.
* Nếu có lỗi thì sửa, rồi chạy lại test scripts, lặp lại cho đến khi thỏa.
* Nếu không có lỗi, kiểm tra xem có cần cải tiến mã không, nếucó thì cải tiến, rồi chạy lại test script lần nữa.
* Tiếp tục viết test scripts cho các chức năng khác (nếu còn) dựa trên các đặc tả.



* Điểm mạnh:
* Mã nguồn luôn đồng bộ với test scripts.
* Vì phải viết test cases dựa trên các đặc tả, và viết code sau đó để thỏa các test cases nên hệ thống xây dựng ra sẽ đáp ứng tốt các yêu cầu đặt ra.
* Những chức năng quan trọng thường được viết test script trước, nên chúng sẽ được lập trình trước và được kiểm thử nhiều nhất
* Hạn chế:
* Tính đầy đủ của mã nguồn phụ thuộc vào tính toàn diện, đầy đủ của bộ test cases được thiết kế. Thông thường, khó có thể viết các test cases một cách đầy đủ (thường sót đối với các trường hợp đặc biệt).
* Đôi khi việc viết mã để thỏa test case sau lại phải dẫn đến việc thay đổi thuật toán, các đoạn mã đã có.

***17.Quy ước viết mã (có tên gọi tiếng Anh: Coding standards, Coding conventions, Coding style guidelines) là gì? Nó ảnh hưởng thế nào đến chất lượng phần mềm? Hãy nêu và giải thích 05 quy ước viết mã mà bạn biết. Quy ước viết mã có giống nhau cho các ngôn ngữ lập trình không?***

* Code Convention được tạm dịch là quy ước viết Code. Có thể hiểu một cách đơn giản, Code Convention là một tập hợp các quy ước về cách để viết Code,đặt tên biến, class, hàm, file và rất nhiều quy tắc khác như thụt đầu dòng, comment, cách “.” cách “,”,… để cho các khối Code trở nên “clean” hơn.
* Trong một dự án phần mềm lớn, khi toàn bộ lập trình viên của dự án đều tuân theo một quy tắc viết Code, giúp việc giao tiếp giữa các thành viên trở nên dễ dàng hơn. Dự án cũng sẽ có thể thêm các module chức năng nhanh chóng hơn, việc bảo trì và phát triển hệ thống sau này cũng sẽ dễ dàng hơn.

Vậy, khi Code Convention chúng ta sẽ có những lợi ích như sau:

* Giúp làm việc nhóm hiệu quả hơn
* Thống nhất và tuân thủ theo một chuẩn dễ dàng làm việc hơn
* Giúp người khác nắm bắt Code bạn viết nhanh hơn
* Dễ dàng nâng cấp và cải tiến phần mềm
* Có thể tái sử dụng trong nhiều phần mềm khác nhau
* Thuận lợi trong việc phát triển và bảo trì hệ thống sau này
* các hàm, tên biến, phương thức: camelCase-tên class: PascalCase
* Một dòng Code không nên dài quá 80 ký tự
* Một câu lệnh nên lồng tối đa 4 cấp
* Một hàm không nên chứa quá 5 tham số
* Một hàm không nên quá 30 dòng
* Một class không nên vượt 500 dòng

***18.Tái cấu trúc mã nguồn là gì? Vì sao cần tái cấu trúc mã nguồn? Hãy nêu và giải thích 03 hoạt động tái cấu trúc mã nguồn mà bạn biết. Nêu tên một IDE bạn biết có hỗ trợ tái cấu trúc mã nguồn.***

* Refactoring là một kỹ thuật cho việc tái cấu trúc mã nguồn đã có, thay đổicấu trúc bên trong mà không làm thay đổi hành vi bên ngoài.
* Trong quá trình phát triển hệ thống, mã nguồn luôn thường xuyên bị thayđổi:
* Do thêm tính năng mới
* Do những thay đổi yêu cầu
* Cho dù đã được thiết kế tốt, thì sau một thời gian với những thay đổi, bổsung vào mã nguồn, cấu trúc của chúng dần trở nên phức tạp và không tốt.

Dẫn đến:

* Đọc mã nguồn khó hơn (cấu trúc phức tạp, lộn xộn)
* Khó tìm lỗi hơn (khi xảy ra vấn đề)
* Khó khăn cho việc đáp ứng các thay đổi yêu cầu.
* Về tổng thể: chất lượng, hiệu suất giảm, chi phí sản xuất tăng.
* Theo định nghĩa của Martin Fowler (https://refactoring.com/): Refactoring là việc làm thay đổi cấu trúc bên trong của phần mềm, làm cho nó dễ hiểu,chi phí sửa đổi thấp mà không làm thay đổi hành vi bên ngoài.
* Mục tiêu cơ bản của Refactoring là cải tiến thiết kế (ở giai đoạn viết mã,không phải ở giai đoạn thiết kế).
* Kết quả: làm tăng cohesion và giảm coupling, tuân theo nguyên lýđóng-mở tốt hơn.
* Đối tượng của Refactoring là: mã nguồn.
* Rủi ro khi làm refactoring là: làm hỏng hệ thống.
* Để giảm thiểu rủi ro, cần tuân theo 2 nguyên tắc:
* Tái cấu trúc từng bước nhỏ một.
* Có test scripts để kiểm tra lại sau khi thay đổi.
* Khi nào thì cần thực hiện Refactoring? Đó là khi trong mã nguồn có một sốdấu hiệu không tốt (gọi là “bad smells”):
* Trùng mã (Duplicate code)
* Phương thức quá dài (Long method)
* Khó đọc, khó bảo trì
* Nếu > 10 dòng code 🡪 nên đặt câu hỏi
* Nếu trong phương thức cần có comment 🡪nên suy nghĩ táchphương thức
* Tham số phương thức quá nhiều (Long parameter list)
* Khó đọc, khó bảo trì
* Mỗi lần gọi, phải quan tâm thứ tự tham số
* Giải pháp:
* Parameter Object
* Thay parameter bằng method
* Lớp quá dài (Long class)
* Khó bảo trì
* Giải pháp chia nhỏ class

Tên một IDE bạn biết có hỗ trợ tái cấu trúc mã nguồn.

* NetBeans
* Xcode
* Code::Blocks
* Microsoft Visual Studio

**3 hoạt động tái cấu trúc mã nguồn:**

* Tách method: di chuyển code đến một method mới riêng (hoặc hàm) và thay đoạn code cũ bằng lời gọi đến method
* Tách các biến tạm: thay biến tạm thành các biến khác nhau cógiá trị khác nhau. Mỗi biến nên dùng cho một thứ duy nhất
* Tách biến: trong trường hợp code đọc quá khó hiểu, ta cho kết quả của một biểu thức vào một biến và dùng biến đó thay vào những đoạn biểu thức dài dòng khó hiểu
* Di chuyển method: di chuyển đến lớp sử dụng nhiều nhất

***19.Vì sao cần quản lý mã nguồn? Nêu tên và giới thiệu vắn tắt về một công cụ quản lý mã nguồn mà bạn biết. Nêu tên và mục đích của 05 hoạt quản lý mã nguồn phổ biến. Phân loại các công cụ quản lý mã nguồn.***

* Trong quá trình phát triển chương trình phần mềm, sau một thời gian mã nguồn chương trình sẽ ngày càng nhiều, rất khó để lập trình viên có thể kiểm soát được các chức năng đã thực hiện, cũng như quản lý tất cả mã nguồn đã viết ra. Đặc biệt đối với nhóm lập trình thì việc chia sẻ mã nguồn, quản lý công việc giữa các thành viên trong nhóm càng trở nên cấp thiết nhằm không bị chồng chéo công việc cũng như tăng tốc thực hiện dự án.
* Git: Đây là công cụ quản lý các source code được tổ chức theo dạng các dữ liệu phân tán, do vậy nó được đánh giá vô cùng cao. Ngoài ra, sản phẩm này có khả năng đồng bộ các source code của team lên 1 server mới, giúp thời gian điều chỉnh nhanh và hiệu quả. Bất cứ ai cũng có thể được GIT hỗ trợ cácthao tác về vấn đề kiểm tra source code trong quá trình làm việc
* Commit: Lưu các thay đổi từ thư mục đang làm việc (working coppy) vào server local (local repo)
* Push: Cập nhật các thay đổi từ server local ở máy(local repo) lên server online (remote repo)Các thành viên trong team cần lấy về sử dụng cần làm các thao tác:
* Fetch: Để thực hiện đồng bộ mọi thứ thay đổi mới nhất từ server online (remote repo) về server local (local repo) chúng ta sử dụng lệnh fetch (git fetch)
* Pull: Sau đó để “áp dụng” các thay đổi đó vào source code đang sử dụng. Có nghĩa là là cập nhật từ server local (local repo) vào thư mục đang làm việc (working copy) (git pull)
* Clone: Lấy source từ server về server local của máy (local repo) của người dùng.Phân loại các công cụ quản lý mã nguồn: quản lý phiên bản tập trung và quản lý phiên bản phân tán

PHẦN 3: KIỂM THỬ PHẦN MỀM

***20.Kiểm thử phần mềm là gì? Nêu và giải thích các mức độ kiểm thử trong một dự án phát triển phần mềm.***

* Kiểm thử phần mềm là quá trình vận hành để tìm ra lỗi.
* Một ca kiểm thử (test case) tốt: là ca kiểm thử có xác suất cao tìm ra lỗi chưa phát hiện.
* Mục tiêu: thiết kế các ca kiểm thử để có thể phát hiện ra một cách có hệ thống các loại lỗi khác nhau với chi phí thời gian và công sức ít nhất có thể.

**Các mức độ kiểm thử trong một dự án phát triển phần mềm.**

* unit testing (Kiểm thử đơn vị): thường do Developer phụ trách, họ sẽ đi kiểm tra các module, các hàm, các phương thức, các lớp,… mà họ viết ra nhằm gia tăng sự tin cậy cho các chức năng mà mình viết.
* Integration testing (Kiểm thử tích hợp): Kiểm thử tích hợp là kiểm thử sự tương tác giữa các chức năng với nhau trong hệ thống và được thực hiện bởi Tester
* System testing (Kiểm thử hệ thống): Kiểm thử hệ thống là kiểm thử một hệ thống đã hoàn thành, đã tích hợp đầy đủ các chức năng nhằm kiểm tra xem hệ thống phần mềm đó có đáp ứng đầy đủ các yêu cầu chức năng theo bản đặc tả yêu cầu phần mềm (SRS) hay không. Người thực hiện test level này thường là Tester.
* Acceptance testing (Kiểm thử chấp nhận): Mức độ kiểm thử phần mềm cuối cùng chính là Acceptance Test (Kiểm thử chấp nhận)
* kiểm tra xem hệ thống có đáp ứng đúng nhu cầu và mong đợi của khách hàng hay không. Kiểm thử chấp nhận thường là trách nhiệm của người dùng hoặc khách hàng. Trong kiểm thử hệ thống, khách hàng sẽ kiểm tra xem phần mềm được viết có hoạt động đúng như mong đợi của mình không, có đảm bảo tính tiện dụng, hiệu suất hoạt động có như mong đợi không, có bảo mật tốt hay không,…

***21.Trình bày mục đích của việc kiểm thử phần mềm. Nêu sự khác nhau giữa Xác minh (Software verification) và Thẩm định (Software validation).***

* **Mục đích của việc kiểm thử phần mềm**
* Tìm ra các lỗi phát sinh chương trình.
* Ngăn chặn lỗi.
* Cung cấp thông tin độc lập về chất lượng sản phẩm / phần mềm.
* Cung cấp cho khách hàng một sản phẩm phần mềm chất lượng.
* Đảm bảo rằng kết quả cuối cùng đáp ứng các yêu cầu kinh doanh và người sử dụng.
* **Sự khác nhau giữa Xác minh (Software verification) và Thẩm định (Software validation).**

|  |  |
| --- | --- |
| Giống nhau | |
| * Phát triển đúng theo đặc tả * Đáp ứng nhu cầu của người dùng | |
| Khác nhau | |
| Xác minh(verification) | Thẩm định(validation) |
| Là sự kiểm tra xem sản phẩm có đúng với đặc tả không , tức là chú trọng vào việc phát hiện lỗi phần mềm | Là sự kiểm tra xem sản phẩm có đáp ứng được nhu cầu người dùng không , tức là chú trọng vào sự khác biệt giữa phần mềm làm ra và cái người dùng mong đợi. |

***22.Test case là gì? Cấu trúc của một Test case gồm những thành phần cơ bản nào?***

* **"Test case"** là một quá trình "kiểm tra dữ liệu" đầu vào, có thể là một hành động hoặc một sự kiện nào đó sau đó trả về kết quả truy vấn để kiểm tra từng chức năng của phần mềm hay ứng dụng có hoạt động đúng chức năng hay không?. Việc test case là vô cùng cần thiết.
* **Cấu trúc của một Test case gồm những thành phần cơ bản:**
* Mã yêu cầu/ mã chức năng
* Chức năng cần kiểm thử
* Điều kiện kiểm thử: thể hiện trạng thái hệ thống/môi trường dùng để kiểm thử
* Dữ liệu đầu vào
* Mô tả các bước thực hiện
* Kết quả mong đợi
* Kết quả test

***23.Test plan là gì? Một bản test plan gồm những nội dung nào?***

Test plan hay còn gọi là một bản kế hoạch kiểm thử là một tài liệu tổng quát chotoàn bộ dự án, định nghĩa: Phạm vi kiểm thử, hướng tiếp cận để thực hiện kiểm thử, lịch trình kiểm thử, các phần cần kiểm thử, những người phụ trách các hoạtđộng kiểm thử.

Việc lập kế hoạch kiểm thử có thể thực hiện trước khi thực hiện kiểm thử:

– Có thể song song với quá trình thiết kế hệ thống và lập trình

* **Một bản test plan gồm những nội dung**
* Định nghĩa các đơn vị kiểm thử
* Các tính năng sẽ kiểm thử
* Phương pháp kiểm thử
* Các kết quả kiểm thử
* Phân bổ nhiệm vụ và lịch trình

***24.So sánh các phương pháp kiểm thử hộp đen và kiểm thử hộp trắng***

|  |  |  |
| --- | --- | --- |
|  | Kiểm thử hộp đen | Kiểm thử hộp trắng |
| Đặc điểm | - Xem phần mềm như mộthộp đen, người kiểm thử không cần quan tâm đến cấu trúc và hoạt động bên trong.  - Cách thực hiện: nhập dữliệu đầu vào qua giao diện, thực thi chương trình, xem kết quả đầu ra có đúng với mong đợi không?  - Còn được gọi là kiểm thử chức năng (functional testing) hay kiểm thử hành vi (behavior testing)  - Cơ sở của kiểm thử hộp đen là đặc tả chức năng và dữ liệu đầu vào mà nó sử dụng. | - Là việc kiểm tra các đoạn mã chương trình, xem xét cấu trúc bên trong, xem nó có vận hành đúng theo thiết kế hay không.  - Người kiểm thử yêu cầuphải hiểu biết về lập trình,về sự vận hành của chương trình.  - Các tên gọi khác: glass testing, structural testing,open box testing, clear box testing.    - Cơ sở của kiểm thử hộp trắng: dựa trên mã nguồn:các rẽ nhánh, vòng lặp (if,while, for...) |
| Điểm mạnh | - Không cần truy cập mã nguồn  - Không cần kiến thức về các ngôn ngữ lập trình  - Kiểm thử được thực hiện với góc nhìn của người dung 🡪Mang tính khách quan (tách biệt với góc nhìn của lập trình viên) | Rất hiệu quả trong việc tìm lỗi, đặc biệt là các lỗi ẩn (mà blackbox testing không phát hiện ra) |
| Hạn chế | - Khó thiết kế test cases  - Kiểm thử tất cả các khả năng là không khả thi | - Phải truy cập vào mã nguồn  - Có thể không nhìn ra những chức năng bị thiếu (chưa viết) |

BÀI TẬP

