

SWEBOK3

Hoàng Trương

Published
with GitBook



Mục Lục

Introduction	0
Chương 1: Yêu cầu phần mềm	1
Chương 2: Thiết kế phần mềm	2
Chương 3: Xây dựng phần mềm	3
Chương 4: Kiểm thử phần mềm	4
Chương 5: Bảo trì phần mềm	5
Chapter 6: Software Configuration Management	6
Chương 7: Quản lý công nghệ phần mềm	7
Chapter 9: Software Engineering Models and Methods	8
Chapter 10: Software Quality	8.1
Chương 11: Chuyên nghiệp trong kỹ nghệ phần mềm	9
Chapter 10: Software Quality	10
Chương 8: Quy trình công nghệ phần mềm	11
Chú giải	

Tóm tắt SWEBOK v3.0

Tóm tắt về cuốn sách Software Engineering Body of Knowledge 3.0.

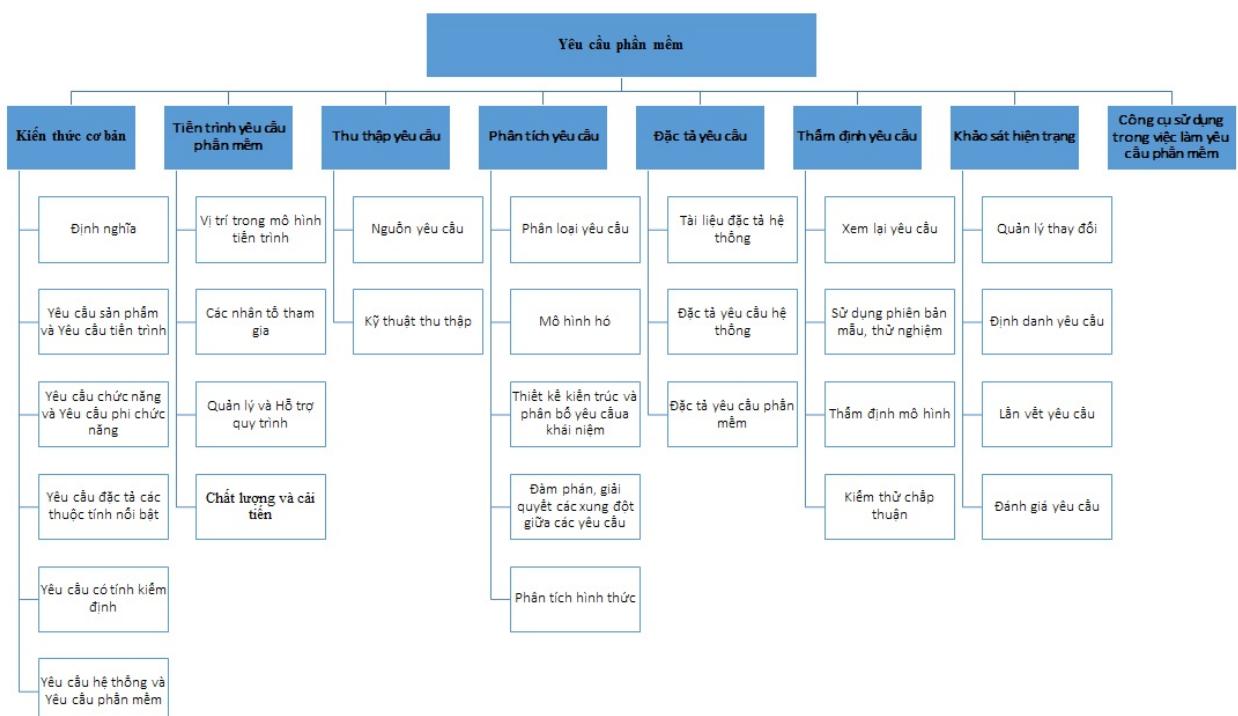
Chương 1: Yêu cầu phần mềm

Thuật ngữ thường dùng

- **CIA**: Confidentiality, Integrity, and Availability (Độ tin cậy, Tính toàn vẹn, tính khả dụng)
- **DAG**: Directed Acyclic Graph (Đồ thị có hướng và không có chu trình)
- **FSM**: Functional Size Measurement (Thước đo chức năng)
- **INCOSE**: International Council on Systems Engineering ()
- **UML**: Unified Modeling Language (Ngôn ngữ mô hình hóa)
- **SysML**: Systems Modeling Language (Ngôn ngữ mô hình hóa hệ thống)

1.1. Kiến thức cơ bản

Tổng quan



Định nghĩa

- Yêu cầu cho 1 phần mềm cụ thể là tổng hợp những yêu cầu từ nhiều người khác nhau

về tổ chức, mức độ chuyên môn và mức độ tham gia, tương tác với phần mềm trong môi trường hoạt động của nó.

- Có thể kiểm chứng một cách riêng rẽ ở mức chức năng(yêu cầu chức năng) hoặc mức hệ thống(yêu cầu phi chức năng)
- Cung cấp các chỉ số đánh giá độ ưu tiên về các mặt khi cân nhắc về nguồn tài nguyên.
- Cung cấp các giá trị trạng thái để theo dõi tiến độ của dự án.

Phân loại

- Theo sản phẩm và tiến trình
 - Yêu cầu sản phẩm: là những đòi hỏi hay ràng buộc mà phần mềm phải thực hiện.
 - Yêu cầu tiến trình: là những ràng buộc liên quan đến việc phát triển phần mềm đó(quy trình, đối tác kiểm thử, phân tích, kĩ thuật sử dụng,...).

Ví dụ: Trong Project 1:

Yêu cầu sản phẩm là xây dựng trang web Trường học điện tử với các tính năng như Giáo viên quản lý câu hỏi, đề thi; Học sinh tham gia làm bài; Admin duyệt câu hỏi của giáo viên trước khi đăng,...

Yêu cầu tiến trình: Phải thực hiện theo mô hình Agile. Sản phẩm cuối cùng bao gồm cả sản phẩm và backlog cho từng Sprint.

- Theo chức năng
 - Yêu cầu chức năng: đặc tả các chức năng mà phần mềm phải thực hiện.
 - Yêu cầu phi chức năng: là các ràng buộc về giải pháp và chất lượng(hiệu năng, việc bảo trì, độ an toàn, bảo mật,...).
 - Yêu cầu đặc tả các thuộc tính nổi bật: là các đặc tả cho các thuộc tính phụ thuộc vào sự vận hành,... đặc biệt là kiến trúc hệ thống. Các thuộc tính này không thể xác định được cho từng thành phần đơn lẻ.
- Theo tính kiểm định
 - Mơ hồ, không thể kiểm định
 - Rõ ràng, định lượng và có thể kiểm định được.
- Theo phạm vi đặc tả
 - Yêu cầu hệ thống: đặc tả các cấu hình, cơ sở hạ tầng, phần cứng, phần mềm, con người, kỹ thuật,... của toàn bộ hệ thống.
 - Yêu cầu phần mềm: đặc tả các chức năng, giao diện,... của các cấu phần phần mềm.

1.2. Tiến trình yêu cầu phần mềm

Vị trí trong mô hình tiến trình

- Xuất phát từ giai đoạn khởi tạo dự án, cho tới khi hoàn thiện các tiến trình trong vòng đời phát triển của phần mềm. Không chỉ là các hoạt động bề nổi nhìn thấy được.
- Được nhận biết như phần cấu hình của mọi việc, và được quản lí như việc quản lí các cấu hình; hoặc là sản phẩm của các quá trình trong vòng đời phát triển.
- Được thiết kế theo từng tổ chức và hoàn cảnh của từng dự án.

Các nhân tố tham gia

- Người dùng: vận hành phần mềm.
- Khách hàng: gồm những người được hưởng mục tiêu cuối cùng của phần mềm, những giá trị thương mại mà phần mềm hướng đến.
- Nhà phân tích thị trường: phân tích nhu cầu thị trường và tương tác với bên đại diện khách hàng.
- Đại diện pháp lý: kiểm soát việc tuân thủ quy định, quy ước chung, quy định pháp lý.
- Kỹ sư phần mềm: thu lợi nhuận hợp pháp từ việc phát triển phần mềm.

Quản lý và Hỗ trợ quy trình

- Quản lí nguồn tài nguyên được sử dụng trong các tiến trình.
- Cân đối nguồn nhân lực, tài chính, đào tạo và công cụ.

Chất lượng và cải tiến

- Xác định vai trò của tiến trình xây dựng yêu cầu về các mặt chi phí, thời gian và sự thỏa mãn của khách hàng với sản phẩm.
- Định hướng tiến trình theo các chuẩn chất lượng và xây dựng mô hình cải tiến cho phần mềm và hệ thống.
- Bao gồm:
 - Độ bao phủ theo các mô hình và chuẩn cải tiến.
 - Việc đo đạc và đánh giá tiến trình.
 - Việc thực hiện và lên kế hoạch cải tiến.
 - Việc cài đặt và lên kế hoạch cho an ninh.

1.3. Thu thập yêu cầu

Là giai đoạn đầu tiên trong việc xây dựng sự hiểu biết về sản phẩm phần mềm và các vấn đề cần thiết phải giải quyết (ví dụ *cần hiểu biết về các chức năng của phần mềm*). Đây cũng là giai đoạn mà các bên liên quan (stakeholders) được xác định. Thiết lập các mối quan hệ

giữa các nhóm phát triển và khách hàng.

Một trong những nguyên tắc cơ bản của quá trình thu thập yêu cầu là sự trao đổi giữa các bên liên quan. Sự trao đổi liên tục qua toàn bộ vòng đời phát triển phần mềm (SDLC), quá trình trao đổi với các bên liên quan khác nhau tại mỗi các thời điểm khác nhau. Trước khi bắt đầu phát triển, các chuyên gia thu thập yêu cầu có thể tạo ra các kênh cho sự giao tiếp này. Họ sẽ là trung gian giữa khách hàng và kỹ sư phần mềm.

Một số lợi ích của thu thập yêu cầu:

- Tạo được niềm tin của khách hàng khi họ được tham gia vào giai đoạn thu thập yêu cầu.
- Giảm việc phải làm lại trong quá trình phát triển
- Quá trình phát triển sẽ nhanh hơn, giảm được những chi phí cho những yêu cầu không cần thiết.
- Hạn chế phạm vi hệ thống bị phình rộng.

1.3.1. Nguồn yêu cầu - Requirements Sources

Các yêu cầu có rất nhiều nguồn trong đặc thù phần mềm và điều quan trọng là tất cả các nguồn tiềm năng cần được xác minh và đánh giá. Phản này nhằm nâng cao nhận thức của các nguồn khác nhau của yêu cầu phần mềm và những framework để quản lý chúng.

Những điểm chính của nguồn yêu cầu bao gồm:

- Mục tiêu - Goal. Các mục tiêu về giá trị và giá thành thường mơ hồ, không rõ ràng. Kỹ sư phần mềm cần chú ý để xác định rõ các mục tiêu đó. Nghiên cứu tính khả thi là sẽ giúp giảm giá thành của quá trình phát triển. Ví dụ kỹ sư phần mềm cần xác định chi phí xây dựng server với chi phí đi mua cái nào sẽ tối ưu hơn để lựa chọn.
- Hiểu biết về các lĩnh vực. Các kỹ sư phần mềm cần có kiến thức về các lĩnh vực như: mua sắm, ngân hàng, chăm sóc sức khỏe,... lĩnh vực mà phần mềm được sử dụng. Việc hiểu biết về các lĩnh vực sẽ giúp cho người thu thập yêu cầu thu thập được những thông tin chính xác cao.
- Các bên liên quan. Nhiều phần mềm đã được chứng minh không đạt yêu cầu vì nó chỉ tập trung vào yêu cầu của một số bên mà bỏ qua các bên khác. Do đó phần mềm đã giao rất khó để sử dụng hoặc phá vỡ văn hóa hoặc tổ chức chính trị của tổ chức khác hàng. Các kỹ sư phần mềm cần phải xác định, miêu tả và quản lý các yêu cầu của các bên liên quan. Ví dụ phần mềm cho người không chuyên thì sử chuột và các menu chọn, nhưng với người thành thạo thì cần có các host-key để rút ngắn thời gian tương tác
- Nguyên tắc kinh doanh. Là những điều kiện hoặc các ràng buộc được xác định để các doanh nghiệp hoạt động được. "Một sinh viên không thể đăng ký vào các khóa học học kỳ tiếp theo nếu vẫn còn một số môn chưa thanh toán học phí" sẽ là một ví dụ của nguyên tắc kinh doanh đó cho các phần mềm đăng ký môn học của trường đại học.
- Môi trường vận hành. Các yêu cầu sẽ được bắt nguồn từ môi trường mà trong đó phần

mềm sẽ được thực thi. Ví dụ như ràng buộc thời gian trong phần mềm thời gian thực hoặc ràng buộc hiệu năng trong môi trường kinh doanh.

- Môi trường tổ chức. Phần mềm thường có thể bị ràng buộc bởi cấu trúc, văn hóa và tổ chức chính trị. Các kỹ sư phần mềm cần phải hiểu biết về chúng, phần mềm không nên ép buộc thay đổi ngoài ý muốn trong quá trình kinh doanh.

1.3.2. Kỹ thuật thu thập- Elicitation Techniques

Một khi các nguồn yêu cầu được xác định, các kỹ sư phần mềm có thể bắt đầu thu thập thông tin yêu cầu từ chúng. Phần này tập trung vào các kỹ thuật để thu thập các thông tin cần thiết từ các bên liên quan.

Các kỹ sư phần mềm cần phải linh hoạt với các sự việc xảy ra ví dụ: người dùng gặp khó khăn trong việc mô tả yêu cầu của họ, có thể thông tin quan trọng không được nói ra hoặc có thể không muốn hoặc không thể hợp tác.

Thu thập không phải là hoạt động thụ động, ngay cả khi các bên liên quan sẵn sàng hợp tác các kỹ sư phần mềm phải cố gắng để thu thập thông tin chính xác nhất.

Một số kỹ thuật thu thập yêu cầu như:

- Phỏng vấn.** Phỏng vấn là cách truyền thống để thu thập yêu cầu. Ưu điểm: Thu thập được được những thông tin trực tiếp các thông tin có chất lượng cao, tính chân thực và độ tin cậy có thể kiểm nghiệm được trong quá trình phỏng vấn. Nhược điểm: Đòi hỏi người phỏng vấn phải có trình độ cao, am hiểu, có kỹ năng xử lý các tình huống. Khó triển khai được trên quy mô rộng. Tiếp cận khách hàng là việc tương đối khó vì cần hẹn trước.
- Kịch bản.** Kỹ sư phần mềm cung cấp một hệ thống các câu hỏi bằng việc sử dụng các câu hỏi như "What if" và "How is this done". Các loại kịch bản phổ biến được sử dụng là mô tả các trường hợp sử dụng. Ví dụ: Điều gì sẽ xảy ra với phần mềm khi bị mất kết nối mạng,...
- Các bản mẫu.** Kỹ thuật này sẽ làm rõ các yêu cầu không rõ ràng. Có thể sử dụng mockup hoặc màn hình thiết kế hoặc các bản thử nghiệm để xác minh những yêu cầu từ khách hàng. Ví dụ: Thu thập các yêu cầu về thiết kế hoặc chức năng của phần mềm.
- Cuộc hội họp.** Một nhóm người sẽ mang lại nhiều cái nhìn sâu sắc hơn vào các yêu cầu phần mềm. Mọi người sẽ cùng suy nghĩ, tinh chỉnh các yêu cầu. Lợi thế của phương pháp này là các yêu cầu mâu thuẫn sẽ dễ dàng xử lý hơn.

1.4. Phân tích yêu cầu

Nhằm mục đích:

- Phát hiện và giải quyết xung đột giữa các yêu cầu
- Tìm ra những giới hạn của phần mềm và cách phần mềm tương tác với tổ chức và môi trường hoạt động của nó.

- Nghiên cứu các yêu cầu hệ thống để lấy được các yêu cầu phần mềm.

1.4.1. Mô hình hóa khái niệm

Xây dựng các mô hình trong một vấn đề thực tế là chìa khóa của phân tích yêu cầu phần mềm. Mục đích của nó là để hiểu rõ về những vấn đề xảy ra cũng như miêu tả được giải pháp của vấn đề.

Do đó mô hình khái niệm bao gồm các mô hình của các thực thể từ miền vấn đề, cấu hình để phản ánh các mối quan hệ trong thế giới thực và ràng buộc.

Có rất nhiều loại mô hình có thể được phát triển bao gồm biểu đồ use case, mô hình luồng dữ liệu, mô hình trạng thái, mô hình dựa trên mục tiêu, tương tác người dùng, mô hình dữ liệu, mô hình đối tượng...

Các yếu tố ảnh hưởng đến sự lựa chọn mô hình bao gồm:

- Vấn đề tự nhiên. Một số loại phần mềm đòi hỏi một số khía cạnh được phân tích đặc biệt nghiêm ngặt. ví dụ mô hình trạng thái và mô hình tham số của phần mềm thời gian thực quan trọng hơn so với hệ thống thông tin.
- Sự thành thạo của kỹ sư phần mềm. Kỹ sư phần mềm có kinh nghiệm sẽ lựa chọn các mô hình hay phương pháp để được kết quả tốt hơn.
- Các yêu cầu về quy trình của khách hàng. Khách hàng có thể áp đặt các ký hiệu ưa thích của họ hoặc phương pháp hoặc ngăn cản bất cứ cái gì mà họ thấy không quen thuộc. Nhân tố này có thể xung đột với các nhân tố trước đó.

1.4.2. Thiết kế kiến trúc và phân bổ yêu cầu

Thiết kế kiến trúc là điểm mà tại đó quá trình yêu cầu trùng lặp với phần mềm hoặc các hệ thống thiết kế. Trong nhiều trường hợp, các hành vi kỹ sư phần mềm như là kiến trúc sư phần mềm bởi vì quá trình phân tích và xây dựng các yêu cầu đòi hỏi rằng các thành phần kiến trúc / thiết kế đó sẽ chịu trách nhiệm đáp ứng các yêu cầu được xác định.

Phân bổ là quan trọng để cho phép phân tích chi tiết các yêu cầu. Do đó, ví dụ, khi một bộ các yêu cầu đã được phân bổ cho một thành phần, các yêu cầu cá nhân có thể được phân tích thêm để khám phá thêm các yêu cầu về cách thành phần tương tác với thành phần khác để đáp ứng các yêu cầu được giao.

Trong các dự án lớn, phân bổ thúc đẩy một vòng mới của phân tích cho mỗi hệ thống. Thiết kế kiến trúc được xác định chặt chẽ với các mô hình khái niệm.

1.4.3. Đàm phán, giải quyết các xung đột giữa các yêu cầu

Điều này liên quan đến việc giải quyết vấn đề giữa hai yêu cầu của các bên liên quan cùng các tính năng không tương thích, giữa các yêu cầu và nhân lực hoặc giữa yêu cầu chức năng và yêu cầu phi chức năng.

Trong tất cả các trường hợp kỹ sư phần mềm không được tự đưa ra các quyết định mà cần thiết tham khảo từ các bên liên quan để đạt được một sự đồng thuận trên sự thỏa hiệp thích hợp.

Tuy nhiên, thường rất khó khăn để có được thông tin thực. Ngoài ra, các yêu cầu thường phụ thuộc vào nhau, và có ưu tiên tương đối. Trong thực tế, các kỹ sư phần mềm thực hiện các yêu cầu ưu tiên thường xuyên mà không biết về tất cả các yêu cầu. Nó cũng bao gồm một phân tích từ các kỹ sư phần mềm ước tính chi phí thực hiện từng yêu cầu hoặc liên quan đến các yêu cầu khác.

1.4.4. Phân tích hình thức - Formal Analysis

Formal Analysis đã có một tác động trên một số lĩnh vực ứng dụng, đặc biệt là các hệ thống toàn vẹn cao. Các hình thức thể hiện của các yêu cầu đòi hỏi một ngôn ngữ với ngữ nghĩa định nghĩa chính thức(vd: Ngôn ngữ Z).

Việc sử dụng một phân tích hình thức cho các yêu cầu biểu hiện có hai lợi ích. Đầu tiên, nó cho phép các yêu cầu thể hiện bằng ngôn ngữ được xác định một cách chính xác và rõ ràng, do vậy tránh được khả năng hiểu sai.

Thứ hai, yêu cầu có thể được lý giải trên, cho phép đặc tính mong muốn của phần mềm cụ thể để chứng minh.

Phân tích hình thức nhất là tập trung vào giai đoạn khá muộn của phân tích yêu cầu.

1.5. Đặc tả yêu cầu

Đặc tả yêu cầu là một mô tả của hệ thống phần mềm được phát triển, đưa ra các yêu cầu chức năng và phi chức năng, và có thể bao gồm một tập hợp các ca sử dụng (use cases) để mô tả tương tác giữa người dùng với phần mềm.

Đặc tả yêu cầu tạo cơ sở cho một thỏa thuận giữa khách hàng và nhà cung cấp về những gì phần mềm đã làm được tốt cũng như những gì chưa được như mong đợi. Nó cũng cung cấp một cơ sở thực tế để ước tính giá thành sản phẩm, rủi ro và lịch trình.

Đối với các hệ thống phức tạp có 3 loại tài liệu được tạo ra là: định nghĩa hệ thống, yêu cầu hệ thống và các yêu cầu phần mềm. Đối với sản phẩm phần mềm đơn giản chỉ cần 1 trong 3 tài liệu.

1.5.1. Tài liệu đặc tả hệ thống.

Còn được biết như là tài liệu yêu cầu người dùng hay là tài liệu vận hành ghi lại những yêu cầu hệ thống. Nó xác định yêu cầu hệ thống ở mức cao với cách nhìn từ domain. Độc giả của tài liệu bao gồm hệ thống người dùng hoặc khách hàng. Vì vậy nội dung của nó phải được diễn đạt bằng những từ ngữ của những lĩnh vực riêng. Tài liệu sẽ liệt kê các yêu cầu

hệ thống cùng với các thông tin cơ bản về đối tượng hệ thống, môi trường mục tiêu của nó, giả định và các yêu cầu phi chức năng.

Nó có thể bao gồm mô hình khái niệm được thiết kế để minh họa cho ngữ cảnh hệ thống, sử dụng kịch bản, và các miền thực thể chính, cũng như luồng công việc.

1.5.2. Đặc tả yêu cầu hệ thống

Người phát triển những dự án phần mềm có những thành phần thuần túy là software và những phần non-software - ví dụ như máy bay hiện đại thường tách biệt yêu cầu hệ thống với yêu cầu phần mềm. Theo quan điểm này, yêu cầu hệ thống được quy định, các yêu cầu phần mềm có nguồn gốc từ các yêu cầu hệ thống, và sau đó các yêu cầu đối với các thành phần phần mềm được xác định.

1.5.3. Đặc tả yêu cầu phần mềm

Đặc tả yêu cầu phần mềm tạo cơ sở cho việc thỏa thuận giữa khách hàng và nhà thầu hoặc các nhà cung cấp về những gì sản phẩm phần mềm có làm việc đúng như mong muốn không. Nó cho phép một đánh giá nghiêm ngặt các yêu cầu trước khi có thể bắt đầu vào việc thiết kế và làm giảm việc thiết kế lại. Nó cũng cần cung cấp một cơ sở thực tế để ước tính giá thành sản phẩm, rủi ro, và lịch trình.

Các tổ chức cũng có thể sử dụng một tài liệu đặc tả yêu cầu phần mềm làm cơ sở để phát triển kế hoạch kiểm tra và xác minh. Đặc tả yêu cầu phần mềm cung cấp một cơ sở thông báo cho chuyển một sản phẩm phần mềm cho người dùng mới hoặc các nền tảng phần mềm. Cuối cùng, nó có thể cung cấp một cơ sở để nâng cao phần mềm. Yêu cầu phần mềm thường được viết bằng ngôn ngữ tự nhiên, nhưng đặc tả yêu cầu phần mềm có thể được bổ sung bằng các mô tả chính thức hoặc gần chính thức. Lựa chọn các ký hiệu thích hợp và các khía cạnh của kiến trúc phần mềm cụ thể được mô tả chính xác hơn so với ngôn ngữ tự nhiên.

Các nguyên tắc chung là ký hiệu nên được sử dụng cho phép các yêu cầu để được mô tả là chính xác càng tốt. Điều này đặc biệt quan trọng đối với các phần mềm an toàn cao và một số loại phần mềm đáng tin cậy khác. Tuy nhiên, sự lựa chọn của các ký hiệu thường được hạn chế bởi việc đào tạo, kỹ năng, và sở thích của các tác giả và độc giả.

Một số chỉ tiêu chất lượng đã được phát triển có thể được sử dụng liên quan đến chất lượng của đặc tả yêu cầu phần mềm như chi phí, hài lòng, hiệu quả, đúng tiến độ, và tái sản xuất. Chỉ tiêu chất lượng cho đặc tả yêu cầu của cá nhân bao gồm mệnh lệnh, chỉ thị, các pha yếu, tùy chọn, và sự duy trì. Các chỉ số cho các tài liệu đặc tả yêu cầu phần mềm bao gồm kích thước, dễ đọc, đặc tả kỹ thuật, chiều sâu và cấu trúc văn bản.

Đặc tả yêu cầu phần mềm bao gồm kích thước, dễ đọc, đặc tả kỹ thuật, chiều sâu và cấu trúc văn bản.

1.6. Thẩm định yêu cầu

Tất cả tài liệu yêu cầu cần được thông qua quá trình thẩm định và kiểm duyệt. Vậy Thẩm định yêu cầu là gì?

Khái niệm

- Thẩm định yêu cầu quan tâm đến việc chứng tỏ rằng các yêu cầu định nghĩa được hệ thống mà khách hàng thực sự muốn. Các yêu cầu phải được thẩm định để đảm bảo rằng người thực thi, người lập trình hiểu được yêu cầu.
- Việc thẩm định phải đảm bảo dễ hiểu, nhất quán và hoàn thiện
- Thẩm định yêu cầu được quan tâm đến quá trình kiểm tra tài liệu yêu cầu có đảm bảo đầu ra là 1 phần mềm hoàn chỉnh, đúng đắn.

Các kỹ thuật thẩm định yêu cầu

- xem lại yêu cầu
- sử dụng phiên bản mẫu, thử nghiệm
- thẩm định mô hình
- kiểm thử chấp thuận

1.6.1. Xem lại yêu cầu

Có lẽ phương tiện phổ biến nhất của việc thẩm định là sự kiểm tra hoặc xem lại các tài liệu yêu cầu. Một nhóm người được giao nhiệm vụ tìm các lỗi, sai sót, thiếu sự rõ ràng, và độ lệch so với tiêu chuẩn. Thành phần của nhóm này đặc biệt quan trọng (ít nhất cần có đại diện khách hàng để có được định hướng đúng đắn), và họ có thể cung cấp sự hướng dẫn trong việc tìm kiếm thông tin chuẩn xác. Việc nhận xét có thể được thành lập sau khi hoàn thành các tài liệu định nghĩa hệ thống, các tài liệu đặc tả hệ thống, đặc tả cơ bản cho 1 phiên bản mới sắp phát hành, hoặc bất kì bước nào khác trong quá trình làm yêu cầu. Ví dụ. Công ty đưa ra 1 tiêu chuẩn chung khi làm tài liệu, các kí hiệu, mô hình đối tượng, cần phải được thống nhất, Nhưng 1 nhân viên mới vào chưa nắm rõ được các tiêu chuẩn này nên làm sai 1 số chỗ. Phương pháp xem lại yêu cầu sẽ giúp tìm ra sai sót này giúp đồng nhất trong quá trình viết tài liệu.

1.6.2. Sử dụng phiên bản mẫu, thử nghiệm

Sử dụng phiên bản mẫu, thử nghiệm là dùng một mô hình chạy được của hệ thống để kiểm tra các yêu cầu. Ưu điểm của phương pháp này nhằm giúp dễ dàng trong việc giải thích những yêu cầu của phần mềm, giúp khách hàng có những phản hồi kịp thời để làm rõ hệ

thống đang sai ở đâu, cũng như phát hiện ra những yêu cầu mới. Ví dụ Việc sử dụng mô hình giao diện người dùng (Mockup,..) giúp người lập trình cũng như khách hàng dễ hiểu, tiết kiệm hơn việc miêu tả đơn thuần dùng văn bản hoặc mô hình đồ họa. Sự biến động hay sự thay đổi yêu cầu sau khi dùng bản thử nghiệm là rất thấp bởi vì có sự thống nhất giữa người lập trình, khách hàng và các bên liên quan. Bên cạnh những ưu điểm đó, phương pháp dùng phiên bản mẫu cũng có một số nhược điểm như sau. Dùng phiên bản thử nghiệm làm phân tán sự tập trung của người dùng. Ví dụ, Phiên bản mẫu là phiên bản chưa hoàn thiện về mặt thẩm mĩ cũng như chức năng. Điều đó khiến người dùng nhầm tưởng rằng chất lượng sản phẩm có chất lượng không tốt. Dùng phiên bản mẫu có thể tốn kém hơn cho việc phát triển. Ví dụ, khách hàng quá tập trung vào chức năng của phiên bản mẫu sẽ dẫn đến lỗi phát sinh, người làm yêu cầu lại phải sửa lỗi đó. Do đó việc giải thích đây chỉ là bản mẫu, thử nghiệm là đặc biệt quan trọng. Vì vậy sẽ tránh lãng phí nguồn nhân lực.

1.6.3. Thẩm định mô hình

Thẩm định model là thẩm định lại chất lượng các model(information model, behavior model, structure model) đã được phát triển trong suốt quá trình phân tích. Ví dụ trong mô hình đối tượng, chúng ta phải kiểm tra xem liên kết giữa các đối tượng, giữa sự trao đổi dữ liệu giữa các đối tượng có chuẩn xác không. Các model phải đủ các tiêu chí: Hoàn thiện, nhất quán và chuẩn xác.

1.6.4. Kiểm thử chấp thuận

Một đặc điểm quan trọng của yêu cầu phần mềm là nó có thể kiểm định rằng sản phẩm cuối cùng phải thỏa mãn các yêu cầu. Viết các testcase dành cho các yêu cầu để kiểm tra khả năng đáp ứng được các yêu cầu end-user. Sản phẩm cuối cùng phải thỏa mãn các testcase

1.7. Khảo sát hiện trạng

Thực trạng có rất nhiều thay đổi, do đó quản lý những thay đổi và duy trì những yêu cầu là chìa khóa quyết định sự thành công của phần mềm. Ví dụ : không phải tổ chức nào cũng có thói quen làm tài liệu cũng như quản lý yêu cầu đặc biệt với những công ty mới thành lập hoặc những công ty có nguồn nhân lực hạn chế. Khi những công ty này mở rộng, số lượng khách hàng trở lên lớn hơn, sản phẩm của họ bắt đầu phát triển, thì lúc này việc tìm lại những yêu cầu và thúc đẩy thêm nhiều tính năng để đáp ứng nhu cầu thị trường và sự thay đổi của môi trường là rất cần thiết. Do đó, tài liệu yêu cầu và quản lý những thay đổi là chìa khóa cho sự thành công của bất kì quá trình yêu cầu nào.

1.7.1. Quản lý thay đổi

Quản lý thay đổi là trung tâm của quản lý yêu cầu. Yêu cầu phần mềm luôn luôn thay đổi:
+Môi trường doanh nghiệp và kĩ thuật thay đổi: Phần cứng mới => giao diện mới Ví dụ: Màn hình thay đổi, sắc nét hơn nên giao diện cũng phải chăm chút hơn. Luật thay đổi, nhu cầu doanh nghiệp thay đổi => thay đổi chức năng Ví dụ: Luật doanh nghiệp không cho tiết lộ thông tin người dùng. Do đó hệ thống quản lí user cần phải bảo mật hơn

- Khách hàng, người sử dụng thay đổi => Thay đổi chức năng Ví dụ: Khách hàng muốn tạo chức năng đặt lịch gửi mail. Do đó phần mềm cần phải có chức năng email schedual.
- Xung đột giữa các yêu cầu mới誕生, và giữa yêu cầu mới với yêu cầu cũ => Quản lý thay đổi là trung tâm và đặc biệt quan trọng của quản lý các yêu cầu.

1.7.2. Định danh yêu cầu

- Yêu cầu bao gồm không chỉ đặc tả hệ thống mà còn những thông liên quan, giúp dễ dàng quản lí và diễn giải yêu cầu
- Định danh yêu cầu cần được định nghĩa, ghi nhận và cập nhật như phần mềm trong quá trình phát triển và bảo trì.
- Bao gồm:
 - việc phân loại yêu cầu Ví dụ: functional requirement and non-functional requirement
 - phương pháp xác minh Ví dụ: . xác minh bằng cách xem lại yêu cầu . xác minh bằng sử dụng phiên bản mẫu, thử nghiệm
 - thâm định mô hình
 - kiểm thử chấp thuận
 - kế hoạch kiểm thử
 - thuộc tính duy nhất để thuận tiện cho việc tham chiếu giữa các yêu cầu và lần vết. Ví dụ: thông tin của yêu cầu duy nhất để tiện cho việc thêm và chỉnh sửa sau khi có sự thay đổi.

1.7.3. Lần vết yêu cầu

- Lần vết yêu cầu có liên quan với việc khôi phục nguồn của yêu cầu và dự đoán những ảnh hưởng của các yêu cầu.
- Truy vết để thực hiện phân tích những ảnh hưởng khi yêu cầu thay đổi.
- một yêu cầu nên được truy về những yêu cầu khác và các bên liên quan để thúc đẩy nó(ví dụ: từ yêu cầu phần mềm về yêu cầu hệ thống)
- Ngược lại. một yêu cầu nên được truy đến những yêu cầu khác nhằm thỏa mãn nó(ví dụ: từ yêu cầu hệ thống đến yêu cầu phần mềm)

1.7.4. Đánh giá yêu cầu

- Đánh giá kích thước của sự thay đổi yêu cầu.
 - Ví dụ: đánh giá độ khó khăn của việc triển khai thay đổi yêu cầu, đánh giá xem nó ảnh hưởng tới những phân hệ nào trong hệ thống.
- Đánh giá chi phí cho việc phát triển hoặc duy trì 1 yêu cầu
 - Ví dụ: đánh giá nguồn lực, chi phí, mandate cho việc thay đổi yêu cầu.

1.8. Công cụ sử dụng trong việc làm yêu cầu phần mềm

- Công cụ cho việc vẽ model (CASE tool , starUML)
- Công cụ cho việc quản lý yêu cầu (spreadsheet, cơ sở dữ liệu,...)

Thiết kế phần mềm

Giới thiệu

Khái niệm thiết kế được định nghĩa theo 2 cách sau:

- **Thiết kế** là quy trình định nghĩa ra kiến trúc, thành phần, interfaces và các thuộc tính khác của một hệ thống hoặc một thành phần. Trong quy trình này, yêu cầu phần mềm được phân tích để đưa ra một cấu trúc của phần mềm làm cơ sở để làm ra phần mềm.
- **Thiết kế** là kết quả của một quá trình. Nó mô tả kiến trúc của một phần mềm như là phần mềm được phân rã và tổ chức như thế nào trong các thành phần và các interfaces giữa các thành phần như thế nào. Nó cũng có thể mô tả các thành phần ở mức chi tiết cho phép có thể xây dựng được phần mềm.

Thiết kế phần mềm đóng vai trò quan trọng: trong suốt quá trình thiết kế, những kỹ sư phần mềm sẽ đề xuất các mô hình tạo thành loại kế hoạch chi tiết cho giải pháp để có thể thực hiện được. Chúng ta có thể phân tích và đánh giá những mô hình này có hay không phù hợp với những yêu cầu khác nhau. Chúng ta có thể sử dụng kiểm tra và thẩm định thay thế những giải pháp và những đánh đổi (tradeoffs). Cuối cùng chúng ta sử dụng những mô hình kết quả để lên kế hoạch các hoạt động phát triển tiếp theo như là: thẩm định và kiểm thử hệ thống, thêm vào đó sử dụng chúng như là đầu vào hay là điểm bắt đầu của xây dựng và kiểm thử phần mềm.

Trong danh sách chuẩn của vòng đời phát triển phần mềm như ISO/IEC/IEEE Software Life Cycle Process, thiết kế phần mềm bao gồm 2 hoạt động tương ứng với phân tích yêu cầu phần mềm và xây dựng phần mềm:

- **Thiết kế kiến trúc** (còn được gọi là thiết kế mức cao): là phát triển mức kiến trúc cao nhất và đưa ra cách tổ chức phần mềm và chỉ ra các thành phần khác nhau trong phần mềm
- **Thiết kế chi tiết**: chỉ ra chi tiết và đầy đủ về thành phần tạo điều kiện xây dựng phần mềm trong pha sau đó

Đầu ra của thiết kế phần mềm sẽ được sử dụng cho quá trình xây dựng và kiểm thử nên việc đánh giá một thiết kế có phù hợp hay không rất quan trọng, nếu một thiết kế sai sẽ dẫn đến tất cả các quá trình sau đó cũng sai và cần phải chỉnh sửa nếu thiết kế được chỉnh sửa.

1. Nguyên tắc thiết kế phần mềm cơ bản

Phần này đưa ra khái niệm, quan niệm và thuật ngữ hình thành nền tảng cơ bản để hiểu biết về vai trò và phạm vi của thiết kế phần mềm

1.1 Khái niệm thiết kế chung

Theo nghĩa chung, thiết kế có thể được xem như là một hình thức giải quyết vấn đề. Thiết kế là một quá trình áp dụng nhiều kỹ thuật và các nguyên lý để tạo ra mô hình của một thiết bị, một tiến trình hay một hệ thống đủ chi tiết mà theo đó có thể chế tạo ra sản phẩm vật lý tương ứng với nó. Mục tiêu thiết kế là để tạo ra một mô hình biểu diễn của một thực thể mà sau này sẽ được xây dựng

1.2 Bối cảnh của thiết kế phần mềm

Thiết kế phần mềm là một phần quan trọng của quy trình phát triển phần mềm. Để hiểu vai trò của thiết kế phần mềm, chúng ta có thể nhìn và vòng đời phát triển phần mềm để thấy nó là một thành phần gắn với vòng đời này.

1.3 Quy trình thiết kế phần mềm

Thiết kế phần thường được xem như là một quy trình 2 bước:

- Thiết kế kiến trúc (cũng được xem như là thiết kế mức cao high-level design or top-level design) mô tả phần mềm được tổ chức thành các thành phần như thế nào
- Thiết kế chi tiết mô tả hành động mong muốn của những thành phần

Đầu ra của 2 quy trình này là tập mô hình và tài liệu ghi lại những quyết định quan trọng đã được thực hiện cùng lời giải thích cho mỗi lý do. Bằng cách ghi lại các lý do đó công việc bảo trì dài hạn của phần mềm được nâng cao

1.4 Nguyên tắc thiết kế phần mềm

Nguyên tắc là “một giả định, giáo lý hoặc luật căn bản và toàn diện”. **Nguyên tắc thiết kế phần mềm** là quan niệm chính cung cấp kiến thức cơ bản cho khái niệm và hướng tiếp cận thiết kế phần mềm khác nhau. Nguyên tắc thiết kế phần mềm bao gồm: trừu tượng hóa (abstraction); ghép nối và liên kết (coupling and cohesion); phân rã và modul hóa (decomposition and modularization); đóng gói/ẩn thông tin (encapsulation/information hiding); tách giao diện và thực hiện (separation of interface and implementation); đầy đủ, toàn vẹn và nguyên thủy (sufficiency, completeness, and primitiveness); và tách mối quan tâm (separation of concerns)

- **Abstraction:** là một cách nhìn của một đối tượng mà tập trung vào thông tin liên quan để cụ thể hóa mục đích và tránh bỏ xót thông tin". Trong bối cảnh của thiết kế phần mềm, 2 cơ chế trừu tượng hóa chìa khóa là tham số hóa và cụ thể hóa. Trừu tượng bởi tham số hóa trừu tượng đến từ biểu diễn dữ liệu chi tiết bởi biểu diễn dữ liệu như tên những tham số. Trừu tượng hóa bởi cụ thể hóa dẫn đến 3 loại trừu tượng: trừu tượng thủ tục, trừu tượng dữ liệu và trừu tượng điều khiển (trừu tượng tương tác lẫn nhau)
 - Trừu tượng thủ tục (trừu tượng hàm) cung cấp cơ chế để trừu tượng những thủ tục dễ định nghĩa hoặc những thao tác thành những thực thể. Trừu tượng thủ tục đã được áp dụng rộng rãi và các ngôn ngữ lập trình hầu như tất cả đều cung cấp hỗ trợ khái niệm này (ví dụ pascal, java,...)
 - Trừu tượng dữ liệu: đây là nguyên tắc chính trong hướng đối tượng. Trong kiểu trừu tượng này, thay vì chỉ tập trung vào thao tác, chúng ta tập trung và dữ liệu đầu tiên và sau đó những thao tác tác động lên dữ liệu. Một ví dụ đơn giản là queue data và những thao tác liên quan như add() and delete(). Trong trừu tượng hóa thủ tục, thao tác add và delete chỉ là riêng biệt không có quan hệ với dữ liệu. Điểm mạnh của trừu tượng hóa dữ liệu so với trừu tượng hóa thủ tục là dữ liệu và các thao tác liên quan được đưa ra vì vậy rất dễ để sửa code khi dữ liệu thay đổi.
 - Trừu tượng điều khiển liên quan để sử dụng các chương trình con và liên quan đến luồng điều khiển.
- **Coupling and Cohesion:** Ghép nối là một độ đo của độ phụ thuộc lẫn nhau giữa các module trong chương trình máy tính, trong khi đó liên kết là độ đo độ mạnh của mối liên kết giữa các phần tử trong một module.
- **Phân rã hóa và module hóa:** Phân rã hóa và modul hóa nghĩa là phần mềm lớn được chia thành một số thành phần định danh (đã định nghĩa interface) mà mô tả tương tác giữa các thành phần. Thông thường mục tiêu là thay thế những chức năng và trách nhiệm trong những thành phần khác nhau.
- **Đóng gói và ẩn thông tin:** nghĩa là nhóm và đóng gói chi tiết bên trong của một trừu tượng và làm cho những chi tiết không thể được truy cập từ bên ngoài
- **Tách giao diện và thực hiện:** liên quan đến việc xác định một thành phần bằng cách xác định một giao diện public (được biết đến như là client) mà là tách từ chi tiết của thành phần được hiện thực hóa như thế nào.
- **Tính đầy đủ, toàn vẹn và nguyên thủy:** mục tiêu của tính đầy đủ, toàn vẹn và nguyên thủy nghĩa là chắc rằng một thành phần chỉ tương ứng với những đặc điểm quan trọng của một trừu tượng. Nguyên thủy nghĩa là thiết kế nên được dựa trên mô hình dễ thực hiện

- **Tách mối quan tâm.** Một mối quan tâm là một “khu vực quan tâm với sự liên quan đến thiết kế phần mềm”. Một mối quan tâm thiết kế là một lĩnh vực của thiết kế mà liên quan đến một hay nhiều các bên liên quan (stakeholders). Mỗi kiến trúc nhìn một hay nhiều khung nhìn quan tâm. Tách mối quan tâm bởi những khung nhìn cho phép quan tâm các bên liên quan (stakeholders) tập trung vào một việc tại một thời điểm và yêu cầu và cung cấp một phương tiện quản lý phức tạp.

2. Những vấn đề chính trong thiết kế kiến trúc phần mềm

Một số vấn đề quan trọng phải được xử lý trong khi thiết kế phần mềm. Đặc biệt là những lo ngại về chất lượng phần mềm mà có thể kể đến như: hiệu suất, bảo mật, độ tin cậy, khả năng sử dụng, vv... Một số vấn đề quan trọng khác là làm thế nào để phân rã, tổ chức và đóng gói những thành phần phần mềm. Đây là nguyên tắc cơ bản mà tất cả các phương pháp thiết kế phải giải quyết nó bằng cách này hay cách khác. Ngược lại, những vấn đề “liên quan đến các khía cạnh của các hành vi phần mềm mà lại không nằm trong miền ứng dụng mà nằm ở các miền khác có liên quan” Những vấn đề đó thường xuyên chồng chéo với các chức năng của hệ thống và được gọi những khía cạnh mà đa phần không phải là đơn vị phân rã của phần mềm mà là thuộc tính ảnh hưởng đến hiệu suất hoặc ngữ nghĩa của các thành phần một cách có hệ thống.

2.1 Đồng thời (concurrency)

Đồng thời là nhiều việc xảy ra tại cùng một thời điểm. Thiết kế để có tính đồng thời có liên quan đến phân rã phần mềm thành quy trình, nhiệm vụ, quá trình và đối phó với các vấn đề liên quan đến tính hiệu quả, tính nguyên tố, đồng bộ hóa và lập kế hoạch. Thiết kế này đảm bảo dễ phân chia công việc cũng như có thể hoàn thành công việc trong thời gian ngắn nhất.

2.2 Điều khiển và xử lý các sự kiện

Vấn đề thiết kế này liên quan tới làm thế nào tổ chức dữ liệu và dòng dữ liệu cũng như làm thế nào để xử lý các sự kiện tạm thời và phản xạ qua các cơ chế lời gọi ngầm và gọi lại.

2.3 Dữ liệu bền vững (data persistence)

Vấn đề của thiết kế này liên quan tới làm thế nào để xử lý dữ liệu tồn tại lâu dài

2.4 Phân phối các thành phần

Vấn đề của thiết kế này liên quan đến làm thế nào để phân phối các phần mềm trên phần cứng (bao gồm phần cứng máy tính và phần cứng mạng), làm thế nào các thành phần giao tiếp được với nhau, và làm thế nào tầng giữa có thể được sử dụng để đối phó với không tương thích phần mềm.

2.5 Lỗi và xử lý ngoại lệ và lỗi dung nạp (error and exception handling and fault tolerance)

Vấn đề của thiết kế này liên quan đến làm thế nào để phòng chống, chịu đựng và các xử lý lỗi và đối phó với các điều kiện ngoại lệ

2.6 Tương tác và trình bày (Interaction and presentation)

Vấn đề thiết kế này liên quan tới làm thế nào để cấu trúc và tổ chức tương tác với những người dùng và biểu diễn thông tin (ví dụ, chia giao diện và khung nhìn logic sử dụng hướng tiếp cận MVC)

Chú ý rằng chủ đề này không chỉ chi tiết giao diện người dùng, đó là nhiệm vụ của thiết kế giao diện người dùng

2.7 Bảo mật (security)

Thiết kế cho bảo mật liên quan đến làm thế nào để ngăn chặn tiết lộ trái phép, sáng tạo, thay đổi, xóa, hoặc từ chối truy cập đến thông tin từ các nguồn khác. Nó cũng quan tâm làm thế nào để chịu được các cuộc tấn công bảo mật hoặc sự xâm phạm bởi hạn chế thiệt hại, tiếp tục dịch vụ, tốc độ sửa chữa và phục hồi, và thất bại và phục hồi an toàn. Kiểm soát truy cập là một khái niệm an ninh cơ bản và ta cũng nên đảm bảo sử dụng đúng mật mã

3. Kiến trúc và cấu trúc phần mềm

Một kiến trúc phần mềm là “tập hợp các cấu trúc cần thiết để suy luận về hệ thống, trong đó bao gồm các yếu tố phần mềm, mối quan hệ giữa chúng và đặc tính của cả hai. Trong suốt những năm 1990, kiến trúc phần mềm bắt đầu nổi lên như một ngành học rộng liên quan đến việc nghiên cứu các cấu trúc phần mềm và kiến trúc theo một cách chung. Điều đó dẫn đến một số khái niệm thú vị về thiết kế phần mềm ở những mức độ khác nhau của trừu tượng hóa. Mọi số khái niệm có thể hữu ích trong việc thiết kế kiến trúc (ví dụ phong cách kiến trúc) cũng như trong suốt quá trình thiết kế chi tiết (ví dụ design pattern). Những khái niệm thiết kế này cũng được sử dụng để thiết kế những chương trình tương tự.

3.1 Cấu trúc và góc nhìn

Khía cạnh mức cao khác nhau của thiết kế phần mềm có thể được mô tả và tài liệu hóa. Những khía cạnh này thường được gọi là các góc nhìn “Một góc nhìn biểu diễn một phần khía cạnh của kiến trúc phần mềm mà biểu diễn cụ thể chính xác của hệ thống phần mềm”. Các góc nhìn thích hợp với những vấn đề khác nhau liên quan đến phần mềm – ví dụ, góc nhìn logic (đáp ứng các yêu cầu chức năng) với góc nhìn tiến trình (vấn đề đồng thời) với góc nhìn vật lý (vấn đề phân phối) với góc nhìn phát triển (làm thế nào để thiết kế được break down thành các thành phần đơn vị với đại diện rõ ràng của sự phụ thuộc giữa các đơn vị). Nhiều tác giả sử dụng những thuật ngữ khác nhau- như hành vi, chức năng, cấu trúc, góc nhìn mô hình dữ liệu. Tóm lại, thiết kế phần mềm là một sản phẩm nhiều góc nhìn được tạo bởi quy trình thiết kế và quan điểm độc lập tương đối và trực giao.

3.2 Kiểu kiến trúc

Kiểu kiến trúc là một chuyên môn hóa của phần tử và các loại liên quan, cùng với một bộ những hạn chế về cách nó có thể được sử dụng. Một vài tác giả chỉ ra một số kiểu kiến trúc chính như sau:

- Kiến trúc thường (ví dụ, phân tầng, pipes and filter, blackboard)
- Các hệ thống phân tán (ví dụ client- server, three- tiers, broker)
- Các hệ thống tương tác (ví dụ, MVC, Presentation- Abstraction- Control, WPF)
- Các hệ thống mô phỏng (ví dụ, microkernel, reflection)
- Các kiểu khác (ví dụ, batch, interperters, process control, rule- based)

3.3 Mẫu thiết kế (Design Patterns)

Mẫu là một giải pháp phổ biến để giải quyết các vấn đề phò biến trong ngữ cảnh đưa ra. Trong khi kiểu kiến trúc có thể được nhìn như mẫu mô tả tổ chức mức cao của phần mềm, mẫu thiết kế có thể sử dụng mô tả cụ thể ở mức thấp. Những mẫu thiết kế mức thấp bao gồm:

- Mẫu tạo (ví dụ, builder, factory, prototype, singleton)
- Mẫu cấu trúc (ví dụ, adapter, bridge, composite, decorator, façade, fly- weight, proxy)
- Mẫu hành vi (ví dụ, command, interperter, iterator, mediator, memento, observer, state, strategy, template, visitor)

3.4 Những quyết định thiết kế kiến trúc

Thiết kế kiến trúc là một quá trình sáng tạo. Trong suốt quy trình thiết kế, nhà thiết kế phần mềm phải tạo một số quyết định cơ bản ảnh hưởng sâu sắc tới các phần mềm và quy trình phát triển phần mềm. Nên nghĩ rằng thiết kế kiến trúc tạo thành từ quan điểm quyết định hơn là quan điểm hoạt động. Thông thường, tác động vào chất lượng thuộc tính và hoán đổi giữa các thuộc tính cạnh tranh là cơ sở cho quyết định thiết kế

3.5 Tương tự giữa chương trình và framework

Một cách tiếp cận cung cấp cho việc sử dụng lại thiết kế phần mềm và thành phần là sử dụng những chương trình tương tự. Điều này có thể thực hiện bằng xác định sự tương đồng giữa các phần mềm bằng cách thiết kế các thành phần tái sử dụng và tùy vào sự khác nhau giữa các phần mềm. Trong lập trình hướng đối tượng, một khái niệm chìa khóa có liên quan đến khung là một khung: một phần hệ thống phần mềm hoàn toàn có thể được mở rộng bằng cách cài đặt các công cụ thích hợp

4. Thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một phần quan trọng quá trình thiết kế phần mềm. Thiết kế giao diện và xử lý tương tác với người sử dụng là một yếu tố quan trọng trong việc sử dụng phần mềm. Người thiết kế phải làm sao để phù hợp với kỹ năng, kinh nghiệm và mong đợi từ phía người sử dụng phần mềm.

4.1: Nguyên tắc cơ bản trong thiết kế giao diện

- **Dễ học:** Phần mềm cần phải dễ học cách sử dụng, do đó người dùng có thể nhanh chóng bắt đầu làm việc sử dụng phần mềm đó
- **Quen thuộc với người sử dụng:** Giao diện nên dùng các thuật ngữ và khái niệm rút ra từ kinh nghiệm của những người sẽ dùng hệ thống nhiều nhất
- **Tính nhất quán:** giao diện cần nhất quán sao cho các thao tác gần giống nhau có thể được kích hoạt theo cùng kiểu.
- **Ngạc nhiên tối thiểu:** Người dùng không bao giờ bị bất ngờ về hành vi của hệ thống
- **Khôi phục được:** Giao diện nên có các cơ chế cho phép người dùng khôi phục lại tình trạng hoạt động bình thường sau khi gặp lỗi
- **Hướng dẫn người dùng:** Giao diện nên có phản hồi có nghĩa khi xảy ra lỗi và cung cấp các tiện ích trợ giúp theo ngữ cảnh
- **Người dùng đa dạng:** Giao diện nên cung cấp các tiện ích tương tác thích hợp cho các loại người dùng hệ thống khác nhau

4.2: Vấn đề trong thiết kế giao diện

Hai vấn đề cần xem xét:

- *Người dùng cung cấp thông tin cho hệ thống bằng cách nào?*
- *Hệ thống nên trình bày thông tin (output) cho người dùng như thế nào?*

4.3: Các kiểu tương tác

Các kiểu tương tác phổ biến:

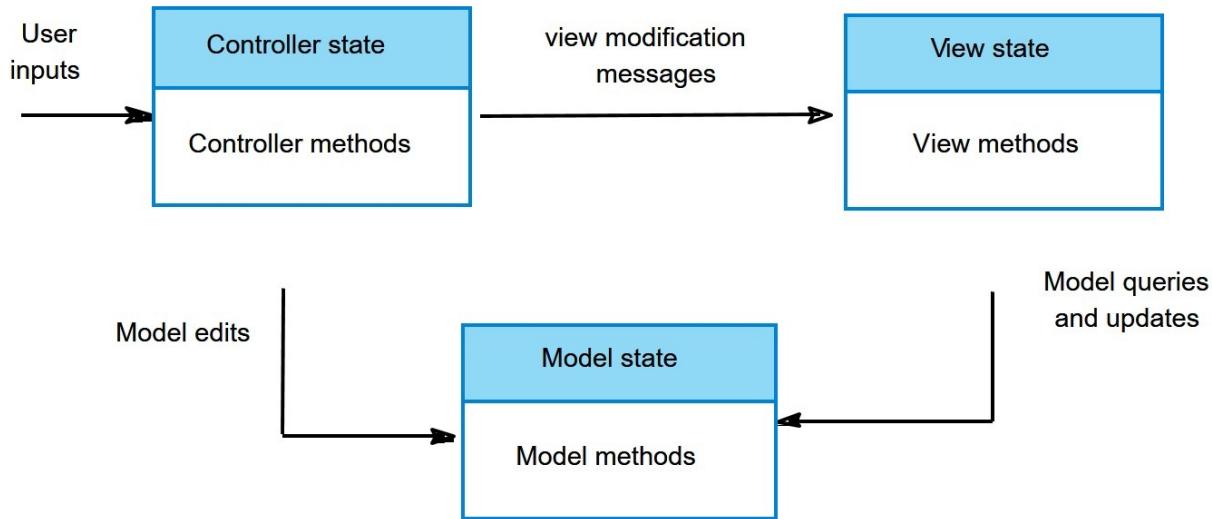
- **Thao tác trực tiếp** – Direct manipulation
- **Chọn lựa bằng menu** – Menu selection
- **Điền form** – Form fill-in
- **Dòng lệnh** – Command language
- **Ngôn ngữ tự nhiên** – Natural language

Kiểu tương tác	Ưu điểm chính	Nhược điểm chính	Ví dụ
Thao tác trực tiếp	Tương tác trực quan, nhanh chóng và dễ hiểu	Có thể khó cài đặt. Chỉ thích hợp khi có ẩn dụ hình ảnh cho các tác vụ và đối tượng	Trò chơi điện tử và các ứng dụng có drag & drop
Chọn lựa bằng menu	Tránh lỗi cho người dùng, không phải làm nhiều thao tác	Chậm chạp với người sử dụng có kinh nghiệm. Có thể phức tạp nếu có nhiều lựa chọn menu	Đa số các hệ thống thông dụng
Điền form	Nhập dữ liệu đơn giản, dễ học, có thể kiểm tra được	Tốn không gian hiển thị, rắc rối khi lựa chọn của người dùng không khớp với kiểu dữ liệu của form	Đăng ký thông tin cá nhân, khai thuế
Dòng lệnh	Mạnh và linh động	Khó học, xử lý lỗi kém	Terminal, Autocad
Ngôn ngữ tự nhiên	Đáp ứng được người dùng không chuyên, dễ mở rộng	Cần gõ nhiều, các hệ thống hiểu ngôn ngữ tự nhiên không đáng tin cậy	Trợ lý ảo

4.4: Biểu diễn thông tin

Thông tin có thể được trình bày trực tiếp (ví dụ text trong một trình soạn thảo) hoặc được biến đổi thành một dạng biểu diễn khác (ví dụ dạng đồ họa)

Model-View-Controller là cách tiếp cận hỗ trợ nhiều kiểu biểu diễn dữ liệu



Có 2 loại thông tin cần được biểu diễn:

- **Thông tin tĩnh**: Tạo ở lúc bắt đầu và không thay đổi trong phiên làm việc.
- **Thông tin động**: Thay đổi trong phiên làm việc và phải thông báo cho người sử dụng

Các kỹ thuật hiển thị lượng lớn thông tin:

- **Hình ảnh**: có thể cho thấy quan hệ giữa các thực thể và các xu hướng của dữ liệu
- **Màu sắc**: thường dùng để *highlight* các thông tin đặc biệt

Hướng dẫn về việc sử dụng màu sắc:

- Hạn chế số màu và mức độ sặc sỡ
- Dùng sự thay đổi màu để báo hiệu thay đổi trạng thái hệ thống.
- Dùng kí hiệu màu (color coding) để hỗ trợ công việc người dùng đang cố làm. Highlight những điểm người dùng cần chú ý.
- Dùng kí hiệu màu một cách cẩn trọng và nhất quán
- Cẩn thận về hiệu ứng cặp đôi của màu sắc. Một số tổ hợp màu gây khó đọc, ví dụ như người ta không thể cùng lúc chú ý cả hai màu đỏ và xanh lam

4.5: Quy trình thiết kế giao diện

Thiết kế giao diện là một quy trình lặp đi lặp lại với sự liên lạc chặt chẽ giữa người dùng và người thiết kế. Ba hoạt động chính trong quy trình:

- **User analysis:** Tìm hiểu người dùng sẽ làm gì với hệ thống;
- **System prototyping:** phát triển một loạt các bản mẫu để thử nghiệm
- **Interface evaluation:** thử nghiệm các bản mẫu cùng với người dùng

4.6: Quốc tế hóa và địa phương hóa

Trong quá trình thiết kế cần phải xem xét đến việc ngôn ngữ theo chuẩn quốc tế và chuẩn địa phương. Tức là giao diện phần mềm có thể thích ứng với sự khác nhau về khu vực, ngôn ngữ và yêu cầu kỹ thuật của thị trường. Quốc tế hóa là quá trình thiết kế một ứng dụng bao gồm nhiều ngôn ngữ để có thể thích nghi với những khu vực không có sự thay đổi quan trọng về quy trình. Địa phương hóa là sự thích ứng của quốc tế hóa với một khu vực hoặc ngôn ngữ cụ thể bằng cách thêm vào các thành phần của địa phương và dịch văn bản. Các yếu tố cần được quan tâm như biểu tượng, số, tiền tệ, thời gian và các đơn vị đo lường.

4.7: Biểu tượng và khái niệm quen thuộc

Người thiết kế giao diện sử dụng các biểu tượng và khái niệm để tạo được sự quen thuộc giữa các phần mềm với những hệ thống đã được biết đến trên thế giới. Như vậy người dùng sẽ dễ dàng hơn trong việc tìm hiểu và sử dụng giao diện.

Ví dụ: Chức năng xóa tập tin có thể gắn vào biểu tượng thùng rác.

Khi thiết kế giao diện, nhà thiết kế không được sử dụng nhiều hơn một ý nghĩa, hay chức năng trong một biểu tượng. Bằng cách sử dụng những biểu tượng và khái niệm quen thuộc, hỗ trợ biểu diễn tốt hơn các thông tin cần được quốc tế hóa, nhưng cần phải chú ý không nên áp dụng theo cùng một cách với tất cả các địa phương và khu vực khác nhau.

5. Phân tích và đánh giá chất lượng thiết kế phần mềm

Phần này gồm các phân tích và đánh giá chất lượng trong thiết kế phần mềm:

- Cần thực hiện một số đánh giá UI để đánh giá mức độ thích hợp
- Đánh giá đầy đủ và toàn bộ thì quá đắt và không thực tế cho hầu hết các hệ thống
- Một giao diện cần được đánh giá theo một đặc tả về tính sử dụng.

5.1: Các thuộc tính về tính sử dụng

Thuộc tính	Mô tả
Khả năng học	Người dùng mới cần bao lâu để có thể hoạt động hiệu quả với hệ thống?
Tốc độ vận hành	Tốc độ phản ứng của hệ thống có đáp ứng tốt công việc của người dùng?
Chiu lỗi	Mức độ dung thứ lỗi của hệ thống đối với lỗi người dùng.
Khả năng khôi phục	Khả năng hệ thống khôi phục từ lỗi của người dùng.
Tương thích	Hệ thống gắn bó chặt chẽ với một kiểu làm việc đến đâu?

5.2: Kĩ thuật đánh giá và phân tích

- Câu hỏi điều tra để lấy phản hồi của người dùng.
- Quay video về việc sử dụng hệ thống rồi sau đó đánh giá nội dung.
- Cài các đoạn mã thu thập thông tin về các tiện ích được sử dụng và lỗi của người dùng.
- Phần mềm có chức năng thu thập phản hồi trực tuyến của người dùng.

5.3: Biện pháp

Các biện pháp có thể được sử dụng để phân tích và đánh giá các khía cạnh khác nhau của việc thiết kế phần mềm. Ví dụ: cấu trúc, chất lượng, kích thước, hiệu năng, ...

Các biện pháp này được chia thành 2 loại:

- **Chức năng dựa trên thiết kế:** Chức năng được xây dựng bằng cách phân tích các giao diện của hệ thống,
- **Thiết kế hướng đối tượng:** Sử dụng lớp các đối tượng làm trung tâm, từ đó xây dựng ra chức năng và giao diện

6. Quy ước thiết kế phần mềm

Tại sao cần có quy ước, ký hiệu chung hay tạo các mô hình trong thiết kế phần mềm?

- Truyền tải được nhiều thông tin về phần mềm đến cho người đọc hơn là dùng những dữ liệu thô như văn bản mô tả phần mềm.
- Mô hình giúp chúng ta tổ chức, trình bày trực quan, thấu hiểu và tạo nên các hệ thống phức tạp
- Tất cả mọi người cùng hiểu được phần mềm được xây dựng và hoạt động như thế nào.

Các loại ký hiệu được sử dụng trong thiết kế phần mềm:

1. Ký hiệu dùng trong thiết kế kiến trúc, tổ chức của phần mềm (**static view**).
2. Ký hiệu dùng cho quá trình thiết kế chi tiết, hành vi của phần mềm (**dynamic view**).

6.1. Mô hình tĩnh (Static view)

Những ký hiệu, mô hình này được dùng trong phân rã mức cao của thiết kế phần mềm, tức là mô tả cấu trúc, các thành phần chính của phần mềm và sự kết nối giữa chúng. Dưới đây là một số mô hình thường dùng trong thiết kế phần mềm ở mức cao.

- **Ngôn ngữ đặc tả kiến trúc (Architecture description languages - ADLs)**: được sử dụng để mô tả một kiến trúc phần mềm. Có nhiều ngôn ngữ ADL khác nhau được phát triển bởi các tổ chức như Wright (được phát triển bởi Carnegie Mellon), ACME (Carnegie Mellon), xADL (UCI), Darwin (Imperial College London), DAOP-ADL (Trường đại học Málaga - Tây Ban Nha). Các thành phần cơ bản của một ngôn ngữ ADL là thành phần, kết nối và cấu hình hệ thống.
- **Biểu đồ lớp (Class and object diagrams)**: mô tả quan sát tĩnh của hệ thống thông qua các lớp và các mối quan hệ của chúng
- **Biểu đồ thành phần (Component diagrams)**: biểu đồ mô tả các thành phần và sự phụ thuộc của chúng trong hệ thống. Các thành phần của hệ thống có thể là mã nguồn, thành phần mã nhị phân (tệp mã đích, thư viện,...) và thành phần thực thi.
- **Class responsibility collaborator cards (CRCs)**: được dùng trong thiết kế hướng đối tượng, sử dụng để biểu thị tên của các thành phần (class) cùng với trách nhiệm và các thành phần hợp tác với nó. CRCs ban đầu được đề xuất bởi Ward Cunningham và Kent Beck như một công cụ giảng dạy, sau được phát triển và sử dụng trong thực tế thiết kế phần mềm.
- **Biểu đồ triển khai (Deployment diagrams)**: chỉ ra cấu hình các phần tử xử lý lúc chương trình chạy, các nút trên mạng và các tiến trình phần mềm thực hiện trên những phần tử đó. Nó chỉ ra mối quan hệ giữa các phần cứng và phần mềm của hệ thống. Biểu đồ triển khai chỉ ra toàn bộ các nút trên mạng, kết nối giữa chúng và các tiến trình chạy trên chúng.
- **Sơ đồ thực thể quan hệ (Entity-relationship diagrams - ERD)**: được sử dụng để đại diện cho mô hình khái niệm dữ liệu lưu trữ trong kho thông tin.
- **Ngôn ngữ mô tả giao diện (Interface description languages - IDLs)**: ngôn ngữ lập

trình được sử dụng để xác định các giao diện (tên và các loại xuất khẩu hoạt động) của các thành phần phần mềm.

- **Biểu đồ cấu trúc (Structure charts):** Chúng được sử dụng trong lập trình có cấu trúc để sắp xếp module của chương trình vào một cái cây. Mỗi mô-đun được đại diện bởi một cái hộp, trong đó có tên của module. Cấu trúc cây biểu thị mối quan hệ giữa các module.

6.2. Mô hình động (Dynamic view)

Có nhiều loại ký hiệu, mô hình dùng cho quá trình thiết kế chi tiết đang được áp dụng phổ biến hiện nay.

- **Sơ đồ hoạt động (Activity diagram):** Biểu đồ hoạt động tập trung vào công việc được thực hiện trong khi thực thi một thủ tục (hàm). Nó nắm bắt hành động (công việc và những hoạt động phải được thực hiện) cũng như kết quả của chúng theo sự biến đổi trạng thái.
- **Biểu đồ giao tiếp (Communication diagram):** được sử dụng để hiển thị sự tương tác xảy ra trong một nhóm của các đối tượng; trọng tâm là về đối tượng, các liên kết và những trao đổi giữa chúng.
- **Sơ đồ luồng dữ liệu (Data flow diagrams - DFDs):** một công cụ mô tả mối quan hệ thông tin giữa các đối tượng, cung cấp bức tranh tổng thể của hệ thống và một thiết kế sơ bộ về thực hiện các chức năng. Biểu đồ luồng dữ liệu có thể được sử dụng để phân tích an ninh, vì nó cung cấp các hướng có thể tấn công và tiết lộ thông tin bí mật.
- **Bảng và biểu đồ quyết định (Decision tables and diagrams):** sử dụng để mô tả sự kết hợp phức tạp của các hành động trong phần mềm.
- **Biểu đồ tiến trình / lưu đồ (Flowcharts):** Frank Gilbreth thành viên của ASME (the American Society of Mechanical Engineers) giới thiệu lần đầu năm 1921. Nó mô tả một quá trình bằng cách sử dụng những hình ảnh hoặc những ký hiệu kỹ thuật ... nhằm mô tả đầy đủ nhất đầu ra và dòng chảy của quá trình, tạo điều kiện cho việc điều tra các cơ hội cải tiến bằng việc hiểu biết chi tiết về quá trình làm việc của nó.
- **Biểu đồ tuần tự (Sequence diagrams):** minh họa các đối tượng tương tác với nhau ra sao. Chúng tập trung vào các chuỗi thông điệp được gửi và nhận giữa các đối tượng.
- **Biểu đồ trạng thái (State transition and state chart diagrams):** nắm bắt vòng đời của các đối tượng, các hệ thống con (Subsystem) và các hệ thống. Chúng cho ta biết các trạng thái mà một đối tượng có thể có và các sự kiện (các thông điệp nhận được, các khoảng thời gian đã qua đi, các lỗi xảy ra, các điều kiện được thỏa mãn) sẽ ảnh hưởng đến những trạng thái đó như thế nào dọc theo tiến trình thời gian
- **Ngôn ngữ đặc tả chính thức (Formal specification languages):** ngôn ngữ văn bản mà sử dụng các khái niệm cơ bản từ toán học (ví dụ, logic, thiết lập, trình tự) để xác định một cách chặt chẽ và trừu tượng phần mềm giao diện thành phần và hành vi.
- **Mã giả và ngôn ngữ thiết kế chương trình (Pseudo code and program design**

languages - PDLs): cấu trúc ngữ lập trình như ngôn ngữ được sử dụng để mô tả, thiết kế chi tiết, hành vi của một phần mềm hoặc phương pháp.

7. Chiến lược và phương pháp thiết kế phần mềm

Có nhiều chiến lược hỗ trợ cho quá trình thiết kế qua các phương pháp tiếp cận khác nhau. Tuy nhiên chẳng có một chiến lược nào tốt nhất cho các dự án. Hai chiến lược thiết kế đang được dùng rộng rãi và cho thấy hiệu quả tích cực là thiết kế hướng chức năng và thiết kế hướng đối tượng. Mỗi chiến lược đều có những ưu, nhược điểm riêng phụ thuộc vào ứng dụng phát triển và nhóm phát triển phần mềm. Hai cách tiếp cận này là bổ sung và hỗ trợ cho nhau chứ không đối kháng nhau.

7.1 Thiết kế hướng chức năng

Thiết kế hướng chức năng là một cách tiếp cận thiết kế phần mềm trong đó bản thiết kế được phân giải thành một bộ các mô-đun được tác động lẫn nhau, mà mỗi mô-đun có một chức năng được xác định rõ ràng.

Đây là một phương pháp cổ điển. Người ta dùng các biểu đồ dòng dữ liệu mô tả việc xử lý dữ liệu logic, các lược đồ cấu trúc để chỉ ra cấu trúc của phần mềm và mối quan hệ giữa các thành phần.

Thiết kế hướng chức năng gắn với các chi tiết của một thuật toán của chức năng nhưng các thông tin trạng thái của hệ thống không bị che dấu. Điều này có thể gây ra vấn đề khi một chức năng thay đổi trạng thái theo cách mà các chức năng khác không ngờ tới thì hệ thống sẽ trực tiếp. Do đó cách tiếp cận chức năng để thiết kế là thắng lợi nhất khi mà khối lượng thông tin trạng thái của hệ thống là nhỏ nhất và thông tin dùng chung nhau là rõ ràng nhất.

7.2. Thiết kế hướng đối tượng.

Hệ thống được nhìn nhận như một bộ các đối tượng, phân tán, mỗi đối tượng có những thông tin trạng thái riêng của nó.

- Thiết kế hướng đối tượng là dựa trên việc che dấu thông tin do dữ liệu dùng chung bị loại bỏ. Các đối tượng liên lạc với nhau bằng cách trao đổi thông báo.
- Các đối tượng là các thực thể độc lập, sẵn sàng thay đổi mà không ảnh hưởng tới các đối tượng khác.
- Các đối tượng có thể phân tán và hành động tuần tự hoặc song song.

Ưu điểm

- Dễ bảo trì và các đối tượng là độc lập.
- Có thể dùng lại một số thành phần của đối tượng đã được thiết kế trước đó.
- Thiết kế dễ hiểu: nhìn rõ được mối quan hệ giữa các thực thể

Nhược điểm

- Cách nhìn tự nhiên nhiều hệ thống là cách nhìn chức năng nên việc thích nghi với cách nhìn đối tượng đôi khi là khó khăn. Làm sao để tìm ra các đối tượng thích hợp trong một hệ thống cũng là một vấn đề khó khăn.

7.3. Thiết kế lấy cấu trúc dữ liệu làm trung tâm

Các kỹ sư phần mềm cần mô tả các đầu vào và đầu ra cấu trúc dữ liệu và sau đó phát triển cấu trúc điều khiển của chương trình dựa trên các sơ đồ cấu trúc dữ liệu.

7.4. Thiết kế hướng thành phần

Một thành phần của phần mềm là một đơn vị độc lập, có giao diện và có thể được triển khai một cách độc lập. Chiến lược thiết kế hướng thành phần dựa trên các vấn đề liên quan đến việc cung cấp, phát triển, và tích hợp các thành phần như vậy để cải thiện, tái sử dụng. Một phần mềm tái sử dụng những thành phần khác hay sử dụng phần mềm đã được dựng sẵn phải đáp ứng các yêu cầu bảo mật tương tự như phần mềm mới.

7.5. Các phương pháp khác

Ngoài các hướng thiết kế trên còn nhiều hướng thiết kế khác cũng đang được áp dụng hiện nay và mang lại những hiệu quả nhất định. Gần đây xuất hiện một kiến trúc phần mềm mới được kỳ vọng là chìa khóa giải quyết vấn đề "đơn giản hóa" phần mềm là SOA (Service-Oriented Architecture) - kiến trúc hướng dịch vụ. SOA là tập hợp các dịch vụ kết nối "mềm dẻo" với nhau, có giao tiếp được định nghĩa rõ ràng và độc lập với nền tảng hệ thống, và có thể tái sử dụng. SOA là cấp độ cao hơn của phát triển ứng dụng, chú trọng đến quy trình nghiệp vụ và dùng giao tiếp chuẩn để giúp che đi sự phức tạp kỹ thuật bên dưới. Nói cách khác, SOA là:

- Một kiểu kiến trúc phần mềm gồm nhiều thành phần độc lập được thể hiện thành những dịch vụ (service), mỗi dịch vụ thực hiện quy trình nghiệp vụ nào đó của doanh nghiệp.
- Các thành phần được nối kết qua cổng giao tiếp, có tính kế thừa các thành phần đang tồn tại, và sự tương tác giữa chúng không cần quan tâm đến việc chúng được phát triển trên nền tảng công nghệ nào. Điều này khiến hệ thống có thể mở rộng và tích hợp một cách dễ dàng.

SOA giúp tái sử dụng phần mềm, linh hoạt khi mở rộng, kết nối và tích hợp.

8. Công cụ thiết kế phần mềm

Công cụ thiết kế phần mềm có thể được sử dụng để hỗ trợ tạo ra các mô hình phần mềm trong quá trình phát triển phần mềm. Nó có thể giúp việc thiết kế phần mềm trở lên rõ ràng, linh hoạt và hiệu quả hơn.

- Chuyển các yêu cầu phần mềm thành một mô hình thiết kế trực quan.
- Cung cấp mô tả từ tổng quan đến chi tiết từng thành phần của phần mềm và giao diện của nó
- Trợ giúp cho quá trình đánh giá chất lượng phần mềm.

Một số công cụ thiết kế phần mềm đang được sử dụng nhiều hiện nay ở Việt Nam:
StarUML, Rational Rose, ...

Thành viên:

- [Phan Văn Thanh](#)
- [Lê Thị Len](#)
- [Hoàng Thị Vân Anh](#)

Chương 3: Xây dựng phần mềm

Nguyễn Việt Anh Tạ Tuấn Anh

Giới thiệu

Xây dựng phần mềm xem chi tham chiếu đến quy trình xây dựng phần mềm chi tiết dựa vào lập trình, kiểm thử, kiểm thử từng phần, kiểm thử tích hợp, tìm và sửa lỗi. Trong lĩnh vực tri thức xây dựng phần mềm (KA) được liên kết tới tất cả các lĩnh vực tri thức khác. Xây dựng phần mềm liên hệ mật thiết với Thiết kế phần mềm và Kiểm thử phần mềm bởi vì quy trình xây dựng phần mềm tham gia vào thiết kế và kiểm thử một phần mềm cụ thể. Quy trình xây dựng phần mềm sử dụng đầu và và yêu cầu đầu vào để kiểm thử. Danh giới giữa thiết kế, xây dựng và kiểm thử sẽ thay đổi phụ thuộc vào quy trình vòng đời của phần mềm được sử dụng trong một dự án.

Thông thường thiết kế chi tiết được thực hiện trước khi xây dựng và tồn tại nhiều nguồn lực trong quá trình xây dựng phần mềm. Do vậy, lĩnh vực tri thức xây dựng phần mềm có liên hệ mật thiết với lĩnh vực tri thức thiết kế phần mềm.

Qua việc xây dựng, các kỹ sư phần mềm thực hiện cả kiểm thử từng phần (unit test) và kiểm thử tích hợp (integration test). Do vậy, lĩnh vực tri thức xây dựng phần mềm liên hệ mật thiết với kiến thức kiểm thử phần mềm. Xây dựng phần mềm sản sinh ra tối đa số các tài liệu phục vụ việc quản trị trong dự án phần mềm như (các tệp mã nguồn, tài liệu dự án: tài liệu kịch bản kiểm thử, tài liệu yêu cầu,...). Do vậy kiến thức xây dựng phần mềm cũng liên hệ mật thiết đến việc quản lý cấu hình phần mềm. Trong khi chất lượng phần mềm thực sự rất quan trọng trong tất cả lĩnh vực tri thức phần mềm, do vậy kiến thức quản lý chất lượng phần mềm có liên hệ mật thiết với kiến thức xây dựng phần mềm. Từ đó xây dựng phần mềm yêu cầu kiến thức về thuật toán và kỹ năng lập trình có mối liên hệ với kiến thức thành lập máy tính và khoa học máy tính. Ngành khoa học máy tính hỗ trợ thiết kế và xây dựng sản phẩm phần mềm và cũng liên quan đến quản trị dự án giống như quản trị xây dựng phần mềm có thể coi như là những thử thách.

1. Những nguyên tắc cơ bản xây dựng phần mềm.

- Tối thiểu hóa độ phức tạp.
- Ứng phó thay đổi.
- Xây dựng kiểm thử.
- Sử dụng lại
- Các tiêu chuẩn xây dựng.

Bốn khái niêm cơ bản đầu ứng dụng trong việc thiết kế và xây dựng phần mềm. Các phần dưới đây định nghĩa các khái niêm và diễn tả chúng ứng dụng trong việc xây dựng phần mềm như thế nào.

1.1. Tối thiểu hóa độ phức tạp

Hầu hết con người bị giới hạn trong khả năng tổ nắm bắt và tổ chức cấu trúc phức tạp và thông tin trong bộ não thông qua thời gian dài. Điều này chỉ ra nguyên nhân chủ yếu ảnh hưởng đến con người chú trọng vào máy tính, coi trọng việc xây dựng phần mềm tối thiểu hóa độ phức tạp của vấn đề. Việc tối thiểu hóa độ phức tạp áp dụng trong mọi khía cạnh xây dựng phần mềm và công việc xây dựng kế hoạch kiểm thử phần mềm. Việc xây dựng phần mềm, giảm thiểu độ phức tạp tập trung vào việc viết mã nguồn đơn giản, dễ đọc hơn là việc viết chương trình minh làm tăng độ phức tạp.

1.2. Ứng phó với sự thay đổi

Hầu hết phần mềm sẽ thay đổi theo thời gian, do vậy phải có chính sách để ứng phó với nhiều hình thức thay đổi của xây dựng phần mềm, hay thay đổi môi trường có thể ảnh hưởng tiêu cực đến phần mềm. Ứng phó thay đổi giúp cho các kỹ sư phần mềm xây dựng phần có thể chỉnh sửa, cập nhật phần mềm hay mở rộng phần mềm không gây ra lỗi.

1.3. Xây dựng kế hoạch kiểm thử

Xây dựng kế hoạch kiểm thử có nghĩa rằng các lỗi dễ dàng được phát hiện bởi kỹ sư phần mềm trong khi viết chương trình, cùng với cán bộ kiểm thử trong hoạt động kiểm thử và người dùng dùng phần mềm. Các kỹ thuật cụ thể hỗ trợ việc xây dựng kế hoạch kiểm thử tiêu chuẩn viết mã nguồn, xét duyệt mã nguồn và kiểm thử từng phần, xây dựng kiểm thử tự động (automatic test) hạn chế việc phức tạp hóa mã nguồn với các cấu trúc phức tạp gây khó hiểu.

1.4. Sử dụng lại.

Sử dụng lại là muốn nói đến việc sử dụng những thứ đã có sẵn để giải quyết các vấn đề khác nhau. Trong việc xây dựng phần mềm, các phần được sử dụng lại như: các thư viện, các chương trình nhỏ (module), các thành phần (Components), mà nguồn, và các tài sản thương mại khác. Sử dụng lại được làm có hệ thống tuân theo quy trình lắp đi lắp lại đã kiểm định tốt trước đó. Sử dụng lại một cách có hệ thống có khả năng cải thiện được chất lượng, và chi phí của sản phẩm phần mềm.

Sử dụng lại được hình thành dưới hai hình thức “Xây dựng các thư viện nhằm mục đích sử dụng lại” và “Sử dụng lại các thư viện có sẵn”. Những người đi trước xây dựng ra các thư viện phần mềm cho phép sử lại và những nhà phát triển phần mềm sau này sẽ sử dụng lại

các thư viện của người đi trước để xây dựng phần mềm. Việc sử dụng lại luôn trong suốt trong quá trình phát triển của dự án điều này có nghĩa là thư viện sử dụng lại được phát triển bởi một dự án khác hay của một tổ chức khác.

Ví dụ: Trong Project 1: Website Trường học điện tử được xây dựng dựa trên các plugin có sẵn của wordpress. Đồng thời chỉnh sửa những plugin đó cho phù hợp với chức năng của hệ thống hiện thời: xuất đề dạng pdf, ...

1.5. Những tiêu chuẩn trong việc xây dựng phần mềm.

Ứng dụng các tiêu chuẩn phát triển bên trong và bên ngoài trong khi xây dựng phần mềm nhằm mục đích tăng tính hiệu quả, giảm thiểu chi phí, và tăng chất lượng của phần mềm ví dụ như việc chọn lựa ngôn ngữ lập trình ảnh hưởng đến vòng đời phát triển của phần mềm. Những vẫn đề ảnh hưởng trực tiếp tới việc xây dựng phần mềm như:

- Phương thức trao đổi.
- Ngôn ngữ lập trình.
- Những quy định, quy ước khi viết mã nguồn.
- Chọn nền tảng xây dựng.

2. Quản lý việc xây dựng phần mềm

2.1. Các quy trình chu kỳ xây dựng phần mềm

Có nhiều quy trình được xây dựng và áp dụng vào trong việc phát triển phần mềm, trong đó có một số mô hình được chú trọng phát triển hơn các mô hình khác. Một vài quy trình xây dựng phần mềm tuần tự như mô hình thác nước, mô hình bàn giao giai đoạn. Những mô hình này coi việc xây dựng giống như một hoạt động chỉ xảy ra khi các công đoạn trước đó như: lấy yêu cầu, thiết kế, và có kế hoạch chi tiết, phải được hoàn thành. Cách tiếp cận tuần tự theo nhu hướng nhấn mạnh các hoạt động mà việc xây dựng bao gồm lấy yêu cầu và thiết kế phải được thực hiện trước và là các hoạt động riêng rẽ, tách biệt. Trong những mô hình này, phần chú trọng nhất có lẽ là viết chương trình. Các mô hình khác có tính lặp đi lặp lại nhiều hơn, giống như phát triển theo mô hình agile, và mẫu. Cách tiếp cận này theo hướng coi phần xây dựng như là một hoạt động đồng thời với các hoạt động phát triển khác. Cách tiếp cận này theo cách làm đồng thời các công việc thiết kế, viết mã, kiểm thử đồng thời. Kết lại, Việc suy xét gì trong công đoạn xây dựng phụ thuộc vào khả năng vận dụng từng mô hình. Nhìn chung, việc xây dựng phát triển phần mềm chủ yếu tập trung vào việc viết mã, bẫy lỗi nhưng vẫn bao gồm các việc khác như: lên kế hoạch xây dựng phần mềm, thiết kế chi tiết, kiểm thử từng phần, kiểm thử tích hợp, v.v...

2.2. Kế hoạch xây dựng phần mềm

Chọn lựa phương pháp xây dựng là một phần quan trọng trong hoạt động lên kế hoạch xây dựng phần mềm. Việc chọn lựa phương pháp xây dựng ảnh hưởng đến chất lượng, hiệu năng hoạt động, kiến trúc và khả năng mở rộng của hệ thống cũng như đòi hỏi một cách chi tiết các công việc nào được hoàn thiện trước khi công việc xây dựng bắt đầu.

Phương thức tiếp cận để xây dựng có tác động đến toàn bộ đội dự án như: làm giảm sự phức tạp, thích nghi với thay đổi, và xây dựng kế hoạch kiểm thử. Mỗi một đầu công việc đều được đem ra nói đến trong quy trình lấy yêu cầu, thiết kế,... nhưng chúng cũng sẽ bị ảnh hưởng của bởi sự chọn lựa phương thức xây dựng.

Kế hoạch xây dựng cũng định nghĩa thứ tự thành phần nào được tạo, được tích hợp theo chiến lược (ví dụ: tích hợp tăng dần hay tích hợp theo từng giai đoạn.), các quy trình quản lý chất lượng phần mềm, việc phân chia, bàn giao nhiệm vụ cụ thể cho từng kỹ sư lập trình còn tùy thuộc vào sử dụng quy trình nào.

2.3. Đo đạc việc xây dựng

Các hoạt động xây dựng và các đối tượng có thể được đo đạc, tính toán bao gồm việc lập trình, làm mịn mã chương trình, sử dụng lại, hủy bỏ mã chương trình, độ phức tạp của mã chương trình, thống kê theo dõi mã chương trình, sửa lỗi, khả năng gây lỗi cao, nỗ lực và lên kế hoạch. Việc đo đạc này thực sự hữu ích trong việc quản lý xây dựng phần mềm đảm bảo chất lượng, cải thiện quy trình xây dựng phần sao cho hiệu quả, v.v...

3. Xem xét thực tiễn

3.1. Thiết kế xây dựng

Một vài dự án xem xét các hoạt động thiết kế tại giai đoạn xây dựng, trong khi các dự án khác chỉ tập trung tại giai đoạn thiết kế. Không nên chú trọng việc đưa ra tại giai đoạn nào chính xác, công việc thiết kế chi tiết đôi khi diễn ra tại giai đoạn xây dựng và các xu hướng thiết kế bị ảnh hưởng của các ràng buộc trong bài toán thế giới thực được mô hình hóa giải quyết bằng phần mềm.

Giống như các công nhân xây dựng thiết kế kiến trúc vật lý phải điều chỉnh phần nhỏ sao cho trình độ nhận thức giữa kế hoạch của người lập và người công nhân xây dựng ngày càng gần nhau hơn, bằng cách người xây dựng kế hoạch giải thích chi tiết cho người công nhân trong quá trình xây dựng.

Chi tiết của công tác thiết kế tại giai đoạn xây dựng được giải thích chi tiết trong bản lĩnh vực tri thức phần mềm nhưng chúng được áp dụng trong phạm vi nhỏ hơn như thuật toán, cấu trúc dữ liệu, và giao diện.

3.2. Các ngôn ngữ xây dựng phần mềm:

Các ngôn ngữ xây dựng phần mềm bao gồm nhiều hình thức giao tiếp để giải quyết vấn đề mà con người có thể giữ vai trò cụ thể. Các ngôn ngữ xây dựng phần mềm và sự ứng dụng của chúng có thể ảnh hưởng đến chất lượng phần mềm, hiệu suất phần mềm, độ tin cậy, dễ sử dụng, v.v... Chúng có thể là những tác nhân gây ra các lỗ hổng về bảo mật. Dạng ngôn ngữ xây dựng phần mềm đơn giản nhất là ngôn ngữ cấu hình trong đó các kỹ sư phần mềm chọn từ một tập hữu hạn những thành phần được định nghĩa trước để tạo ra phần mềm mới hay phần mềm cải biên theo cách cài đặt của người dùng. Các tệp cấu hình dạng văn bản trong hai HĐH Window và Linux là minh chứng của ngôn ngữ cấu hình. Các ngôn ngữ công cụ được xây dựng để xây dựng các ứng dụng dựa vào các công cụ trong bộ công cụ. Chúng phức tạp hơn các ngôn ngữ cấu hình.

Các ngôn ngữ công cụ có lẽ được định nghĩa tương minh giống như các ngôn ngữ lập trình ứng dụng, hay các ứng dụng được xây dựng bởi tập các giao diện của bộ công cụ.

Các ngôn ngữ viết kịch bản: là những ngôn ngữ được sử dụng chung cho các ứng dụng lập trình ứng. Trong một vài ngôn ngữ viết kịch bản bao gồm những kịch bản được gọi là tập các tệp hay các macro.

Các ngôn ngữ phần mềm là những loại ngôn ngữ xây dựng phần mềm linh hoạt nhất. Chúng cũng bao hàm thông tin tối thiểu về một ứng dụng cụ thể và các quy trình phát triển phần mềm do vậy chúng yêu cầu phải trải qua đào tạo và có kỹ năng để sử dụng hiệu quả. Việc chọn lựa ngôn ngữ lập trình có thể ảnh hưởng lớn đến khả năng gây lỗ hổng được giới thiệu trong phần viết mã – ví dụ, cách sử dụng dễ dãi của C và C++ là câu hỏi về cách chọn lựa theo chiều nhìn về an ninh bảo mật. Có các loại chung về ngôn ngữ ký hiệu sử dụng trong ngôn ngữ lập trình phổ biến như:

- Linguistic (ví dụ: C/C++, Java).
- Formal (ví dụ: Event-B).
- Visual (ví dụ: MatLab).

Các ký pháp ngôn ngữ (Linguistic notations) được mong đợi cụ trong việc dùng các chuỗi dạng văn bản trình diễn các công cuộc xây dựng phần mềm phức tạp. Việc biến các chuỗi văn bản thành các mẫu có cấu trúc, các chuỗi văn bản đưa ra các gợi ý tương minh làm cho người đọc chương trình, lập trình dễ dàng mường tượng trong cái gì sẽ xảy ra khi công việc xây dựng phần mềm được thực thi.

Các ký pháp chuẩn (Formal notations) ít khi dựa vào tiềm thức, các định nghĩa phải rõ ràng, rành mạch chính xác không được khó hiểu. Các ký pháp xây dựng chuẩn và các phương pháp chuẩn đều được xây dựng dựa trên nền hầu hết trên các mẫu phổ biến của các ngôn ngữ ký pháp lập trình hệ thống (system programming notations), ở đó độ chính xác, thời gian hoạt động của hành vi và khả năng kiểm thử là quan trọng hơn việc làm sao ánh xạ

sang ngôn ngữ tự nhiên dễ dàng. Các công đoạn xây dựng chuẩn mực đã định nghĩa nghĩa một cách chính xác dùng các biểu tượng, ký hiệu để tránh hiểu lầm giống như sử dụng các ngôn ngữ tự nhiên.

Các ký pháp trực quan hiếm khi dựa vào các ký pháp dùng ngôn ngữ văn bản và công tác xây dựng phô cập mà dùng cách diễn tả trực quan và đưa ra các thực thể, đối tượng trực quan trong việc diễn tả phần mềm. Xây dựng trực quan giải quyết vấn đề phức tạp chỉ bằng cách sử dụng các biểu tượng, ký hiệu và sắp xếp chúng với nhau theo ý đồ. Tuy nhiên, những biểu tượng này có là những công cụ đắc lực trong một vài trường hợp ở đó nhiệm vụ lập trình là đơn giản để xây dựng một phần mềm có giao diện trực quan đơn giản.

3.3. Viết mã chương trình

Viết mã chương trình phần mềm xem xét các kỹ thuật sau:

- Kỹ thuật viết mã nguồn có quy ước đặt tên cũng như giao diện sao cho dễ đọc, dễ hiểu.
- Quy ước đặt tên các biến, các kiểu liệt kê vô hướng, các biến, hằng các thực thể theo quy định.
- Viết mã có cấu trúc.*Chú ý đến mức độ an ninh khi viết mã cho chương trình phần mềm.
- Điều khiển lỗi cho cả trường hợp lỗi dữ liệu đầu vào và lỗi do phát sinh ngoại lệ.
- Sử dụng tài nguyên (sử dụng luồng ‘thread’ và khóa cơ sở dữ liệu)
- Tổ chức mà nguồn (các câu lệnh, các hàm, các lớp, các gói, hay các cấu trúc khác).
- Tài liệu viết mã chương trình.
- Làm đẹp mã chương trình. * 3.4. Kiểm thử tại giai đoạn xây dựng

Giai đoạn xây dựng bao gồm hai phần thường được thực hiện bởi kỹ sư phần mềm khi viết mã là:

- Kiểm thử từng phần
- Kiểm thử tích hợp

Mục đích của việc kiểm thử xây dựng làm giảm thời gian có lỗi khi viết mà và thời gian để phát hiện ra lỗi đó dẫn đến làm giảm chi phí cho việc sửa chúng. Trong một vài trường hợp, các kịch bản kiểm thử (test cases) được viết trong khi viết mã chương trình. Trường hợp khác, kịch bản kiểm thử được tạo ra trước khi viết mã chương trình.

Kiểm thử tại bước xây dựng bao gồm một tệp các kiểu kiểm thử, chúng được diễn tả chi tiết trong tài liệu kiểm thử phần mềm. Ví dụ: kiểm thử tại bước xây dựng không tham gia và quá trình kiểm thử hệ thống, kiểm thử alpha, kiểm thử beta, kiểm thử sức chịu đựng của hệ thống phần mềm (stress test), kiểm thử cấu hình, kiểm thử tính khả dụng hay các loại kiểm thử đặc thù khác.

Hai tiêu chuẩn mực được đưa ra trong tiêu đề kiểm thử tại giai đoạn xây dựng: IEEE Standard 829-1998, IEEE standard for Software Test Document, và IEEE Standard 1008-1987, IEEE Standard for Software Unit Testing.

3.5. Xây dựng để sử dụng lại

Việc xây dựng cho việc sử dụng lại mong muốn tạo ra thư viện, thành phần được áp dụng cho các dự án phần mềm khác ở hiện tại, tương lai. Việc xây dựng này dựa vào khả năng phân tích và thiết kế nhằm mục đích tránh sao chép cùng một mã chương trình tại nhiều nơi. Do vậy, nó đòi hỏi phải đóng gói (Encapsulation) thành các cấu trúc thư viện tốt hay các thành phần (component).

Các công việc liên quan đến xây dựng phần mềm cho sử dụng lại trong khi viết mã chương trình và kiểm thử như sau:

- Việc viết ứng dụng sao cho dễ dàng thay đổi với cơ chế tham số hóa, biên dịch có điều kiện, các mẫu thiết kế, và v.v...
- Việc bao gói ứng dụng mềm dẻo so cho phần mềm dễ dàng cấu hình và tùy biến.
- Diễn tả và công bố về các tính chất, phương thức của phần mềm.

3.6. Xây dựng với việc sử dụng lại

Xây dựng với việc sử dụng lại có nghĩa là tạo phần mềm mới có sử dụng lại các phần mềm đã có. Phương thức sử dụng phổ biến như là sử dụng lại mã nguồn của các chương trình, thư viện đã có sẵn được cung cấp bởi các thư viện, hệ điều hành, các công cụ đang được sử dụng,....

Bên cạnh đó, các ứng dụng được phát triển rộng rãi ngày nay thường sử dụng các thư viện mã nguồn mở. Phần mềm sử dụng lại phải đáp ứng yêu cầu về chất lượng trong việc phát triển phần mềm (ví dụ: bản quyền, an ninh, độ ổn định, dễ dàng thích nghi với các thư viện khác, dễ dàng mở rộng, nâng phiên bản...)

Các công việc liên quan đến xây dựng phần mềm có sử dụng lại khi viết mã và kiểm thử chương trình như sau:

- Chọn lựa các thư viện có thể sử dụng lại, cơ sở dữ liệu, các thủ tục kiểm thử, hay dữ liệu kiểm thử.
- Đánh giá khả năng sử dụng lại của phần mềm thông qua viết mã, và kiểm thử.
- Tích hợp các tính chất của phần mềm có sử dụng lại vào phần mềm hiện tại.
- Lập báo cáo thông tin về các phần mềm sử dụng lại bằng việc viết mã, kiểm thử và dữ liệu kiểm thử.

3.7. Chất lượng xây dựng

Bên cạnh các kết quả lỗi từ tài liệu yêu cầu, thiết kế được đưa ra trong giai đoạn xây dựng có thể gây ra các vấn đề về chất lượng nghiêm trọng ví dụ: lỗi hỏng về an ninh. Điều này không có nghĩa là chỉ có lỗi về an ninh mà có thể có lỗi ở nhiều phần khác nữa.

Các kỹ thuật để đảm bảo chất lượng việc viết mã chương trình gồm:

- Kiểm thử từng phần và kiểm thử tích hợp
- Xây dựng kiểm thử trước
- Sử dụng lập trình phòng chống (defensive programming) và khẳng định (assertions).
- Gõ rối (debugging).
- Điều tra (inspections).
- Kỹ thuật xem xét lại (review).
- Phân tích tĩnh (static analysis).

Các kỹ thuật chi tiết được chọn lựa phụ thuộc vào bản chất của phần mềm được xây dựng cũng như kỹ năng của các kỹ sư phần mềm thể hiện ra sao trong hoạt động xây dựng phần mềm. Các nhà lập trình phần mềm nên có kỹ năng tốt và cả những hạn chế chung ví dụ: trưởng dự án thường biết được điểm mạnh điểm yếu của từng thành viên trong đội dự án thông qua việc theo dõi quá trình làm việc. Việc phân tích mã nguồn tự động cho một vài ngôn ngữ lập trình thường có sẵn có thể được sử dụng để phân tích các vấn đề trong các dự án phần mềm.

Các hoạt động đánh giá chất lượng xây dựng phần mềm được phân biệt từ các hoạt động đánh giá chất lượng khác thông qua việc tập trung vào mã nguồn, các tài liệu liên quan đến mã nguồn như: bản thiết kế chi tiết và các tài liệu ít được liên hệ với việc viết mã nguồn như tài liệu yêu cầu, thiết kế ở mức cao, và các bản kế hoạch.

3.8. Tích hợp (Integration)

Hoạt động chính trong khi xây dựng phần mềm là tích hợp nhiều hàm, nhiều lớp, thành phần và các hệ thống con thành một hệ thống lớn. Ngoài ra, hệ thống phần mềm cụ thể cần được tích hợp với các phần mềm khác hay các hệ thống phần cứng.

Các suy xét liên quan đến việc xây dựng tích hợp bao gồm lập kế hoạch tích hợp các thành phần nào, các phần cứng nào để hỗ trợ lên bộ khung trong việc xây dựng phát triển và đưa ra các phiên bản phần mềm, các kế hoạch, mức độ kiểm thử và chất lượng của các thành phần trước khi đưa vào sử dụng cũng cột mốc kiểm thử của phiên bản phần mềm.

Các chương trình được kiểm thử theo từng dan giai đoạn hay theo hướng tiếp cận tăng dần. Tích hợp theo từng giai đoạn hay còn được gọi là tích hợp “big bang”, hay tích hợp từng phần đến khi tích hợp tất cả các phần khi các phần đều được hoàn thiện. Tích hợp tăng dần có nhiều điểm ưu việt hơn kiểu tích hợp truyền thống như: dễ dàng xác định được phần lỗi, theo dõi quá trình xây dựng, bàn giao sản phẩm sớm hơn, quan hệ với khách hàng chặt chẽ hơn. Áp dụng phương pháp tích hợp tăng dần, các người lập trình viết và kiểm thử

chương trình dưới hình thức các phần nhỏ sau đó kết hợp chúng thành các phần lớn hơn. Kiến trúc kiểm thử bổ sung, như stubs, drivers, và đối tượng mock thường xuyên sử dụng phương thức tích hợp tăng dần. Bên cạnh việc xây dựng và tích hợp các thành phần tại một thời gian nhất định cũng nhận định được các phản hồi sớm cho người lập trình và khách hàng. Ưu điểm khác của tích hợp tăng dần bao gồm việc dễ dàng phát hiện ra vị trí lỗi, theo dõi tiến trình phát triển của phần mềm, v.v...

4. Những công nghệ xây dựng:

4.1. Thiết kế và sử dụng API.

API là một tập hợp các chữ ký đã được biên dịch và có sẵn trong các thư viện hoặc framework phục vụ việc xây dựng và phát triển phần mềm. Bên cạnh các chữ ký, một API bao gồm các câu lệnh tác động đến hành vi của phần mềm.

Thiết API sao cho dễ đọc, dễ nhớ, dễ học, dễ mở rộng, và khả năng tương thích với phiên bản cũ cao. Các API thường xuyên tồn tại lâu hơn việc lập trình sinh ra nó và có sẵn trong các thư viện hoặc framework. Các API tiếp tục phát triển và

đảm bảo hỗ trợ việc phát triển và duy trì các ứng dụng phần mềm. Sử dụng API tham gia vào trong quá trình học, phát triển phần mềm, kiểm thử, tích hợp do vậy thiết kế sao cho có thể mở rộng cho framework (khung làm việc).

4.2. Các vấn đề thời gian chạy hướng đối tượng

Các ngôn ngữ hướng đối tượng hỗ trợ một dãy cơ chế thời gian chạy (runtime mechanisms) bao gồm đa hình (polymorphism) và phản xạ (reflection). Những cơ chế thời gian chạy này làm tăng tính mềm dẻo và sự linh hoạt của các chương trình hướng đối tượng.

Đa hình (Polymorphism) là những ngôn ngữ lập trình có khả năng hỗ trợ nhiều thao tác hoạt động chung chung mà không biết cho đến khi chạy chương trình những đối tượng nào được sinh ra và được gộp vào trong phần mềm.

Phản xạ (Reflection): một chương trình có khả năng quan sát, thay đổi cấu trúc và hành vi của bản thân trong lúc chạy. Ngoài ra nó còn quan sát các lớp, các trường, các interfaces, các phương thức mà không biết tên của chúng lúc biên dịch. Thêm vào đó, tại thời gian chạy, nó sinh ra thêm các đối tượng mới cùng với các hàm phương thức mới có sử dụng các lớp tham số và tên phương thức.

4.3. Tham số hóa và Generic

Các kiểu tham số hóa cũng được coi như là generics (Java, C#) và các mảng (C++) cho phép định nghĩa các kiểu hay các lớp không cần khai báo chính xác các biến hay các thuộc tính của nó. Các kiểu không xác định thường được sử dụng như các tham số trong thời điểm sử dụng.

4.4. Assertion, Design by contract và Defensive programming

Assertion là một chương trình chạy độc lập được nhúng trong chương trình chính và nó như là một chương trình con hoặc là một macro. Assertion được sử dụng cho các chương trình đòi hỏi có độ tin cậy, chính xác cao. Nó giúp cho người lập trình loại bỏ được các lỗi phát sinh trong khi lập trình và được biên dịch cùng với chương trình chính.

Design by contract là một cách tiếp cận phát triển chương trình dựa trên việc lập ra các tiền điều kiện và hậu điều kiện cho mỗi chương. Khi các tiền điều kiện và hậu điều kiện được sử dụng, mỗi hàm nhỏ hay lớp xây dựng một hợp đồng với phần còn lại của chương trình. Ngoài ra, hợp đồng cung cấp các tiêu chuẩn chính xác về hàm và hỗ trợ việc xác định hành vi của hàm đó. Design by contract được sử dụng trong việc cải tiến chất lượng xây dựng phần mềm.

Defensive programming nghĩa là nguyên lý cơ bản trong lập trình bảo vệ một chương trình hay một hàm khỏi bị hỏng do dữ liệu đầu vào sai. Các cách chung để điều khiển tham số đầu vào sai bao gồm kiểm tra giá trị của tất cả tham số đầu vào và quyết định điều khiển đầu vào sai như thế nào. Các Assertions thường xuyên được sử dụng trong lập trình phòng chống phục vụ việc kiểm tra giá trị đầu vào.

4.5. Cơ chế điều khiển lỗi, điều khiển ngoại lệ và tự khắc phục lỗi.

Error handling là một kỹ thuật cơ bản trong lập trình trong việc kiểm soát lỗi của chương trình phần mềm đảm bảo cho chương trình hoạt động ổn định, thông báo lỗi chính xác.

Exception handling là cơ chế phát hiện và xử lý lỗi hay những sự kiện ngoại lệ.

Fault tolerant là kỹ thuật phát hiện lỗi và xử lý lỗi làm tăng độ tin cậy của chương trình phần mềm

4.6. Các mô hình chạy

Các mô hình chạy trùu tường chi tiết các ngôn ngữ lập trình cụ thể và các quyết định về cách tổ chức phần mềm. Sự khác biệt từ các mô hình phần mềm truyền thống, đặc tính xây dựng trong ngôn ngữ mô hình chạy như xUML (executable UML) có thể được triển khai trong nhiều môi trường phần mềm không có sự thay đổi.

Trình biên dịch mô hình chạy có thể biên một mô hình chạy thành việc sử dụng một tập các quyết định về việc chọn môi trường phần cứng và phần mềm đích. Do đó, việc xây dựng các mô hình chạy có thể được xem như việc xây dựng một phần mềm chạy.

Các mô hình chạy là một nền tảng hỗ trợ kiến trúc Model-Driven Architecture (MDA) của tổ chức Object Management Group (OMG). Dựa vào mô hình chạy là một hình thức chọn lựa mô hình độc lập hệ điều hành (Platform Independent Model). Mô hình PIM cũng là một giải

pháp để giải quyết vấn đề không phụ thuộc vào bất kỳ công nghệ cài đặt nào. Do vậy mô hình cụ thể (Platform Specific Model) bao gồm thông tin cài đặt chi tiết có được bằng việc lắp ráp lại với nhau.

4.7. Kỹ thuật xây dựng State-Based và Table-Driven

Ngôn ngữ lập trình State-Based hay Automata-Based là một công nghệ lập trình sử dụng việc tính toán chuyển đổi trạng thái hữu hạn để diễn tả hành vi của phần mềm. Biểu đồ chuyển đổi trạng thái được diễn ra trong suốt quá trình thực hiện của phần mềm.

Phương thức Table-Driven là một lược đồ lưu trữ, tìm kiếm thông tin chứ không sử dụng các câu lệnh logic như if và case. Lập trình table-driven đơn giản hơn rất nhiều lập trình logic phức tạp và rất dễ sửa đổi. Khi lập trình thì hình thức này người lập trình thường chú ý đến 2 vấn đề lưu trữ thông tin gì vào bảng hay các bảng và cách truy cập thông tin trong bảng như thế nào hiệu quả nhất?

4.8. Cấu hình thời gian chạy và quốc tế hóa

Để tăng thêm sự mềm dẻo, mỗi chương trình thường được xây dựng hỗ trợ cơ chế bắt đồng bộ. Kỹ thuật này gắn các giá trị vào các biến trong phần cấu hình của chương trình trong khi đang chạy. Tham số cấu hình thường xuyên được cập nhật và đọc cấu file cấu hình trong chế độ just-in-time.

Quốc tế hóa: là kỹ thuật xây dựng phần mềm có thể sử dụng và hoạt động được nhiều vùng miền có nền văn hóa khác nhau.

4.9. Xử lý đầu vào theo kỹ thuật grammar-based

Việc xử lý đầu vào theo kỹ thuật grammar-based thực chất phân tích dữ liệu đầu vào theo cú pháp nhất định.

4.10. Những cơ chế đồng thời (Concurrence primitives) Cơ chế đồng thời là sự trừu tượng hóa trong lập trình được cung cấp bởi ngôn ngữ lập trình hay hệ điều hành hỗ trợ việc đồng thời hay đồng bộ. Các cơ chế đồng thời nổi tiếng bao gồm semaphores, monitors và mutexes.

Semaphore là một biến kiểu protected và kiểu dữ liệu trừu tượng tạo ra sự trừu tượng đơn giản nhưng hữu ích trong việc điều khiển sự truy cập tới nguồn tài nguyên chung của nhiều chương trình, tiến trình đồng thời trong môi trường lập trình.

Monitor là một kiểu dữ liệu trừu tượng thể hiện một tập các thao tác được người dùng định nghĩa được chạy với hình thức loại trừ lẫn nhau. Một monitor có định nghĩa các biến, hàm chia sẻ nhưng chỉ cho một tiến trình truy cập tại một thời điểm.

Mutex là một cơ chế đồng bộ giản đơn cho phép một tiến trình truy cập vào tài nguyên dùng chung tại một thời điểm.

4.11. Tầng trung gian (Middleware).

Middleware được xem như một cầu nối giúp các ứng dụng phần mềm có thể làm việc được với nhau.

Middleware là một cơ chế được áp dụng rộng rãi trong hệ thống phần mềm lớn. Nó cung cấp một dịch vụ nằm ở tầng trên của hệ điều hành và nằm dưới tầng ứng dụng.

Middleware cung cấp các thành phần phần mềm đảm bảo việc truyền nhận dữ liệu, thực thi các câu lệnh nếu có yêu cầu từ ứng dụng một cách ổn định, trong suốt với người dùng qua mạng.

4.12. Phương thức xây dựng phần mềm cho hệ thống phân tán

Một hệ thống phân tán bao gồm các máy tính vật lý nằm rải rác khắp nơi được kết nối đến nhau thông qua việc kết nối mạng. Cơ chế này cho phép các người dùng có thể truy cập đến tài nguyên dùng chung.

Việc xây dựng hệ thống phân tán sẽ khác biệt với việc xây dựng hệ thống phần mềm truyền thống như việc giải quyết: song song (parallel), trao đổi và nói chuyện với nhau (communication), và khả năng xử lý và tự khắc phục lỗi (fault tolerant).

Các kiến trúc của việc xây dựng hệ thống phần mềm phân tán thường: kiến trúc 3-tier, kiến trúc n-tier, kiến trúc client-server, v.v...

4.13. Xây dựng hệ thống heterogeneous

Một hệ thống heterogeneous bao gồm những chương trình tính toán nhỏ chuyên biệt. Các chương trình tính toán chuyên biệt này được điều khiển hoạt động độc lập và có giao tiếp với nhau ví dụ như: Digital Signal Processors (DSP), micro controllers và peripheral processors. Các phần tính toán được điều khiển độc lập và giao tiếp với một thành phần khác. Các hệ thống nhúng là những hệ thống heterogeneous. Thiết kế hệ thống heterogeneous yêu cầu kết hợp nhiều ngôn ngữ cụ thể để thiết kế nhiều phần của hệ thống. Những vấn đề chính bao gồm việc kiểm tra phần mềm đa ngôn ngữ, mô phỏng, và giao tiếp. Trong khi phát triển phần mềm, mã nguồn, phần cứng được thực hiện trước đồng thời thông qua trải qua từng bước. Phần cứng thường xuyên được mô phỏng trong một dãy trong hệ thống (FPGAs) hay ASICs. Phần mềm được chuyển thành ngôn ngữ lập trình bậc thấp.

4.14. Phân tích và cải tiến sự hoạt động

Việc phát triển một phần mềm hiệu quả phụ thuộc vào quyết định chọn lựa thiết kế kiến trúc chi tiết, xây dựng cấu trúc dữ liệu, chọn lựa thuật toán ảnh hưởng đến tốc độ xử lý của chương trình.

Phân tích sự hoạt động là công việc điều tra hành vi sử dụng trong một khoảng thời gian nhất định nhằm thu thập thông tin, đánh giá tìm ra các điểm nóng của chương trình phần mềm cần cải thiện.

Cải tiến và làm đẹp mã nguồn là một cách để làm tăng tốc độ xử lý của chương trình cũng như làm cho mã nguồn dễ đọc, dễ bảo trì.

4.15. Những nền tảng chuẩn mực (Platform Standards)

Những nền tảng chuẩn giúp cho người lập trình xây dựng các phần mềm được linh hoạt có thể chạy được trong các môi trường tương thích mà không có sự thay đổi.

Các nền tảng chuẩn giúp cho người lập trình phát triển phần mềm trên thế giới hiện nay như: J2EE cho Java, .NetFramework cho Visual Studio hoặc các framework cho web như: Boostraps, Angular,... Struts cho Java, MVC của Microsoft.

4.16. Viết chương trình kiểm thử trước

Trước khi bắt tay lập trình, người lập trình xây dựng chương trình kiểm thử trước theo kịch bản.

Việc viết chương trình kiểm thử trước giúp cho người lập trình phát hiện được lỗi sớm và sửa lỗi dễ hơn theo lối tư duy lập trình cũ. Ngoài ra nó bắt người lập trình phải tìm hiểu về yêu cầu nghiệp vụ, thiết kế, kiến trúc của chương trình trước khi việc mã nguồn giúp cho việc phát hiện ra vấn đề.

5. Các công cụ xây dựng phần mềm

5.1. môi trường phát triển phần mềm

Môi trường phát triển phần mềm: cung cấp cho người lập trình đầy đủ các tính năng cần thiết trong việc phát triển phần mềm từ việc tạo mới project, màn hình lập trình, màn hình debug, màn hình unit test,v.v...

Ngoài ra nó còn cung cấp các thêm các chức năng biên dịch, chạy thử, cùng với các công cụ kết nối đến chương trình quản lý mã nguồn (Github, TFS,...).

Việc chọn lựa môi trường phát triển phần mềm ảnh hưởng đến mục đích xây dựng phần mềm và chất lượng của phần mềm.

5.2. Công cụ xây dựng giao diện

Công cụ xây dựng giao diện (GUI Builders) giúp cho người phát triển phần mềm trong việc xây dựng màn hình giao diện giữa người dùng và hệ thống nhanh hơn theo đúng tiêu chuẩn WYSWYG.

Các công cụ xây dựng giao diện cung cấp người dùng có cái nhìn trực quan trong việc xây dựng giao diện ví dụ: trong VS 2015, người lập trình chỉ việc kéo thả các icon trong bảng tool để hình thành lên giao diện tương tác người dùng và hệ thống.

5.3. Các công cụ kiểm thử từng phần (Unit Testing Tools)

Kiểm thử từng phần kiểm tra chức năng hoạt động của từng module phần mềm một cách độc lập từ các phần tử phần mềm khác một cách riêng rẽ. Kiểm tra từng phần thường được làm tự động. Các người phát triển có thể các công cụ kiểm thử từng phần và các framework, người lập trình có thể viết mã theo chuẩn để kiểm thử xác thực sự đúng đắn từng phần bằng việc cung cấp tập các dữ liệu đầu vào. Việc kiểm thử riêng lẻ giống như một đối tượng, và người thực hiện kiểm thử chạy tất cả các trường hợp nếu có trường hợp nào bị lỗi sẽ đánh dấu và thông báo cho người kiểm thử biết.

5.4. Các công cụ Performance analysis, Profiling and Slicing

Việc phân tích hoạt động của phần mềm dựa được sử dụng để quyết định có cải tiến mã nguồn hay không. Để phân tích hoạt động của phần mềm thường sử dụng công cụ Profiling.

Công cụ profiling sẽ theo dõi mã nguồn khi chạy và ghi lại bao nhiêu mỗi câu lệnh chạy bao nhiêu lần và cần bao nhiêu thời gian để chạy câu lệnh đó. Nó giúp cho người lập trình nhìn sâu vào trong chương trình khi nó chạy và biết được chương trình hoạt động như thế nào và biết được chỗ nào cần phải cải tiến để tăng sự hoạt động của chương trình.

Công cụ slicing sẽ tính toán các câu lệnh theo một thuật toán chuyên biệt để đánh giá sự tối ưu của từng dòng lệnh cũng như phát hiện ra lỗi logic của chương trình cần phải cải tiến.

Chapter 4: Software Testing

Nhóm 4:

Phạm Văn Thiện

Ngô Thế Hải Anh

Nguyễn Anh Tuấn

Giới thiệu

Kiểm thử phần mềm bao gồm việc kiểm chứng động (dynamic), đó là một chương trình cung cấp các hành vi dự kiến trên một tập hữu hạn các trường hợp thử nghiệm, phù hợp được lựa chọn từ các miền thực hiện thường là vô hạn.

Các vấn đề quan trọng trong việc mô tả các kiến thức kiểm thử phần mềm (knowledge area - KA) dưới đây:

- **Dynamic**

Thuật ngữ này có nghĩa là kiểm thử luôn luôn bao gồm thực thi chương trình khi lựa chọn đầu vào (input). Để được chính xác, giá trị đầu vào đơn (alone) không phải lúc nào cũng đủ để xác định 1 bài test, vì một hệ thống không xác định phức tạp có thể phản ứng với các đầu vào cùng với các hành vi khác nhau, tùy thuộc vào trạng thái hệ thống. Tuy nhiên trong KA này, thuật ngữ đầu vào sẽ phải được duy trì, với quy ước bao hàm rằng ý nghĩa của nó cũng bao gồm một trạng thái đầu vào quy định trong những trường hợp quan trọng. Các kỹ thuật tĩnh khác nhau và hỗ trợ cho kiểm thử động. Các kỹ thuật tĩnh được bao gồm trong Software Quality KA. Điều đáng chú ý là thuật ngữ không thống nhất giữa các cộng đồng khác nhau và một số sử dụng thuật ngữ “kiểm thử” trong mối liên hệ với các kỹ thuật tĩnh.

- **Kiểm thử tĩnh:** Kiểm thử tĩnh là một hình thức của kiểm thử phần mềm mà phần mềm không được sử dụng thực sự. Điều này ngược với thử nghiệm động. Thường thì nó không kiểm thử chi tiết mà chủ yếu kiểm tra tính đúng đắn của code (mã lệnh), thuật toán hay tài liệu. Chủ yếu là kiểm tra cú pháp của code và/hoặc review code (là kiểm tra xem code có được viết theo đúng tiêu chuẩn code – ví dụ cách đặt tên hàm tên biến, cách sử dụng hàm chung,... đã đưa ra hay chưa) hoặc tài liệu để tìm lỗi bằng cách thủ công. Đây là loại kiểm thử có thể được sử dụng bởi DEV (những người lập trình), làm việc một cách độc lập. Các kỹ thuật review code, kiểm tra và walkthroughs cũng được sử dụng trong kiểm thử tĩnh này. Từ góc nhìn theo quan điểm của kiểm thử hộp đen,

Kiểm thử tĩnh liên quan đến việc xem xét các yêu cầu và các tài liệu thiết kế chi tiết (ví dụ Spec, SRS, BSS). Điều này được thực hiện với một tầm nhìn hướng tới đầy đủ hay phù hợp cho các nhiệm vụ. Lỗi được phát hiện ở giai đoạn phát triển này là ít tốn kém để sửa chữa hơn so với bug phát hiện được ở các giai đoạn sau này trong các quy trình phát triển phần mềm.

- **Finite:**

Ngay cả trong các chương trình đơn giản, rất nhiều trường hợp kiểm thử về mặt lý thuyết đòi hỏi kiểm tra toàn diện có thể yêu cầu nhiều tháng hoặc nhiều năm để thực hiện. Đó là lý do tại sao trong thực tế, một tập đầy đủ các kiểm thử có thể được coi là vô hạn, và sự kiểm thử được tiến hành trên một tập tất cả các bài kiểm thử khả thi, có thể được xác định theo các tiêu chí rủi ro và ưu tiên. Kiểm thử luôn bao hàm sự cân bằng giữa các tài nguyên hạn chế và tiến độ.

- **Selected:**

Có nhiều kỹ thuật kiểm thử khác nhau được đề xuất trong một tập kiểm thử được chọn, và các kỹ sư phần mềm phải được nhận thức rằng các tiêu chí lựa chọn khác nhau có thể mang lại sự khác nhau về tính hiệu quả. Nhưng làm thế nào để xác định các tiêu chí lựa chọn phù hợp nhất trong điều kiện nhất định là 1 vấn đề phức tạp.

- **Expected:**

Phải có khả năng, mặc dù không phải lúc nào cũng dễ dàng, để quyết định xem các kết quả quan sát của chương trình thử nghiệm được chấp nhận hay không; nếu không, các nỗ lực thử nghiệm là vô ích. Các hành vi quan sát có thể được so sánh với nhu cầu của người sử dụng (thường được gọi là thử nghiệm để xác nhận), chống lại một đặc điểm kỹ thuật (thử nghiệm để xác minh), hoặc, có lẽ, đối với các hành vi được mong đợi từ các yêu cầu bao hàm hoặc mong đợi (xem thử nghiệm Chấp nhận trong các yêu cầu phần mềm KA).

Trong những năm gần đây, kiểm thử không còn được coi là một hoạt động mà chỉ bắt đầu sau khi giai đoạn coding được hoàn thành với mục đích hạn chế của việc phát hiện lỗi. Kiểm thử phần mềm trở nên phổ biến trong suốt quá trình phát triển và bảo trì. Kế hoạch kiểm thử phần mềm nên bắt đầu với giai đoạn đầu của quy trình yêu cầu phần mềm, các kế hoạch và quy trình thử nghiệm phải được phát triển một cách có hệ thống và liên tục - và có thể tinh chế - như tiến hành phát triển phần mềm. Những hoạt động lập kế hoạch kiểm thử và thử nghiệm thiết kế cung cấp các đầu vào hữu ích cho các nhà thiết kế phần mềm và giúp họ làm nổi bật những khuyết điểm, thiết sót.

Đối với nhiều tổ chức, phương pháp tiếp cận đến chất lượng phần mềm là một trong những sự ngăn chặn: tức là ngăn chặn vấn đề tốt hơn là sửa chữa chúng. Kiểm thử có thể được nhìn thấy, sau đó, được dùng như một phương tiện để cung cấp thông tin về các tính năng và chất lượng các thuộc tính của phần mềm và cũng để xác định lỗi trong những trường hợp ngăn chặn lỗi không có hiệu quả. Sự thật hiển nhiên là phần mềm có thể chứa lỗi, thậm

chỉ sau khi hoàn thành việc kiểm thử bao quát. Các lỗi phần mềm có sau đó sẽ được giải quyết bằng bảo trì sửa chữa. Mục bảo trì phần mềm có trong phần Software Maintenance KA (chương 5).

Trong Kỹ thuật quản lý chất lượng phần mềm, được phân loại thành 2 kỹ thuật đáng chú ý là Tĩnh (không thực thi mã) và động (thực thi mã). KA này tập trung và các kỹ thuật động. Kiểm thử phần mềm cũng có liên quan đến xây dựng phần mềm.

Chi tiết chủ đề kiểm thử phần mềm

1. Kiểm thử phần mềm cơ bản

1.1. Thuật ngữ liên quan đến kiểm thử.

1.1.1. Định nghĩa về kiểm thử và thuật ngữ liên quan

Kiểm thử phần mềm là một cuộc kiểm tra được tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của sản phẩm hoặc dịch vụ được kiểm thử. Kiểm thử có thể cung cấp cho doanh nghiệp một quan điểm, một cách nhìn độc lập về phần mềm để từ đó cho phép đánh giá và thấu hiểu được những rủi ro trong quá trình triển khai phần mềm.

Trong kỹ thuật kiểm thử không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm (bao gồm các lỗi và các thiếu sót) mà còn là một quá trình phê chuẩn và xác minh một chương trình máy tính / ứng dụng / sản phẩm nhằm:

- Đáp ứng được mọi yêu cầu hướng dẫn khi thiết kế và phát triển phần mềm.
- Thực hiện công việc đúng như kỳ vọng.
- Có thể triển khai được với những đặc tính tương tự.
- Và đáp ứng được mọi nhu cầu của các bên liên quan.

Tùy thuộc vào từng phương pháp, việc kiểm thử có thể được thực hiện bất cứ lúc nào trong quá trình phát triển phần mềm.

1.1.2. Lỗi và chịu lỗi

1.2. Key Issues (Các vấn đề chính)

1.2.1. Tiêu chí lựa chọn kiểm thử/Kiểm tra mức độ đầy đủ tiêu chí

Tiêu chí lựa chọn thử nghiệm có nghĩa là lựa chọn trường hợp kiểm thử hoặc xác định một tập các trường hợp kiểm thử là đủ cho một mục đích cụ thể.

1.2.2. Hiệu quả kiểm thử/Mục tiêu kiểm thử

Hiệu quả kiểm thử được xác định bằng cách phân tích một tập hợp các chương trình thực thi. Lựa chọn kiểm thử được thực hiện có thể được hướng dẫn bởi các mục tiêu khác nhau.

1.2.3. Kiểm thử phát hiện khiếm khuyết

Phát hiện lỗi hoặc những khiếm khuyết của phần mềm để thấy được ứng xử của nó có chính xác hoặc phù hợp với tài liệu đặc tả của nó hay không. Về mặt lý thuyết, chúng ta phải kiểm thử hệ thống một cách cẩn kẽ thì mới khẳng định được chương trình không còn khiếm khuyết. Tuy nhiên, trong thực tế không thể kiểm thử một cách cẩn kẽ được.

1.2.4. Các vấn đề về Oracle

Oracle là bất kỳ người hoặc máy móc dùng để quyết định xem liệu một chương trình có thực hiện một cách chính xác trong một thử nghiệm nhất định và phù hợp kết quả là đạt hoặc thất bại (các nguyên tắc hay cơ chế phát hiện vấn đề). Kiểm thử không thể xác định hoàn toàn được tất cả các lỗi bên trong phần mềm. Thay vào đó, nó so sánh trạng thái và hành vi của sản phẩm với các oracle. Các oracle này có thể bao gồm (nhưng không giới hạn ở) các đặc tả phần mềm, hợp đồng, sản phẩm tương đương, các phiên bản trước của cùng một sản phẩm, phù hợp với mục đích dự kiến nhằm đáp ứng sự kỳ vọng của người dùng, khách hàng, quy định của pháp luật hiện hành và các tiêu chuẩn liên quan khác.

1.2.5. Giới hạn lý thuyết và thực tiễn của kiểm thử

1.2.6. Vấn đề về đường dẫn không khả thi

Đường dẫn không khả thi là các đường dẫn luồng điều khiển không thể được thực thi bởi bất kỳ dữ liệu đầu vào.

1.2.7. Khả năng kiểm thử

Khả năng kiểm thử là mức độ mà một thành phần phần mềm (hệ thống phần mềm, module, tài liệu thiết kế) hỗ trợ kiểm thử trong một bối cảnh kiểm thử nhất định. Nếu khả năng kiểm thử của thành phần phần mềm cao, thì việc tìm kiếm lỗi trong hệ thống (nếu có) bằng việc kiểm thử được dễ dàng hơn.

1.3. Mối quan hệ của kiểm thử với các hoạt động khác

- Kiểm thử vs kỹ thuật quản lý chất lượng phần mềm tĩnh
- Kiểm thử vs Bằng chứng về tính đúng đắn và xác minh hình thức
- Kiểm thử vs Gỡ lỗi
- Kiểm thử vs Xây dựng chương trình

2. Test Levels

Kiểm thử phần mềm thường được thực hiện ở các cấp độ khác nhau trong suốt quá trình phát triển và bảo trì. Mức có thể được phân biệt dựa trên các đối tượng thử nghiệm, được gọi là các target, hoặc về mục đích, được gọi là các objective (tùy cấp thử nghiệm).

2.1. Mục tiêu của kiểm thử (Target of the Test)

Mục tiêu của thử nghiệm có thể khác nhau: một mô-đun duy nhất, một nhóm các mô-đun như (liên quan theo mục đích, sử dụng, hành vi, hoặc cơ cấu), hay toàn hệ thống. Ba giai đoạn thử nghiệm có thể được phân biệt: đơn vị, tích hợp và hệ thống.

2.1.1. Kiểm thử đơn vị

Unit testing đề cập đến các kiểm thử để chứng thực (xác minh - verify) chức năng của một phần riêng biệt của code, thường ở mức hàm (function level). Trong một môi trường hướng đối tượng (object-oriented environment), kiểm thử đơn vị thường được sử dụng ở mức lớp (class) và kiểm thử các đơn vị nhỏ nhất bao gồm các hàm constructor và destructor.

Loại kiểm thử này thường được viết bởi các DEV như công việc của họ trong việc code (loại test white-box), để bảo đảm rằng từng hàm riêng biệt hoạt động đúng theo mong muốn. Một hàm có thể có nhiều kiểm thử, để bắt được các trường hợp hoặc các nhánh trong code.

Unit testing một mình không thể bảo đảm chức năng của một bộ phận của phần mềm mà là sử dụng để bảo đảm rằng các khối kiến trúc của phần mềm làm việc độc lập với nhau. Unit testing (kiểm thử đơn vị) cũng được gọi là component testing (kiểm thử thành phần).

2.1.2. Kiểm thử tích hợp

Integration testing là một loại kiểm thử phần mềm mà tìm kiếm để kiểm tra các giao diện giữa các thành phần dựa vào thiết kế của phần mềm. Các thành phần phần mềm có thể được tích hợp lại với nhau theo cách lặp đi lặp lại (từng phần nhỏ ghép lại với nhau, rồi ghép tiếp phần nhỏ khác vào nữa, hành động này lặp lại cho đến khi kết hợp toàn bộ phần mềm) hoặc tất cả các thành phần cùng tích hợp một lần (gọi là “big bang”). Thông thường trước đây được xem là một cách làm tốt hơn từ khi nó cho phép các vấn đề về giao diện được xác định vị trí nhanh hơn và cố định. Integration testing làm việc để tìm ra lỗi (defect) trong các giao diện và giao tiếp giữa các thành phần (mô-đun). Các nhóm thành phần phần mềm đã được kiểm thử lớn dần từng bước tương ứng với các yếu tố của thiết kế kiến trúc đã được tích hợp và kiểm thử cho đến khi phần mềm hoạt động như một hệ thống.

2.1.3. Kiểm thử hệ thống

System testing kiểm thử một hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đáp ứng được yêu cầu. Kiểm thử tích hợp hệ thống chứng thực rằng hệ thống đã được tích hợp với các hệ thống bên ngoài hoặc hệ thống thứ ba đã được xác định trong các yêu cầu hệ thống.

2.2. Mục đích của kiểm thử

Kiểm thử được tiến hành trong bối cảnh mục tiêu cụ thể, trong đó được ghi nhận nhiều hơn hoặc ít hơn một cách rõ ràng và có độ chính xác khác nhau. Trong đó nêu rõ mục đích của kiểm thử một cách chính xác, hỗ trợ về mặt định lượng đo lường và kiểm soát trong quá

trình kiểm thử. Kiểm thử có thể được dùng để xác minh các tính chất khác nhau. Trường hợp kiểm thử có thể được thiết kế để kiểm tra các thông số kỹ thuật chức năng được thực hiện một cách chính xác, hiệu suất, độ tin cậy, khả dụng...

2.2.1. Nghiệm thu/Năng lực kiểm thử

Nghiệm thu/Năng lực kiểm thử xác định một hệ thống đáp ứng tiêu chuẩn nghiệm thu của nó, thường là bằng cách kiểm tra các hành vi hệ thống mong muốn đối với các yêu cầu của khách hàng.

2.2.2. Kiểm thử cài đặt

Thông thường, sau khi hoàn thành hệ thống và chấp nhận kiểm thử, các phần mềm được xác minh sau khi cài đặt trong môi trường mục tiêu. Cài đặt kiểm thử có thể được xem như là hệ thống thử nghiệm được tiến hành trong môi trường hoạt động của cấu hình phần cứng và hoạt động hạn chế khác.

2.2.3. Kiểm thử Alpha và Beta

Alpha testing là việc kiểm thử hoạt động chức năng thực tế hoặc giả lập do người dùng/khách hàng tiềm năng hoặc một nhóm test độc lập thực hiện tại nơi sản xuất phần mềm. Alpha testing thường dùng cho phần mềm đóng gói sẵn để bán (ví dụ như MS office, window, chương trình diệt virus) là một hình thức kiểm thử chấp nhận nội bộ, trước khi phần mềm được tiến hành kiểm thử beta.

Beta testing được thực hiện sau alpha testing. Các phiên bản của phần mềm - được biết như là các phiên bản beta – chúng được phát hành tới một số lượng giới hạn khán giả bên ngoài nhóm sản xuất phần mềm. Sản phẩm được phát hành đến một số nhóm người để test nhiều hơn nữa có thể chắc chắn rằng sản phẩm có một số bug. Thỉnh thoảng, các phiên bản beta được phát hành rộng rãi để tăng phạm vi phản hồi từ một lượng người sử dụng tương lai lớn nhất.

2.2.4. Độ tin cậy và đánh giá

Kiểm thử cải thiện độ tin cậy bằng cách xác định và sửa lỗi. Ngoài ra, các biện pháp thống kê về độ tin cậy có thể được bắt nguồn bằng cách tạo ra một cách ngẫu nhiên trường hợp kiểm tra theo hoạt động của phần mềm.

2.2.5. Kiểm thử hồi quy

Tập trung vào việc tìm kiếm lỗi sau khi xảy ra việc thay đổi code. Đặc biệt, nó tìm kiếm theo cách hồi qui (đệ qui) hoặc kiểm tra các bug cũ có bị lại hay không. Hồi qui như vậy xảy bất cứ khi nào mà chức năng phần mềm trước đây làm việc đúng đã ngưng làm việc theo mong đợi. Điển hình, hồi qui xảy ra như là một kết quả không mong muốn của việc thay đổi chương trình, khi phần code của phần mềm mới được phát triển xung đột với code cũ đang

có. Phương pháp thông thường của kiểm tra hồi quy là bao gồm việc chạy lại các kiểm thử trước đây và kiểm tra xem có lỗi đã được fixed trước đây bị lỗi lại (bị lại các lỗi cũ đã fixed rồi). Độ sâu của việc kiểm thử phụ thuộc vào các giai đoạn trong quá trình phát hành và rủi ro của các tính năng được thêm vào. Chúng có thể được hoàn thành – vì việc thay đổi đã thêm vào sau bản phát hành hoặc coi nó là mạo hiểm, rất hời hợt, bao gồm các kiểm thử thường hợp đúng (positive) trên từng chức năng – nếu các thay đổi được thêm vào trước khi phát hành hoặc coi nó ít rủi ro.

2.2.6. Kiểm thử hiệu năng

Kiểm thử hiệu năng được thực hiện để xác định hệ thống hoặc hệ thống con thực hiện một khối lượng công việc cụ thể nhanh thế nào. Nó cũng có thể dùng để xác nhận và xác minh những thuộc tính chất lượng khác của hệ thống như là khả năng mở rộng, độ tin cậy và sử dụng tài nguyên. Load testing là khái niệm chủ yếu của việc kiểm thử mà có thể tiếp tục hoạt động ở một mức tải cụ thể, cho dù đó là một lượng lớn dữ liệu hoặc lượng lớn người sử dụng. Cái này gọi chung là khả năng mở rộng của phần mềm. Hoạt động load testing liên quan khi thực hiện hoạt động phi chức năng thường được gọi là kiểm thử sức chịu đựng (độ bền).

2.2.7. Kiểm thử bảo mật

Kiểm thử bảo mật tập trung vào việc xác minh rằng phần mềm được bảo vệ khỏi các cuộc tấn công từ bên ngoài. Đặc biệt, kiểm thử bảo mật xác minh tính bảo mật, tính toàn vẹn và tính sẵn sàng của hệ thống và dữ liệu của nó.

2.2.8. Kiểm thử Stress

Stress testing là một hình thức kiểm thử được sử dụng để xác định tính ổn định của một hệ thống hoặc một thực thể được đưa ra. Nó liên quan đến những kiểm thử vượt quá khả năng bình thường của hệ thống, thường để xác định điểm phá vỡ, để quan sát các kết quả.

Stress testing có thể có nghĩa cụ thể hơn trong một số ngành công nghiệp nhất định, như là kiểm thử tính mồi của vật liệu.

2.2.9. Kiểm thử Back-to-Back (so sánh)

Khi triển khai nhiều phiên bản phần mềm từ cùng một đặc tả. Kiểm thử hộp đen cho các sản phẩm này được thực hiện với cùng ca kiểm thử và cùng các dữ liệu vào. So sánh các kết quả thu được, nếu có khác biệt nghĩa là có sai trong một sản phẩm nào đó.

2.2.10. Kiểm thử phục hồi

Kiểm thử phục hồi nhằm mục đích kiểm tra khả năng khởi động lại của phần mềm sau khi hệ thống bị treo hoặc crash.

2.2.11. Kiểm thử giao diện

Các nhược điểm giao diện rất phổ biến trong các hệ thống phức tạp. Kiểm thử giao diện nhằm xác minh các thành phần giao diện chính xác để cung cấp những trao đổi chính xác của dữ liệu và kiểm soát thông tin. Thông thường các trường hợp thử nghiệm được tạo ra từ các đặc tả giao diện. Mục tiêu cụ thể của thử nghiệm giao diện là để mô phỏng việc sử dụng các API của các ứng dụng của người dùng cuối. Điều này liên quan đến việc tạo ra các thông số của các cuộc gọi API, thiết lập các điều kiện môi trường bên ngoài, và định nghĩa của dữ liệu nội bộ có ảnh hưởng đến các API.

2.2.12. Kiểm thử cấu hình

Trong trường hợp phần mềm được xây dựng để phục vụ người dùng khác nhau, kiểm thử cấu hình dùng để kiểm chứng các phần mềm theo quy định cấu hình khác nhau.

2.2.13. Kiểm thử tính khả dụng và tương tác người - máy

Nhiệm vụ chính của phần này là đánh giá như thế nào là dễ dàng cho người dùng cuối để tìm hiểu và sử dụng phần mềm. Nói chung, nó có thể liên quan đến việc thử nghiệm các chức năng phần mềm hỗ trợ việc sử dụng, tài liệu hướng dẫn nhằm hỗ trợ người sử dụng, và khả năng của hệ thống để phục hồi từ lỗi người sử dụng

3. Kỹ thuật kiểm thử

Một trong những trọng tâm của việc kiểm thử đó là phát hiện nhiều lỗi nhất có thể. Nhiều kỹ thuật đã được phát triển để làm việc này. Những kỹ thuật này để thử phá một chương trình trình một cách hệ thống bằng việc chỉ ra đầu và mà sẽ tạo ra các trạng thái của chương trình. Ví dụ, xem xét các lõi con của miền đầu vào, các kịch bản, trạng thái và luồng dữ liệu.

Việc phân loại các kỹ thuật kiểm thử được trình bày ở đây là dựa trên cách kiểm thử được thực hiện: từ trực giác và kinh nghiệm của kỹ sư phần mềm, các chi tiết kỹ thuật, cấu trúc mã nguồn, lỗi thực hoặc lỗi tưởng tượng được tìm ra, việc sử dụng được dự đoán trước, mô hình hoặc bản chất của ứng dụng. Một loại tiến hành với 2 hoặc nhiều kỹ thuật kết hợp lại.

Một vài kỹ thuật này được xếp loại như white-box hoặc glass-box nếu kiểm thử dựa trên thông tin về cách phần mềm được thiết kế hoặc viết mã nguồn. Hoặc được xem như là black-box nếu nó chỉ dựa trên trạng thái đầu vào đầu ra của phần mềm. Dưới đây bao gồm vài kỹ thuật thông dụng, nhưng một vài người kiểm thử phụ thuộc vào một số kỹ thuật hơn các kỹ thuật khác

3.1. Dựa trên trực giác và kinh nghiệm của kỹ sư phần mềm

3.1.1. Tùy biến

Có lẽ kỹ thuật được thực hiện nhiều nhất là kiểm thử tùy biến: các kiểm thử có được dựa trên kỹ năng, trực quan và kinh nghiệm của kỹ sư phần mềm với cá chương trình quen thuộc. Kiểm thử tùy biến có thể hữu ích cho việc định danh các trường hợp kiểm thử mà không dễ thực hiện bằng các kỹ thuật bình thường.

3.1.2. Kiểm thử khám phá

Kiểm thử theo kiểu khám phá được định nghĩa là tiến hành đồng thời việc học tập, thiết kế và thực hiện kiểm thử, nó không được thực hiện theo kế hoạch kiểm thử đã được lên trước và linh động về việc thiết kế, thực hiện và sửa đổi. Hiệu quả của kiểm thử khám phá phụ thuộc và kinh nghiệm của kỹ sư phần mềm, điều này có được từ nhiều nguồn như quan sát hành vi của sản phẩm trong quá trình kiểm thử, sự quen thuộc với các ứng dụng, môi trường, quá trình lỗi, các dạng lỗi có thể có, nguy cơ khi tích hợp với các sản phẩm riêng biệt....

3.2. Kỹ thuật dựa trên miền đầu vào

3.2.1. Phân vùng tương đương

Phân vùng tương đương liên quan đến việc phân chia miền đầu vào thành các vùng dựa trên tiêu chuẩn hoặc quan hệ. Tiêu chuẩn hoặc quan hệ này có thể là những kết quả tính toán khác nhau, quan hệ dựa trên luồng điều khiển hoặc luồng dữ liệu, hoặc một sự phân biệt được tạo ra giữa các đầu vào hợp được chấp nhận và đầu vào không hợp lệ sẽ bị từ chối và đưa ra lỗi.

3.2.2. Kiểm thử theo cặp

Các trường hợp kiểm thử có được bằng việc kết hợp các giá trị tiêu biểu cho mỗi cặp của một tập hợp biến đầu vào thay vì xét đến toàn bộ các kết hợp có thể có. Kiểm thử theo cặp thuộc về kiểm thử tổ hợp, nhìn chung bao gồm mức tổ hợp cao hơn so với cặp: những kỹ thuật này được xem như t-wise, cái mà toàn bộ tổ hợp đầu vào t được xem xét.

3.2.3. Phân tích giá trị biên

Các trường hợp kiểm thử được chọn ra sẽ nằm trên hoặc gần với biên của miền đầu vào, nơi tỉ lệ lỗi thường tập chung cao nhất. Trường hợp ngoại lệ của kỹ thuật này là kiểm thử sự bền vững, nơi mà trường hợp kiểm thử được chọn bên ngoài miền biến đầu vào để kiểm thử sự bền vững của sản phẩm trong việc xử lý đầu vào lỗi.

3.2.4. Kiểm thử ngẫu nhiên

Các kiểm thử được thực hiện đơn thuần một cách ngẫu nhiên. Dạng kiểm thử này đặt dưới đầu của miền đầu vào bởi vì miền đầu vào phải được biến đổi để có thể lấy ra được những điểm ngẫu nhiên. Kiểm thử ngẫu nhiên cung cấp một tiếp cận đơn giản để kiểm thử tự động, gần đây các dạng nâng cao của kiểm thử ngẫu nhiên được đưa ra, mẫu đầu vào

ngẫu nhiên được hướng bởi tiêu chuẩn lựa chọn đầu vào khác. Kiểm thử Fuzz hay Fuzzing là dạng đặc biệt của kiểm thử ngẫu nhiên, tập chung vào việc phá vỡ phần mềm, nó chủ yếu được dùng để kiểm thử an ninh của phần mềm.

3.3. Kỹ thuật dựa trên mã nguồn

3.3.1. Tiêu chuẩn dựa trên luồng điều khiển

Tiêu chuẩn phủ dựa trên luồng điều khiển được tập chung vào việc phủ toàn bộ các câu lệnh, khối lệnh hoặc kết hợp đặc biệt của các câu lệnh trong chương trình. Điểm mạnh nhất của tiêu chuẩn này là kiểm thử đường đi, cái mà tập chung vào để thực thi toàn bộ đường đi luồng điều khiển từ khi vào đến khi kết thúc trong biểu đồ luồng điều khiển của chương trình. Vì kiểm thử vét kiệt đường đi là không khả thi do vòng lặp, tiêu chuẩn ít chính xác tập chung vào độ phủ của đường đi các mà giới hạn các bước lặp như phủ câu lệnh, phủ nhánh và kiểm thử quyết định. Sự đầy đủ của các kiểm thử như thế được đo đạc theo phần trăm, ví dụ, khi tất cả các nhánh được thử thi ít nhất một lần khi kiểm thử thì độ phủ nhánh là 100%.

3.3.2. Tiêu chuẩn dựa trên luồng dữ liệu

Trong kiểm thử dựa trên luồng dữ liệu, biểu đồ luồng điều khiển được diễn giải với thông tin về việc bằng cách nào các biến chương trình được định nghĩa, sử dụng và kết thúc. Tiêu chuẩn mạnh nhất yêu cầu từ định nghĩa của biến đến việc sử dụng định nghĩa đó được thực thi. Để giảm số đường đi yêu cầu, những chiến lược yếu hơn như định nghĩa tất cả và sử dụng tất cả được dùng.

3.3.3. Các mô hình tham khảo cở kiểm thử dựa trên mã nguồn

Mặc dù bản thân không phải là một kỹ thuật, cấu trúc điều khiển của một chương trình có thể được hình ảnh hoá để biểu diễn việc sử dụng một biểu đồ như kỹ thuật dựa trên mã nguồn. Một biểu đồ là đồ thị có hướng, các nút và cung biểu diễn cho các yếu tố của chương trình. Ví dụ, các nút có thể biểu diễn các lệnh hoặc tần xuất không bị can thiệp của lệnh và cung có thể biểu diễn sử trao đổi giữa các nút.

3.4. Kỹ thuật dựa trên lỗi

3.4.1. Dự đoán lỗi

Trong việc dự đoán lỗi, các trường hợp kiểm thử được thiết kế đặc biệt bởi kỹ sư phần mềm người mà cố gắng dự đoán trước các lỗi có thể xảy ra trong chương trình cho trước. Một nguồn thông tin tốt là lịch sử lỗi được tìm ra trong những dự án có trước cũng như kỹ năng của kỹ sư phần mềm.

3.4.2. Kiểm thử đột biến

Một đột biến là một phiên bản chỉnh sửa nhẹ của chương trình đang kiểm thử, với một số thay đổi nhẹ để khác biệt với chương trình cũ. Mỗi lần kiểm thử là kiểm thử phiên bản gốc và toàn bộ các đột biến: nếu trường hợp kiểm thử thành công trong việc định danh sự khác biệt giữa chương trình và đột biến thì đột biến sẽ bị tiêu diệt. Nói cách khác, khi một đột biến được kiểm thử mà trả kết quả khác so với bản gốc thì đột biến bị tiêu diệt và xem như bản gốc đạt tiêu chuẩn. Khi bản đột biến mà cho kết quả giống bản gốc thì xem như kiểm thử thất bại. Hiệu quả của kỹ thuật này được đánh giá bằng số lượng lớn của các đột biến bị tiêu diệt. Các đột biến thường được tạo ra một cách tự động và kiểm thử thực hiện theo một cách có hệ thống.

3.5. Kỹ thuật dựa trên việc sử dụng

3.5.1. Hồ sơ vận hành

Trong việc kiểm thử sự đánh giá độ tin cậy, môi trường kiểm thử được tạo ra gần với môi trường vận hành của phần mềm hoặc gọi là hồ sơ vận hành. Mục tiêu là để suy ra từ các kết quả đã được theo dõi độ tin cậy tương lai của phần mềm khi được sử dụng thật. Để làm điều này, đầu vào được gán các xác suất hoặc hồ sơ theo tần suất xảy ra trong thực tế. Hồ sơ vận hành so thể được dùng trong kiểm thử hệ thống để hướng dẫn bắt nguồn của các trường hợp kiểm thử sẽ đánh giá sự tin cậy và thử nghiệm liên quan đến việc sử dụng và các chức năng khác nhau giống với những gì xảy ra trong môi trường kiểm thử vận hành.

3.5.2. Quan sát người dùng

Các nguyên tắc sử dụng có thể cung cấp hướng dẫn cho việc tìm ra vấn đề trong thiết kế giao diện. Nguồn thông tin cho vấn đề này có thể được lấy qua các nguồn như phỏng vấn và làm khảo sát người dùng.

3.6. Kỹ thuật dựa trên mô hình

3.6.1. Bảng quyết định

Những bảng quyết định biểu diễn các mối quan hệ logic giữa các điều kiện và hành động. Các trường hợp kiểm thử có được một cách hệ thống bằng việc xem xét mọi tổ hợp của điều kiện và các hành động phản ứng. Một kỹ thuật liên quan là đồ thị nguyên nhân - ảnh hưởng.

3.6.2. Máy trạng thái hữu hạn

Bằng việc mô hình hóa một chương trình như một máy trạng thái hữu hạn, việc kiểm thử được lựa chọn để bao phủ các trạng thái và sự chuyển tiếp.

3.6.3. Chi tiết kỹ thuật thông thường

Trạng thái hóa cả chi tiết kỹ thuật trong ngôn ngữ hình thức cho phép sự dẫn xuất tự động các trường hợp kiểm thử chức năng và cùng lúc cung cấp một dự đoán cho việc kiểm tra kết quả kiểm thử. TTCN3 là một ngôn ngữ được phát triển để viết ra các trường hợp kiểm thử. Định nghĩa được hiểu với những yêu cầu riêng biệt của kiểm thử các hệ thống truyền thông, vì thế nó đặc biệt thích hợp cho kiểm thử các giao thức truyền thông phức tạp.

3.6.4. Mô hình luồng công việc

Mô hình luồng công việc chỉ ra tần suất của các hoạt động được thực hiện bởi con người hay ứng dụng phần mềm, thường được biểu diễn thông qua các định nghĩa hình ảnh. Mỗi tần suất của hành động cấu thành một luồng công việc.

3.7. Kỹ thuật dựa trên tính chất của phần mềm

Các kỹ thuật bên trên áp dụng cho toàn bộ các loại phần mềm. Các kỹ thuật phụ cho kiểm thử dựa trên tính chất của phần mềm như:

- Phần mềm hướng đối tượng
- Phần mềm dựa trên bộ phận
- Phần mềm nền web
- Các chương trình tương tranh
- Phần mềm dựa trên giao thức
- Hệ thống thời gian thực
- Hệ thống an toàn nguy cấp
- Phần mềm hướng dịch vụ
- Phần mềm mã nguồn mở
- Phần mềm nhúng

3.8. Chọn và kết hợp các kỹ thuật

3.8.1. Kết hợp chức năng và cấu trúc

Các kỹ thuật kiểm thử dựa trên mô hình và mã nguồn thường tương phản như kiểm thử chức năng và cấu trúc. Hai cách tiếp cận lựa chọn kiểm thử này không được xem như thay thế cho nhau mà được xem là bổ sung cho nhau; trong thực tế, chúng sử dụng các nguồn thông tin khác nhau và đã chỉ ra các loại vấn đề tiêu biểu khác nhau. Chúng có thể được dùng kết hợp và phụ thuộc vào xem xét đầu tư.

3.8.2. Tắt định và ngẫu nhiên

Các trường hợp kiểm thử được chọn theo cách tắt định, theo một trong số các kỹ thuật hoặc thực hiện ngẫu nhiên từ sự phân phối đều vào như trong kiểm thử độ tin cậy. Một vài so sánh theo kiểu phân tích và kinh nghiệm được chọn để phân tích các điều kiện mà tạo cách tiếp cận có hiệu quả hơn.

4. Các phép đo liên quan đến việc kiểm thử

4.1. Đánh giá chương trình đang kiểm thử

4.1.1. Các phép đo chương trình hỗ trợ lên kế hoạch và thiết kế kiểm thử

Các phép đo dựa trên kích cỡ của phần mềm (ví dụ: các dòng mã nguồn hoặc kích thước chức năng) hoặc cấu trúc phần mềm có thể được dùng để hướng dẫn kiểm thử. Các phép đo cấu trúc bao gồm các phép đo mà quyết định tần suất của mô đun này gọi thực hiện mô đun khác.

4.1.2. Các loại lỗi, phân loại và thống kê

Văn học kiểm thử đa dạng trong phân loại và nguyên tắc phân loại lỗi. Để tạo ra kiểm thử hiệu quả hơn thì quan trọng là biết loại lỗi nào có thể được phát hiện trong phần mềm đang kiểm thử và tần xuất tương đối những lỗi đó tồn tại trong quá khứ. Thông tin này có thể hữu ích trong việc dự đoán chất lượng cũng như trong việc cải tiến quá trình.

4.1.3. Mật độ lỗi

Một chương trình đang kiểm thử có thể được đánh giá bằng việc tính toán các lỗi đã được tìm ra như tỉ lệ giữa số lỗi được tìm thấy và độ lớn của chương trình.

4.1.4. Đánh giá độ tin cậy

Một ước lượng thông kê của độ tin cậy phần mềm có thể được dùng để đánh giá một sản phẩm phần mềm và quyết định kiểm thử có thể dừng lại hay không bằng việc quan sát độ tin cậy được tạo ra.

4.1.5. Những mô hình phát triển độ tin cậy

Những mô hình phát triển độ tin cậy cung cấp dự đoán độ tin cậy dựa trên lỗi. Chúng giả định rằng các lỗi gây ra các thất bại được quan sát đều được sửa, độ tin cậy của sản phẩm đưa ra một xu thế tăng dần. Rất nhiều mô hình kiểu này được đưa ra. Đáng chú ý, những mô hình này được chỉ ra là 2 loại là mô hình đếm lỗi và mô hình thời gian giữa các lỗi.

4.2. Đánh giá các kiểm thử được thực hiện

4.2.1. Các phép đo phủ

Một vài tiêu chuẩn kiểm thử yêu cầu rằng các trường hợp kiểm thử thực hiện một tập yếu tố được định danh trong chương trình hoặc trong chi tiết kỹ thuật một các hệ thống. Để đánh giá độ chi tiết của các kiểm thử được thực hiện, các kỹ sư phần mềm cần thể theo dõi các yếu tố được bao phủ để họ có thể đo đạc tỉ lệ giữa yếu tố được phủ và tổng số một các linh động. Ví dụ, có thể đo tỉ lệ phần trăm của các nhánh được phủ trong biểu đồ dòng chảy

chương trình hoặc phần trăm yêu cầu chức năng được thực hiện và danh sách chức năng trong tài liệu chi tiết kỹ thuật. Các tiêu chí phù hợp dựa trên mã nguồn yêu cầu sự đo đạc thích hợp của chương trình đang kiểm thử.

4.2.2. Gieo lỗi

Trong việc gieo lỗi, vài lỗi được đưa nhân tạo vào chương trình trước khi kiểm thử. Khi kiểm thử được tiến hành, và trong số này sẽ được bộc lộ cũng như và lỗi có sẵn. Theo lý thuyết, phụ thuộc vào việc cái nào và bao nhiêu lỗi nhân tạo được tìm ra, hiệu quả kiểm thử và số lỗi thực có thể được đánh giá. Trong thực tế, các nhà thống kê đặt ra câu hỏi thử phân phối và biểu diễn của các lỗi được gieo trước đó quan hệ với lỗi thật và một cỡ bé mẫu thử lên bất kỳ phéo ngoại suy nào được dựa trên. Một số khác tranh cãi rằng kỹ thuật này nên dùng thận trọng vì việc chèn lỗi vào phần mềm liên quan đến mạo hiểm rõ ràng rằng có thể quên các lỗi đó trong chương trình.

4.2.3. Điểm đột biến

Trong kiểm thử đột biến, tỉ lệ các đột biến bị tiêu diệt với tổng số đột biến có thể là phép đo cho sự hiệu của của tập kiểm thử được thực hiện.

4.2.4. Sự so sánh và hiệu quả tương đối của các kỹ thuật khác nhau

Và nghiên cứu được tiến hành để so sánh hiệu quả tương đối của việc sử dụng các kỹ thuật khác nhau. Quan trọng là để được chính xác như đối với thuộc tính dựa vào đó các kỹ thuật đang được đánh giá. Cách diễn dải có thể bao gồm số các kiểm thử cần để tìm ra lỗi đầu tiên, tỉ lệ của số lỗi được tìm ra thông qua kiểm thử và toàn bộ số lỗi được tìm ra trong và sau khi kiểm thử, và cải thiện độ tin cậy được bao nhiêu. Các so sánh phân tích và thực nghiệm giữa các kỹ thuật khác nhau được tiến hành theo từng ý niệm về hiệu quả quy định ở trên.

5. Quá trình kiểm thử

Khái niệm về kiểm thử, chiến lược, kỹ thuật và đo lường cần phải được tích hợp vào một quá trình xác định và kiểm soát. Quá trình kiểm thử hỗ trợ các hoạt động kiểm tra và cung cấp hướng dẫn cho các tester và đội tester, từ việc lập kế hoạch để đánh giá đầu ra, theo cách như vậy để đảm bảo cung cấp rằng các mục tiêu kiểm thử sẽ được đáp ứng trong một chi phí hiệu quả kịp thời.

5.1. Nhận nhận thực tế

5.1.1. Thái độ / Lập trình giảm thiểu “cái tôi”

Một yếu tố quan trọng của kiểm thử thành công là một thái độ hợp tác hướng tới thử nghiệm và hoạt động đảm bảo chất lượng. Những người quản lý có vai trò quan trọng trong việc thúc đẩy một tiếp nhận thuận lợi nói chung để hướng tới phát hiện lỗi và sửa chữa

trong quá trình phát triển và bảo trì phần mềm.

Ví dụ: bằng cách vượt qua những suy nghĩ của quyền sở hữu cá nhân giữa các lập trình và bằng cách thúc đẩy một môi trường cộng tác với đội ngũ chịu trách nhiệm về bất thường trong các mã lập trình.

5.1.2. Hướng dẫn kiểm thử

Các giai đoạn kiểm thử có thể được hướng dẫn bởi các mục tiêu khác nhau.

Ví dụ: Kiểm thử dựa trên rủi ro sử dụng các rủi ro sản phẩm ưu tiên và tập trung các chiến lược kiểm thử, và kiểm thử dựa trên kịch bản xác định các trường hợp kiểm thử dựa trên kịch bản phần mềm quy định.

5.1.3. Quản lý quá trình kiểm thử

Các hoạt động kiểm thử được tiến hành ở các cấp độ khác nhau (xem thêm ở level 2: Test levels) phải được tổ chức lại với nhau, cùng với con người, công cụ, chính sách và các biện pháp thành phần xác định, đó là một phần không thể thiếu của vòng đời.

5.1.4. Tài liệu kiểm thử và sản phẩm công việc

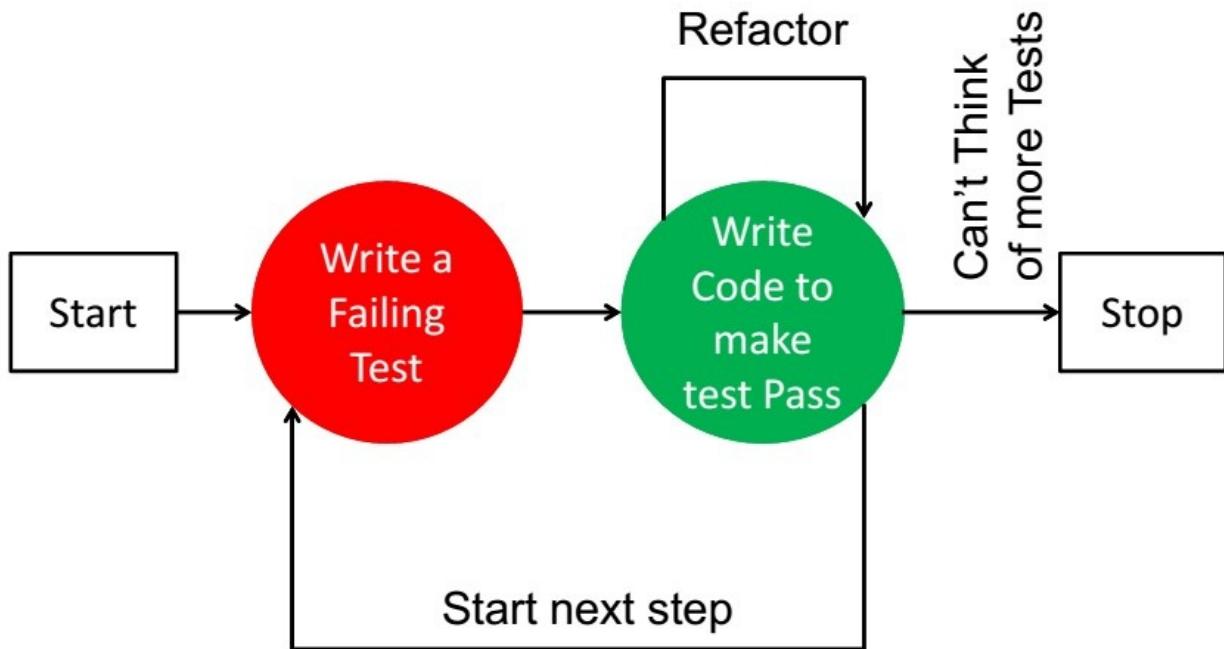
Tài liệu kiểm thử là một phần không thể thiếu khi chính thức quá trình kiểm thử. Tài liệu kiểm thử có thể bao gồm các kế hoạch kiểm tra, đặc điểm kỹ thuật thiết kế kiểm tra, đặc điểm kỹ thuật quy trình kiểm tra, kiểm tra các trường hợp đặc điểm kỹ thuật, kiểm tra log và báo cáo sự việc kiểm tra.

Các phần mềm được kiểm thử được ghi nhận như một tác phẩm kiểm thử. Tài liệu kiểm thử nên được tạo ra và liên tục cập nhật tới cùng một mức độ chất lượng như các loại tài liệu khác trong công nghệ phần mềm. Tài liệu kiểm thử cũng nên chịu sự giám sát của quản lý cấu hình phần mềm. Hơn nữa, tài liệu kiểm thử bao gồm các sản phẩm công việc mà có thể cung cấp các tài liệu hướng dẫn cho người sử dụng và đào tạo.

5.1.5. Phát triển hướng kiểm thử (TDD)

Phát triển hướng kiểm thử (TDD) ban đầu là một trong những thực hành cốt lõi XP (extreme programming) và bao gồm các kiểm thử đơn vị bằng văn bản trước khi các mã lập trình được kiểm tra (xem phương pháp Agile trong mô hình công nghệ phần mềm và phương pháp KA).

Bằng cách này, TDD phát triển các trường hợp kiểm thử như là một thay thế cho tài liệu đặc tả yêu cầu phần mềm hơn là một kiểm tra độc lập mà phần mềm đã thực hiện đúng các yêu cầu. Thay vì một chiến lược kiểm thử, TDD là một thực tế đòi hỏi các nhà phát triển phần mềm để xác định và duy trì kiểm tra đơn vị (unit tests). Điều này có thể có một động thái tích cực vào việc xây dựng nhu cầu người dùng và các đặc tả yêu cầu phần mềm.



Ví dụ: một mô hình khi áp dụng TDD

Phát triển hướng kiểm thử TDD (Test-Driven Development) là một phương pháp tiếp cận cải tiến để phát triển phần mềm trong đó kết hợp phương pháp Phát triển kiểm thử trước (Test First Development) và phương pháp Điều chỉnh lại mã nguồn (Refactoring).

5.1.6. Đội kiểm thử nội bộ và độc lập

Quá trình kiểm thử cũng có thể liên quan đến việc tổ chức các nhóm kiểm thử. Các nhóm kiểm thử có thể bao gồm các thành viên nội bộ (có nghĩa là, trong nhóm dự án, tham gia hay không trong xây dựng phần mềm), các thành viên bên ngoài (với hy vọng mang lại một quan điểm độc lập, không thiên vị), hoặc cả hai thành viên nội bộ hay bên ngoài. Cần nhắc về chi phí, thời gian, mức độ trưởng thành của tổ chức có liên quan và quan trọng của các ứng dụng có thể hướng dẫn các quyết định.

5.1.7. Chi phí / dự toán và đo lường quá trình kiểm thử

Một số các biện pháp liên quan đến các nguồn tài nguyên dành cho việc kiểm tra, cũng như hiệu quả tìm kiếm lỗi tương ứng của các giai đoạn kiểm thử khác nhau, được sử dụng bởi các nhà quản lý để kiểm soát và cải thiện quá trình kiểm thử. Những biện pháp kiểm tra có thể bao gồm các khía cạnh như số trường hợp kiểm thử được chỉ định, số trường hợp kiểm tra được thực hiện, số lượng trường hợp kiểm thử đạt yêu cầu, số lượng trường hợp kiểm thử không đạt yêu cầu trong số những cái khác.

Thẩm định báo cáo giai đoạn kiểm thử có thể được kết hợp với phân tích nguyên nhân gốc để đánh giá hiệu quả quá trình kiểm thử trong việc tìm lỗi càng sớm càng tốt. Việc đánh giá đó có thể kết hợp với việc phân tích rủi ro. Hơn nữa, các tài nguyên dành cho việc kiểm thử

phải phù hợp với việc sử dụng/quan trọng của các ứng dụng: các kỹ thuật khác nhau có các chi phí khác nhau và mang lại mức độ khác nhau của độ tin tưởng trong mức độ tin cậy của sản phẩm.

5.1.8. Kết thúc kiểm thử

Một quyết định cần phải đưa ra là việc kiểm thử như thế nào là đủ và khi nào giai đoạn kiểm thử được chấm dứt. Thông qua đo đạc, xác định các chức năng đảm bảo, cũng như ước tính mật độ lỗi hoặc độ tin cậy của hoạt động.

Các quyết định cũng bao gồm việc xem xét về chi phí và rủi ro phát sinh bởi sự thất bại còn có thể xảy ra, như trái ngược với chi phí phát sinh bằng cách tiếp tục kiểm thử (xem tiêu chuẩn lựa chọn kiểm thử / tiêu chuẩn thử nghiệm Adequacy ở mục 1.2)

5.1.9. Tái sử dụng kiểm thử và Mẫu kiểm thử

Để thực hiện kiểm thử, bảo trì một cách có tổ chức và hiệu quả chi phí, các kiểm thử từng phần của phần mềm nên được tái sử dụng một cách hệ thống. Một kho lưu trữ kiểm thử nên chịu sự kiểm soát của quản lý cấu hình phần mềm để mà thay đổi yêu cầu phần mềm hoặc thiết kế có thể được phản ảnh trong thay đổi cho quản lý các kiểm thử.

Các giải pháp kiểm thử được áp dụng để kiểm thử một số loại ứng dụng trong những hoàn cảnh nhất định, với những động cơ đăng sau các quyết định đưa ra, tạo thành một mô hình kiểm thử mà chính nó có thể được ghi lại để tái sử dụng sau này trong các dự án tương tự.

5.2. Hoạt động kiểm thử

Như thể hiện trong các mô tả dưới đây, quản lý thành công của các hoạt động kiểm thử phụ thuộc mạnh mẽ vào quá trình quản lý cấu hình phần mềm (xem quản lý cấu hình phần mềm KA)

5.2.1. Lập kế hoạch

Giống như tất cả các khía cạnh khác nhau của quản lý dự án, các hoạt động kiểm thử phải được quy hoạch. Khía cạnh quan trọng của việc lập kế hoạch kiểm thử bao gồm sự phối hợp của các nhân viên, tính sẵn có của các cơ sở và trang thiết bị kiểm thử, tạo ra và bảo trì tất cả các tài liệu liên quan đến kiểm thử, và lên kế hoạch cho những kết quả không mong muốn có thể. Nếu có nhiều hơn cơ sở của các phần mềm đang duy trì, sau đó xem xét một kế hoạch chính là thời gian và nỗ lực cần thiết để đảm bảo rằng các môi trường kiểm thử được thiết lập để cấu hình đúng.

5.2.2. Tạo ra các trường hợp kiểm thử

Việc tạo ra các trường hợp kiểm thử dựa trên mức độ kiểm thử được thực hiện và kỹ thuật kiểm thử cụ thể. Các trường hợp kiểm thử nên được dưới sự kiểm soát của quản lý cấu hình phần mềm và bao gồm các kết quả dự kiến cho mỗi kiểm thử.

5.2.3. Phát triển môi trường kiểm thử

Môi trường được sử dụng để kiểm thử cần có sự tương thích với các công cụ phần mềm được đề nghị. Nó tạo điều kiện phát triển và kiểm soát các trường hợp kiểm thử, cũng giống như ghi lại và phục hồi các kết quả mong đợi, kịch bản và các tài liệu kiểm thử khác.

5.2.4. Thực hiện kiểm thử

Thực hiện các kiểm thử nên thể hiện một nguyên tắc cơ bản của thí nghiệm khoa học: tất cả mọi thứ được thực hiện trong quá trình kiểm thử nên được thực hiện và ghi chép rõ ràng đủ để người khác có thể nhân rộng kết quả. Do đó, việc kiểm thử nên được thực hiện theo thủ tục bằng cách sử dụng một phiên bản xác định rõ ràng của phần mềm bên dưới các kiểm thử.

5.2.5. Đánh giá kết quả kiểm thử

Các kết quả của kiểm thử cần được đánh giá để xác định việc kiểm thử có thành công hay không. Trong hầu hết các trường hợp, thành công nghĩa là phần mềm thực hiện như mong đợi và không có các kết quả bất ngờ nào. Không phải tất cả các kết quả bất ngờ là nhất thiết phải lỗi.

Trước khi một lỗi có thể được xóa đi, một phân tích và nỗ lực gỡ lỗi là cần thiết để cô lập, xác định và mô tả nó. Khi kết quả kiểm thử là đặc biệt quan trọng, một hội đồng xét duyệt chính thức có thể được triệu tập để đánh giá chúng.

5.2.6. Báo cáo / nhật ký kiểm thử

Hoạt động kiểm thử có thể sử dụng các nhật ký kiểm thử để xác định một kiểm thử đã được tiến hành, ai là người thực hiện các kiểm tra, những cấu hình phần mềm nào được sử dụng và các thông tin nhận dạng khác có liên quan. Kết quả kiểm thử đột xuất hoặc không chính xác có thể được ghi lại trong một hệ thống báo cáo sự cố, các dữ liệu mà tạo cơ sở cho việc gỡ lỗi sau đó và sửa chữa các vấn đề đã được quan sát như những thất bại trong quá trình kiểm thử. Ngoài ra, bất thường không được phân loại là lỗi có thể được ghi lại trong trường hợp chúng lần lượt xuất hiện là nghiêm trọng hơn nhiều so với suy nghĩ đầu tiên. Báo cáo kiểm thử cũng là đầu vào để thay đổi quy trình quản lý theo yêu cầu (xem điều khiển cấu hình phần mềm trong quản lý cấu hình phần mềm KA)

5.2.7. Theo dõi khiếm khuyết

Các khiếm khuyết có thể được theo dõi và phân tích để xác định khi chúng được đưa vào phần mềm, tại sao chúng được tạo ra, (ví dụ, yêu cầu kém được xác định, khai báo biến không đúng, rò rỉ bộ nhớ, lập trình lỗi cú pháp), và khi họ có thể quan sát thấy lần đầu tiên trong phần mềm. Khiếm khuyết về thông tin được sử dụng để xác định những khía cạnh của kiểm thử phần mềm và các quá trình khác cần cải thiện và hiệu quả của phương pháp tiếp cận trước đây diễn ra như thế nào.

6.Công cụ kiểm thử phần mềm

6.1.Công cụ kiểm thử hỗ trợ

Kiểm thử đòi hỏi nhiều công việc cần thực hiện, chạy nhiều chương trình và xử lý một lượng lớn thông tin. Các công cụ thích hợp có thể làm giảm bớt gánh nặng của việc lặp lại, hoạt động tẻ nhạt và làm cho việc kiểm thử ít dễ bị lỗi. Công cụ tinh vi có thể hỗ trợ thiết kế kiểm thử và tạo ra các trường hợp kiểm thử, làm cho nó hiệu quả hơn.

6.1.1. Lựa chọn công cụ

Hướng dẫn cho các nhà quản lý và các người kiểm thử về cách chọn công cụ kiểm tra rằng sẽ hữu ích nhất để tổ chức và các quy trình là một chủ đề rất quan trọng, như là một công cụ tạo ảnh hưởng lớn để hiệu quả kiểm thử và thật sự hiệu quả. Việc lựa chọn công cụ phụ thuộc vào các dấu hiệu khác nhau, chẳng hạn như lựa chọn phát triển, mục tiêu đánh giá, các cơ sở thực hiện,...Nói chung, có thể không có một công cụ duy nhất để đáp ứng các nhu cầu cụ thể, do đó một bộ công cụ có thể là một lựa chọn thích hợp.

6.2. Danh mục các công cụ kiểm thử.

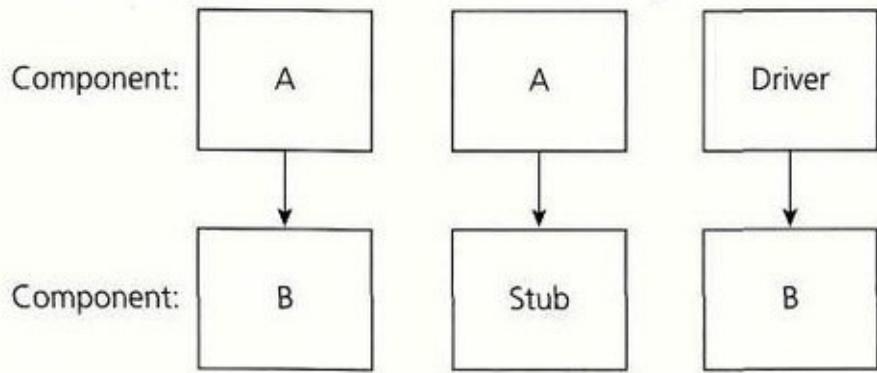
Chúng ta phân loại các công cụ có sẵn theo chức năng của chúng:

- **Test harnesses (drivers, stubs):**

- Trong kiểm thử phần mềm, test harness (khai thác kiểm thử) để cập đến một bộ sưu tập các phần mềm và bộ dữ liệu kiểm thử được cấu hình sẵn để kiểm thử một đơn vị chương trình bằng cách chạy nó trong nhiều điều kiện khác nhau và chúng ta sẽ theo dõi hành vi và kết quả đầu ra của nó.
- Cung cấp một môi trường điều khiển, trong đó việc kiểm thử có thể được thực hiện và kết quả đầu ra có thể được ghi lại. Để thực hiện kiểm thử các phần của một chương trình, các drivers và stubs được cung cấp để mô phỏng một cách tương ứng.
- Sự khác nhau giữa Drivers và Stubs:

Drivers: là loại giả mô-đun, cái được gọi là "các chương trình gọi", được sử dụng trong kiểm thử tích hợp từ dưới lên, được sử dụng khi chương trình chính đang được xây dựng.

Stub: có nghĩa là một mô hình giả của một module cụ thể.



- Ví dụ:

Giả sử chúng ta cần kiểm thử sự tích hợp giữa mô đun A và B và chúng ta đã xây dựng được mô đun A trong khi mô đun B chưa có trong giai đoạn phát triển.

Như vậy, khi cần kiểm thử mô đun A mà chưa có mô đun B, chúng ta cần tạo ra một giả mô đun để thay thế cho mô đun B, có tính năng tương tự như mô đun B sau đó có thể sử dụng. Trường hợp mô đun giả này được gọi là Stub. Trường hợp ngược lại, chẳng hạn chúng ta đã xây dựng xong mô đun B nhưng mô đun A chưa xây dựng xong, để kiểm thử mô đun B, chúng ta sẽ sử dụng một giải mô đun khác truy xuất dữ liệu đến mô đun B được gọi là Driver.

- **Test generators:** cung cấp các hỗ trợ trong trường hợp tự tạo ra các dữ liệu thực nghiệm cho chương trình để kiểm thử. Các dữ liệu kiểm thử được tạo ra có thể là ngẫu nhiên hoặc dựa trên một yếu tố nào đó như: dựa trên tìm đường đi, dựa trên mô hình, hoặc hỗn hợp của chúng.

- **Capture/replay tools:**

- Nhóm công cụ sử dụng để ghi và phát lại, các kiểm thử viên có thể chạy một ứng dụng và ghi lại sự tương tác giữa người dùng và các ứng dụng. Một kịch bản được ghi lại với tất cả hành động của người dùng bao gồm cả việc di chuyển chuột và các công cụ sau đó có thể tự động thực thi lại các phiên tương tác một cách không hạn chế số lần mà không yêu cầu sự can thiệp của con người.
- Một số công cụ sử dụng trong nhóm:

Công cụ	Nhà cung cấp	Địa chỉ website
QF-Test	QFS	www.qfs.de/en/qftest/
SWTBot	OpenSource	http://wiki.eclipse.org/SWTBot/UsersGuide
GUIDancer and Jubula	BREDEX	http://testing.bredex.de/
TPTP GUI Recorder	Eclipse	http://www.eclipse.org/tptp/

- **Oracle/file comparators/assertion checking tools:**

- Hỗ trợ trong việc quyết định một kết quả kiểm thử có thành công hay không.

- Ví dụ: công cụ Navicat for oracle: là một bộ công cụ của hãng PremiumSoft, có các tính năng phục vụ cho Developer và Administrator như: Database Designer, Table Viewer, SQL Builder, PL/SQL code Debugger,...

• **Coverage analyzers and instrumenters:**

- Phân tích mức độ bao phủ trong tất cả những yêu cầu của các tiêu chuẩn đã được lựa chọn kiểm thử. Các phân tích có thể được thực hiện nhờ vào các công cụ chương trình.
- Ví dụ: COVTOOL là công cụ kiểm thử coverage analyzers sử dụng cho C++ (Link tham khảo: <http://covtool.sourceforge.net/>)

• **Tracers:** ghi lại lịch sử của đường dẫn thực thi một chương trình.

• **Regression testing tools:**

- Các công cụ nhóm này được dùng để kiểm tra lại các phần chức năng của ứng dụng khi có một phần của phần mềm đã được sửa đổi. Trường hợp kiểm thử được tái thực hiện để kiểm tra rằng các chức năng trước đó của ứng dụng đang hoạt động bình thường có hoạt động ổn định nữa không khi có một sự thay đổi được cập nhật vào ứng dụng.
- Đây là phương pháp xác minh. Thẩm định rằng các lỗi được sửa và các tính năng mới được bổ sung đã không được tạo ra các lỗi trong phiên bản làm việc trước của phần mềm.
- Công cụ này cũng giúp đỡ để chọn một tập hợp kiểm thử theo các thay đổi được thực hiện.
- Một số công cụ sử dụng trong nhóm:

Công cụ	Link tham khảo
Winrunner	http://www.mercury.com/us/products/quality-center/functional-testing/winrunner/
QTP	http://www.mercury.com/us/products/
AdventNet QEngine	http://www.adventnet.com/products/qengine/index.html
Regression Tester	http://www.regressiontester.com/
vTest	http://www.verisium.com/products/vTest/index.html
Watir	http://www.openqa.org/watir/
Selenium	http://www.openqa.org/selenium/
actiWate	http://www.actiivate.com/
Rational Functional Tester	http://www.ibm.com/products/us/
SilkTest	http://www.borland.com/

- **Reliability evaluation tools:** hỗ trợ phân tích kết quả kiểm thử và hiển thị đồ họa để đánh giá độ tin cậy liên quan đến các biện pháp theo mô hình lựa chọn.

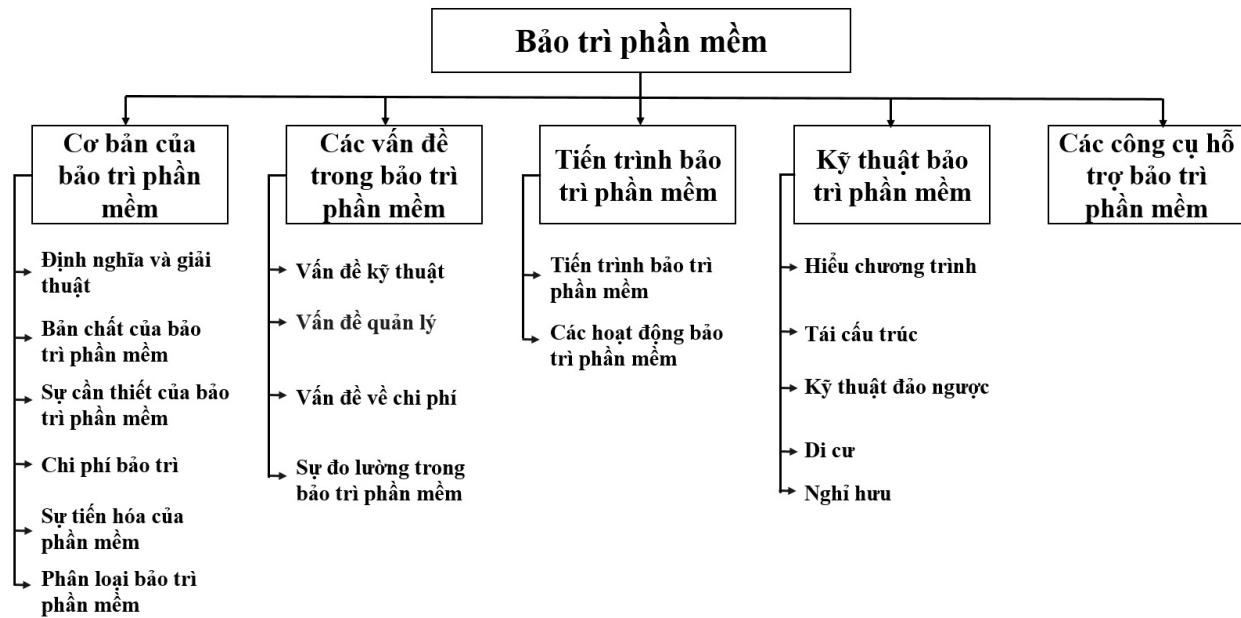
Chương 5: Bảo trì phần mềm

Các thuật ngữ:

- MR (modification request): yêu cầu chỉnh sửa.
- PR (problem report): báo cáo các vấn đề.
- SCM (software configuration management): quản lý cấu hình phần mềm.
- SLA (service level agreement): thỏa thuận dịch vụ theo cấp độ.
- SQA (software quality assurance): quản lý chất lượng phần mềm.
- V&V (verification and validation): xác minh và thẩm định.
- Predelivery: Trước khi bàn giao
- Postdelivery: Sau khi phân phối

Giới thiệu

Nỗ lực phát triển phần mềm dẫn tới khi chuyển giao sản phẩm phải thỏa mãn yêu cầu của khách hàng. Theo đó các sản phẩm phần mềm phải thay đổi và phát triển. Sau khi hoạt động, các lỗi được phát hiện, môi trường hoạt động thay đổi và yêu cầu người dùng mới phát sinh. Giai đoạn bảo trì trong vòng đời phát triển của phần mềm sẽ bắt đầu sau thời gian bảo hành, hoặc sau khi bàn giao sản phẩm, nhưng đa phần hoạt động này sẽ xảy ra sớm hơn. Bảo trì phần mềm là một phần không thể thiếu trong vòng đời phần mềm. Trong lịch sử, phát triển phần mềm được đánh giá cao hơn nhiều so với bảo trì phần mềm trong hầu hết các tổ chức. Điều này đang thay đổi, các tổ chức đã đầu tư vào khâu phát triển phần mềm của họ bằng cách giữ cho phần mềm của họ hoạt động càng lâu càng tốt. Trong tài liệu này, bảo trì phần mềm được xác định là toàn bộ các hoạt động cần thiết để cung cấp sự hỗ trợ một cách hiệu quả về chi phí cho phần mềm. Các hoạt động được thực hiện trong giai đoạn trước khi bàn giao cũng như trong giai đoạn sau khi bàn giao. Hoạt động trước khi bàn giao bao gồm lập kế hoạch cho các hoạt động sau khi bàn giao, bảo trì, và hậu cần xác định các hoạt động chuyển đổi. Hoạt động sau khi bàn giao bao gồm chỉnh sửa phần mềm, training, điều hành hoặc tương tác. Kiến thức về lĩnh vực bảo trì phần mềm liên quan tới tất cả các khía cạnh khác của kỹ thuật phần mềm. Do đó, tài liệu này được liên kết tới tất cả các lĩnh vực khác trong bảo trì phần mềm.



1. Cơ bản về bảo trì phần mềm

Mục đầu tiên giới thiệu các khái niệm và thuật ngữ cái mà hình thành một nền tảng cơ bản để hiểu rõ vai trò phạm vi của bảo trì phần mềm. Các chủ đề cung cấp khái niệm và nhấn mạnh lý do tại sao phần mềm phải có nhu cầu bảo trì. Phân loại bảo trì phần mềm là rất quan trọng để hiểu được ý nghĩa của nó.

1.1 Định nghĩa và thuật ngữ

Mục đích của bảo trì phần mềm được định nghĩa theo các tiêu chuẩn quốc tế về bảo trì: trong bối cảnh của công nghệ phần mềm, bảo trì phần mềm cơ bản là một trong nhiều quy trình kỹ thuật. Mục tiêu của bảo trì phần mềm là sửa đổi phần mềm hiện có trong khi vẫn đảm bảo toàn vẹn. Các chuẩn quốc tế cũng khẳng định tầm quan trọng của việc nên có một vài hoạt động bảo trì nên xảy ra trước khi bàn giao sản phẩm.

1.2 Bản chất của bảo trì phần mềm

Bảo trì phần mềm duy trì các sản phẩm phần mềm trong suốt vòng đời của nó (từ phát triển đến hoạt động). Yêu cầu sửa đổi được đăng nhập và theo dõi, các tác động của những thay đổi được xác định, code và sản phẩm sau khi được sửa đổi, thử nghiệm được tiến hành và một phiên bản mới của các sản phẩm được phát hành. Ngoài ra, tập huấn và hỗ trợ thường xuyên sẽ được cung cấp cho người sử dụng. Những người bảo trì được định nghĩa giống như một tổ chức thực hiện các hoạt động bảo trì; đôi khi sẽ đề cập đến những cá nhân thực hiện các hoạt động khác hẳn với các nhà phát triển. IEEE 14.764 xác định các hoạt động

chính của bảo trì phần mềm giống như thực thi các quy trình, phân tích các lỗi và chỉnh sửa, thực thi chỉnh sửa, kiểm định, tích hợp hoặc loại bỏ. Các hoạt động này được thảo luận tại mục 3.2 – hoạt động bảo trì.

Bảo trì có thể học hỏi các kiến thức từ các nhà phát triển phần mềm. Liên hệ với những nhà phát triển và sự tham gia từ đầu của các nhà phát triển sẽ làm giảm nỗ lực bảo trì tổng thể. Trong một số trường hợp, các nhà phát triển ban đầu không thể hợp tác hoặc đã chuyển sang nhiệm vụ khác, vấn đề này sẽ tạo ra thách thức cho bảo trì. Bảo trì phải mất nhiều sản phẩm (artifacts) (ví dụ source code, tài liệu ...) từ các nhà phát triển hoặc phải hỗ trợ ngay lập tức, sau đây dần dần tự phát triển và duy trì chúng trong một vòng đời phần mềm.

1.3 Sự cần thiết của bảo trì phần mềm

Bảo trì là cần thiết để đảm bảo rằng phần mềm có thể tiếp tục đáp ứng yêu cầu của người sử dụng. Bảo trì được áp dụng cho phần mềm bằng cách sử dụng bất kì mô hình vòng đời phần mềm (ví dụ: xoắn ốc, tuyến tính,...). Sản phẩm phần mềm thay đổi do hoạt động chỉnh sửa sai hoặc không sai. Bảo trì phần mềm phải được thực hiện để:

- Khắc phục lỗi.
- Cải thiện thiết kế.
- Thực hiện các cải tiến.
- Giao diện với các phần mềm khác.
- Thích ứng với các loại phần cứng, phần mềm, tính năng hệ thống,...khác nhau có thể được sử dụng.
- Tiến hóa phần mềm.
- Hủy bỏ phần mềm.

5 đặc điểm chính bao gồm các hoạt động của người bảo trì bao gồm:

- Duy trì kiểm soát các chức năng của phần mềm liên tục.
- Duy trì kiểm soát việc sửa đổi phần mềm.
- Hoàn thiện các chức năng hiện có.
- Xác định các mối đe dọa an ninh và sửa chữa các lỗ hổng an ninh.
- Ngăn ngừa việc xuống cấp hiệu xuất tới mức không thể chấp nhận được.

1.4. Chi phí bảo trì

Bảo trì tiêu thụ một phần lớn các nguồn lực tài chính trong vòng đời phần mềm. Một nhận thức chung về bảo trì thường thấy: bảo trì đơn thuần là sửa lỗi. Tuy nhiên, các nghiên cứu và khảo sát trong những năm qua đã chỉ ra rằng phần lớn, trên 80%, bảo trì phần mềm được sử dụng cho các hành động khắc phục. Nhóm các cải tiến và sửa chữa lại với nhau trong báo cáo quản lý góp phần tạo ra các quan niệm sai lầm về chi phí cao của việc sửa chữa.

Hiểu biết về các loại bảo trì phần mềm giúp hiểu được cơ cấu của chi phí bảo trì. Ngoài ra, có kiến thức về các yếu tố ảnh hưởng tới bảo trì phần mềm sẽ giúp quản lý được chi phí. Một số yếu tố môi trường và mối quan hệ ảnh hưởng tới chi phí bảo trì phần mềm:

- Môi trường hoạt động liên quan đến phần cứng và phần mềm.
- Môi trường tổ chức liên quan đến chính sách, tính cạnh tranh, quy trình, sản phẩm và nhân viên.

1.5. Sự tiến hóa của phần mềm

Quá trình tiến hóa của phần mềm đã được đề cập tới lần đầu tiên vào cuối năm 1960s.

Trong khoảng thời gian hai mươi năm, nghiên cứu đưa đến việc xây dựng “Law of Evolution”. Những phát hiện chính bao gồm một đề nghị đó là duy trì phát triển, tiến hóa, và quyết định bảo dưỡng được hỗ trợ bởi sự thu thập kiến thức về hoạt động của phần mềm theo thời gian. Một số trạng thái được tiếp tục duy trì phát triển, nói cách khác, hiện tại phần mềm lớn không bao giờ là hoàn thành, và nó liên tục phát triển; nó trở nên phức tạp hơn, trừ khi một số hành động được thực hiện để giảm sự phức tạp này.

1.6. Phân loại bảo trì phần mềm

Có ba loại bảo trì được xác định: corrective (điều chỉnh, sửa chữa), adaptive (thích ứng), và perfective (hoàn thiện). Ngoài ra còn có preventative (dự phòng).

- Bảo trì sửa chữa: sửa đổi phản ứng (hoặc sửa chữa) một sản phẩm phần mềm thực hiện sau khi bàn giao để chỉnh sửa các vấn đề được phát hiện. Trong loại này, bảo trì là khẩn cấp, một thay đổi đột xuất thực hiện.
- Bảo trì thích ứng: sửa đổi một sản phẩm phần mềm được thực thi sau khi bàn giao, phần mềm muốn chuyển đổi môi trường. Ví dụ như: nâng cấp hệ điều hành, khi đây việc nâng cấp phần mềm là cần thiết.
- Bảo trì hoàn thiện: sửa đổi một sản phẩm phần mềm sau khi bàn giao để cung cấp cải tiến cho người dùng, cải thiện tài liệu chương trình, và viết lại mã để cải thiện hiệu suất phần mềm, bảo trì, hoặc các thuộc tính phần mềm khác.
- Bảo trì dự phòng: sửa đổi một sản phẩm phần mềm sau khi tìm thấy các lỗi tiềm ẩn trong sản phẩm phần mềm trước khi lỗi hoạt động.

2. Các vấn đề chính trong bảo trì phần mềm

Một số vấn đề quan trọng phải được xử lý để đảm bảo duy trì hiệu quả của phần mềm. Bảo trì phần mềm cung cấp các thách thức độc lập về kỹ thuật và quản lý cho các kỹ sư phần mềm. ví dụ: cố gắng tìm được một lỗi trong phần mềm có chứa lượng lớn các dòng code của nhà phát triển. Tương tự như vậy, cạnh tranh với những nhà phát triển giống như một

trận chiến liên tục. Lập kế hoạch cho một tương lai phát hành sản phẩm, thường bao gồm code cho tới khi tạo ra các bản vá lỗi cho phiên bản hiện tại, cũng tạo ra một thách thức. Phần sau đây trình bày một số vấn đề kỹ thuật và quản lý liên quan tới bảo trì phần mềm. Họ đã được nhóm lại theo các chủ đề sau:

- Vấn đề kĩ thuật
- Vấn đề quản lý
- Ước lượng chi phí
- Đo lường

2.1 Vấn đề kĩ thuật

2.1.1 Giới hạn về hiểu biết

Hiểu biết hạn chế, đề cập đến một cách nhanh chóng, kĩ sư phần mềm có thể hiểu được để có thể thay đổi hoặc sửa chữa lỗi trong phần mềm, cái mà họ phát triển. Nghiên cứu chỉ ra rằng, khoảng một nửa tổng nỗ lực được duy trì dành hết cho tìm hiểu phần mềm phải sửa đổi. Như vậy, chủ đề này là mối quan tâm lớn cho các kĩ sư phần mềm. Vì vậy, thời điểm đầu, kĩ sư phần mềm thường có hiểu biết tương đối hạn chế về phần mềm.

2.1.2 Kiểm thử

Chi phí để lặp đi lặp lại kiểm tra phần mềm chiếm phần lớn thời gian và tiền bạc. Để đảm bảo rằng, các yêu cầu báo cáo vấn đề là hợp lệ, những nhà bảo trì nên tái tạo hoặc xác minh vấn đề bằng cách chạy thử nghiệm. Kiểm tra hồi quy (các lần kiểm tra lại để xác minh sửa đổi không gây ra tác dụng không mong muốn) là một khái niệm kiểm thử quan trọng trong bảo trì. Ngoài ra, thời gian tìm kiếm để kiểm tra là một vấn đề rất khó khăn. Phối hợp kiểm thử khi các thành viên trong đội bảo trì đang làm các công việc khác nhau vẫn còn là một thách thức. Khi phần mềm thực hiện các chức năng quan trọng, nó khó có thể offline để thực hiện kiểm thử. Kiểm thử không thể thực hiện trong môi trường có ý nghĩa sản xuất nhất định. Kiến thức kiểm thử cung cấp thêm thông tin và tài liệu tham khảo về vấn đề này trong chủ đề phụ của nó dựa trên kiểm thử hồi quy.

2.1.3 Phân tích tác động

Phân tích tác động mô tả cách tiến hành, chi phí, một cách phân tích đầy đủ về tác động của sự thay đổi trong phần mềm hiện có. Bảo trì phải có một kiến thức sâu sắc và nội dung của phần mềm. Họ sử dụng kiến thức đó để thực hiện các phân tích tác động, trong đó xác định tất cả các hệ thống và sản phẩm phần mềm bị ảnh hưởng bởi một yêu cầu thay đổi phần mềm và ước tính chi phí phát triển và các nguồn lực cần thiết để thực hiện các thay đổi. Ngoài ra, nguy cơ làm thay đổi được các quyết định. Các yêu cầu thay đổi, đôi khi được

gọi là một yêu cầu sửa đổi (MR) và thường gọi là một bản báo cáo vấn đề (PR), trước tiên phải được phân tích và dịch thành ngôn ngữ phần mềm. Phân tích tác động được thực hiện sau khi một yêu cầu thay đổi đưa vào quy trình quản lý cầu hình. IEEE 14.764 nêu các nhiệm vụ phân tích tác động:

- Phân tích MRs/ PRs.
- Tái tạo hay xác minh các vấn đề.
- Phát triển các tùy chọn để thực hiện sửa đổi.
- Tài liệu hóa MR/PR, kết quả và thực hiện các tùy chọn.
- Chấp thuận cho sửa đổi các tùy chọn. Mức độ nghiêm trọng của vấn đề thường được sử dụng để quyết định làm thế nào và khi nào án định. Các kỹ sư phần mềm sau đó xác định các thành phần bị ảnh hưởng. Một số giải pháp tiềm năng được cung cấp, tiếp theo đó là một khuyến cáo tốt nhất đưa ra. Phần mềm được thiết kế với khả năng bảo trì trong điều kiện phân tích rất nhiều tác động. Thông tin chi tiết có thể được tìm thấy trong cuốn Software Configuration Management KA.

2.1.4 Bảo trì

IEEE 14.764 định nghĩa bảo trì như là khả năng của các sản phẩm phần mềm có thể sửa đổi. Sửa đổi có thể bao gồm sửa chữa, cải tiến, hoặc áp dụng các phần mềm sao cho phù hợp với các môi trường và chi tiết kỹ thuật chức năng theo yêu cầu. Là một đặc trưng chính trong chất lượng phần mềm, bảo trì nên được chỉ định, xem xét lại, và kiểm soát các hoạt động trong quá trình phát triển phần mềm để giảm chi phí bảo trì. Khi thực hiện thành công, bảo trì phần mềm sẽ được cải thiện. Bảo trì thường khó đạt được vì nó không phải là trọng tâm quan trọng nhất trong quá trình phát triển phần mềm. Các nhà phát triển thông thường sẽ bận rộn với nhiều hoạt động khác và thường xuyên bỏ qua yêu cầu của các bảo trì viên. Điều này thường xuyên xảy ra dẫn đến thiếu tài liệu phần mềm, và môi trường kiểm tra, đó là nguyên nhân hàng đầu của những khó khăn trong đọc hiểu chương trình và phân tích tác động. Sự hiện diện của các quy trình, kỹ thuật, và các công cụ hệ thống sẽ giúp tăng cường khả năng bảo trì phần mềm.

2.2 Vấn đề về quản lý.

2.2.1 Liên kết với các mục tiêu của tổ chức.

Mục tiêu của tổ chức mô tả là làm thế nào để chứng minh được tại sao phải có các hoạt động đầu tư vào bảo trì phần mềm. Phát triển phần mềm ban đầu là thường dựa trên dự án, với quy mô, thời gian và ngân sách xác định. Sự nhấn mạnh chính là cung cấp một sản phẩm đáp ứng nhu cầu của người sử dụng phù hợp về thời gian và ngân sách. Ngược lại, bảo trì phần mềm thường có các mục tiêu là mở rộng vòng đời sống của các phần mềm trở

nên càng lâu càng tốt. Ngoài ra, nó có thể được điều khiển bởi sự cần thiết để đáp ứng nhu cầu của người sử dụng khi cập nhật phần mềm và cải tiến. Trong cả 2 trường hợp trên, sự trở lại trên đầu tư sẽ ít nhiều rõ ràng.

2.2.2 Nhân viên

Nhân sự đề cập tới làm thế nào để thu hút và giữ chân các nhân viên bảo trì phần mềm. Bảo trì không phải là thường được xem như một công việc quyền rũ. Như điều tra, nhân viên bảo trì phần mềm chỉ được xem như “công nhân hạng hai” và do đó giảm sút tinh thần.

2.2.3 Quy trình

Các quy trình vòng đời của phần mềm là một tập hợp các hoạt động, phương pháp, cách thực thi và biến đổi, cái mà người sử dụng để phát triển và duy trì phần mềm và các hoạt động liên quan. Ở cấp độ quy trình, phần mềm và các hoạt động bảo trì chia sẻ nhiều điểm chung với việc phát triển (ví dụ, quản lý cấu hình phần mềm là một hoạt động rất quan trọng trong cả 2 hình thức). Bảo trì cũng đòi hỏi một số hoạt động mà không được tìm thấy trong phát triển phần mềm, những hoạt động này là thách thức đối với thực tại quản lý.

2.2.4 Các khía cạnh về tổ chức bảo trì

Khía cạnh tổ chức mô tả làm thế nào để xác định được tổ chức và chức năng sẽ có trách nhiệm bảo trì phần mềm. Các nhóm phát triển phần mềm không nhất thiết phải được giao bảo trì phần mềm khi nó hoạt động. Trong quyết định đâu là nơi mà phần mềm bảo trì, chức năng này sẽ được đặt, ví dụ với phát triển ban đầu đi đến một đội ngũ bảo trì vĩnh viễn (hoặc duy trì), có một đội ngũ bảo trì vĩnh viễn với nhiều lợi ích:

- Cho phép chuyên môn.
- Tạo ra các kênh thông tin liên lạc.
- Thúc đẩy không gian làm việc.
- Giảm sự phụ thuộc vào các cá nhân.
- Cho phép kiểm tra định kỳ. Vì có rất nhiều ưu điểm và khuyết điểm để lựa chọn, quyết định phải được thực hiện trên từng trường hợp. Phân công trách nhiệm bảo trì là một điều quan trọng, bất kể trong một nhóm đơn lẻ, hoặc trong cơ cấu tổ chức.

2.2.5 Gia công phần mềm

Gia công phần mềm ra nước ngoài để bảo trì đã trở thành một ngành công nghiệp lớn. Các tổ chức đang gia công phần mềm nằm trong danh mục của các nhà đầu tư. Tuy nhiên, thường tùy chọn gia công phần mềm rất ít là phần việc quan trọng vì các công ty không muốn mất kiểm soát cốt lõi kinh doanh của họ trong sản phẩm phần mềm. Một trong những

thách thức lớn của gia công phần mềm chính là xác định phạm vi của các dịch vụ bảo dưỡng, các điều khoản thuận cắp độ dịch vụ và chi tiết trong hợp đồng. Các công ty gia công phải đầu tư cơ sở hạ tầng, và thiết lập hỗ trợ từ xa và phải có những người nói ngôn ngữ bản địa. Gia công phần mềm đòi hỏi một sự đầu tư ban đầu tốt, và thiết lập quy trình bảo trì.

2.3 Ước tính chi phí bảo trì.

Các kỹ sư phần mềm phải hiểu các dạng thức khác nhau của bảo trì phần mềm, thảo luận ở trên, để giải quyết vấn đề ước tính chi phí bảo trì phần mềm. Đối với mục đích lập kế hoạch, dự toán chi phí là một khía cạnh quan trọng của kế hoạch để bảo trì phần mềm.

2.3.1 Ước tính chi phí

Mục 2.1.3 mô tả cách phân tích tác động, xác định tất cả các hệ thống và các sản phẩm phần mềm bị ảnh hưởng bởi một yêu cầu thay đổi phần mềm và ước tính nguồn lực cần thiết để phát triển và thực hiện thay đổi. Dự toán chi phí bảo trì được ảnh hưởng bởi nhiều yếu tố kỹ thuật và không kỹ thuật. IEEE 14.764 cho rằng “hai phương pháp phổ biến nhất để tiếp cận ước tính chi phí và nguồn lực cho bảo trì phần mềm là việc sử dụng các mô hình và sử dụng kinh nghiệm. Một sự kết hợp của hai phương pháp này cũng được sử dụng.

2.3.2 Mô hình tham biến

Mô hình tham biến chi phí (mô hình toán học) đã được áp dụng để bảo trì phần mềm. Có ý nghĩa là dữ liệu lịch sử từ bảo trì đã qua là cần thiết để sử dụng và hiệu chuẩn các mô hình toán học. Thuộc tính điều khiển chi phí ảnh hưởng tới dự toán.

2.3.3 Kinh nghiệm

Kinh nghiệm, trong các hình thức của sự phỏng đoán từ chuyên gia, thường được sử dụng để ước tính chi phí trong nỗ lực bảo trì. Rõ ràng, cách tiếp cận tốt nhất để ước lượng bảo trì là kết hợp các dữ liệu lịch sử và kinh nghiệm. Chi phí để tiến hành sửa đổi (về số lượng của con người và số lượng thời gian). Bảo trì ước lượng dữ liệu lịch sử cần được cung cấp như là một kết quả của phép đo chương trình.

2.4 Đo lường bảo trì phần mềm.

Các đối tượng liên quan đến bảo trì phần mềm, mà thuộc tính có thể bị đo lường bao gồm: các quá trình, tài nguyên và các sản phẩm. Có một số biện pháp phần mềm mà có thể được bắt nguồn từ các thuộc tính của phần mềm, quy trình bảo trì, và nhân viên, bao gồm kích thước, độ phức tạp, chất lượng, tính dễ hiểu, khả năng bảo trì, và công sức (effort). Giải

pháp phức tạp của phần mềm cũng có thể thu được bằng cách sử dụng các công cụ thương mại sẵn có. Những biện pháp này tạo thành một điểm khởi đầu tốt cho chương trình đo lường của người bảo trì. Thảo luận về các quy trình phần mềm và đo lường sản phẩm cũng được thể hiện trong tài liệu Software Engineering Process KA. Chủ đề về chương trình đo lường phần mềm được mô tả trong tài liệu Software Engineering Management KA.

2.4.1 Các biện pháp cụ thể

Các nhà bảo trì phải xác định các biện pháp thích hợp cho một tổ chức dựa trên hoàn cảnh riêng của tổ chức đó. Chất lượng phần mềm cho thấy các biện pháp cụ thể để bảo trì. Các biện pháp phụ bao gồm những điều sau đây:

- Phân tích: công sức của các kiểm thử viên hay nguồn lực hoặc để chẩn đoán thiếu sót hoặc nguyên nhân thất bại hoặc để xác định các thành phần phải sửa đổi.
- Tách hay thay đổi: các biện pháp của một kiểm thử viên (measures of the maintainer's effort) kết hợp với việc thực hiện một quy định sửa đổi.
- Tính ổn định: các biện pháp của các hành vi không mong muốn, trong đó gặp phải tại quá trình kiểm thử.
- Khả năng kiểm thử: các biện pháp của các kiểm thử viên và nỗ lực của người sử dụng trong việc cố gắng kiểm tra sửa đổi.
- Các biện pháp khác nhau mà bảo trì sử dụng bao gồm: kích thước của phần mềm, độ phức tạp, tính dễ hiểu và khả năng bảo trì.

Cung cấp các nỗ lực bảo trì phần mềm, bởi chủng loại, cho các loại ứng dụng khác nhau cung cấp thông tin kinh doanh cho người dùng và các tổ chức của họ. Nó cũng cho phép so sánh các hồ sơ bảo trì phần mềm nội bộ trong một cơ quan, tổ chức.

3. Quy trình bảo trì phần mềm

Ngoài các quy trình công nghệ phần mềm tiêu chuẩn và hoạt động được mô tả trong IEEE 14.764, có một số hoạt động mà là duy nhất để bảo trì.

3.1. Các bước trong bảo trì phần mềm

Quy trình bảo trì cung cấp các hoạt động cần thiết và chi tiết đầu vào / đầu ra cho những hoạt động như mô tả trong IEEE 14764. Các hoạt động quy trình bảo trì của IEEE 14.764 được thể hiện trong hình 5.2. Các hoạt động bảo trì phần mềm bao gồm:

- Việc thực hiện quy trình,
- Vấn đề và phân tích sửa đổi,
- Thực hiện sửa đổi,

- Xem xét bảo trì / nghiệm thu,
- Tiến hóa phần mềm
- Nghỉ hưu phần mềm

Gần đây, phương pháp nhanh nhẹn, nhằm thúc đẩy quá trình ánh sáng, đã được cung thích nghi với bảo trì. Yêu cầu này xuất phát từ nhu cầu ngày càng tăng cho quay vòng nhanh các dịch vụ bảo trì. Cải tiến quy trình bảo trì phần mềm được hỗ trợ bởi các mô hình trưởng thành năng lực bảo trì phần mềm chuyên dụng

3.2. Các hoạt động trong bảo trì, bảo dưỡng phần mềm

Quá trình bảo dưỡng gồm các hoạt động và nhiệm vụ cần thiết để sửa đổi một sản phẩm phần mềm hiện có trong khi vẫn giữ toàn vẹn của nó. Những hoạt động và nhiệm vụ là trách nhiệm của các nhà bảo trì. Như đã lưu ý, nhiều hoạt động bảo dưỡng tương tự như phát triển phần mềm. Bảo trì thực hiện phân tích, thiết kế, mã hóa, thử nghiệm, và tài liệu. Họ phải theo dõi các yêu cầu trong hoạt động của họ, giống như được thực hiện trong tài liệu hướng dẫn phát triển và cập nhật là đường cơ sở thay đổi. IEEE 14.764 khuyến cáo rằng khi một nhà bảo trì sử dụng một quá trình phát triển, nó phải được thay đổi để đáp ứng các nhu cầu cụ thể. Tuy nhiên, để bảo trì phần mềm, một số hoạt động liên quan đến quy trình độc đáo để bảo trì phần mềm.

3.2.1. Các hoạt động độc lập

Có một số quy trình, hoạt động, và thực hành là duy nhất để bảo trì phần mềm:

- Hiểu biết chương trình: các hoạt động cần thiết để có được một kiến thức tổng quát về những gì một sản phẩm phần mềm nào và làm thế nào các bộ phận làm việc cùng nhau.
- Chuyển tiếp: một chuỗi kiểm soát và điều phối các hoạt động trong đó phần mềm được chuyển dần từ các nhà phát triển cho nhà bảo trì.
- Sửa đổi yêu cầu chấp nhận / từ chối: sửa đổi yêu cầu công việc vượt quá một kích thước nhất định / nỗ lực / phức tạp có thể bị từ chối bởi người duy trì và định tuyến lại đến một nhà phát triển.
- Bảo trì giúp đỡ phối hợp: một người dùng cuối và bảo trì phối hợp hỗ trợ chức năng mà gây ra sự đánh giá, ưu tiên, và chi phí của các yêu cầu sửa đổi.
- Phân tích tác động: một kỹ thuật để xác định khu vực bị ảnh hưởng bởi sự thay đổi tiềm năng;
- Hiệp định mức độ bảo trì dịch vụ và giấy phép bảo dưỡng và các hợp đồng: thỏa thuận hợp đồng mô tả các dịch vụ và mục tiêu chất lượng.

3.2.2. Hoạt động hỗ trợ

Bảo trì cũng có thể thực hiện các hoạt động hỗ trợ, chẳng hạn như tài liệu hướng dẫn, quản lý cấu hình phần mềm, xác minh và xác nhận, giải quyết vấn đề, phần mềm đảm bảo chất lượng, đánh giá và kiểm toán. Một hoạt động hỗ trợ quan trọng bao gồm đào tạo các nhà bảo trì và người sử dụng.

3.2.3. Các hoạt động lập kế hoạch bảo trì phần mềm

Một hoạt động quan trọng để bảo trì phần mềm đang có kế hoạch, và bảo trì phải giải quyết các vấn đề liên quan đến một số quan điểm quy hoạch, bao gồm cả:

- Lập kế hoạch kinh doanh (cấp độ tổ chức),
- Lập kế hoạch bảo dưỡng (cấp độ chuyển tiếp),
- Lập kế hoạch phát hành / phiên bản (cấp độ phần mềm)
- Lập kế hoạch yêu cầu thay đổi phần mềm cá nhân (cấp độ yêu cầu) Ở cấp độ yêu cầu cá nhân, kế hoạch được thực hiện trong quá trình phân tích tác động (xem phần 2.1.3, Phân tích tác động). Các hoạt động lập kế hoạch phát hành / phiên bản yêu cầu các nhà bảo trì:
 - Thu thập các ngày khả dụng của các yêu cầu cá nhân,
 - Đồng ý với người dùng trên các nội dung của bản phát hành/ phiên bản kế tiếp,
 - Xác định các xung đột tiềm năng và phát triển các giải pháp thay thế,
 - Đánh giá nguy cơ của một thông cáo được đưa ra và phát triển một kế hoạch dự phòng trong trường hợp các vấn đề sẽ phát sinh,
 - Thông báo cho tất cả các bên liên quan

Trong khi các dự án phát triển phần mềm thường có thể kéo dài từ vài tháng đến vài năm, giai đoạn bảo trì thường kéo dài trong nhiều năm. Lập dự toán các nguồn lực là một yếu tố quan trọng của việc lập kế hoạch bảo trì. Lập kế hoạch bảo trì phần mềm nên bắt đầu với quyết định để phát triển một sản phẩm phần mềm mới và nên xem xét các mục tiêu chất lượng. Một tài liệu khái niệm cần được phát triển, sau là một kế hoạch bảo trì. Khái niệm bảo trì cho mỗi sản phẩm phần mềm cần phải được ghi trong kế hoạch và cần giải quyết

- Phạm vi của bảo trì phần mềm
- Thích ứng của quá trình bảo trì phần mềm
- Xác định các tổ chức bảo trì phần mềm
- Ước tính chi phí bảo trì phần mềm.

Bước tiếp theo là phát triển một kế hoạch bảo trì phần mềm tương ứng. Kế hoạch này cần được chuẩn bị trong thời gian phát triển phần mềm và nên xác định cách người dùng sẽ yêu cầu sửa đổi phần mềm hoặc báo cáo các vấn đề. Lập kế hoạch bảo trì phần mềm được đề cập trong IEEE 14764. Nó cung cấp hướng dẫn cho một kế hoạch bảo trì. Cuối cùng, ở cấp

cao nhất, các tổ chức bảo trì sẽ phải tiến hành các hoạt động lập kế hoạch kinh doanh (ngân sách, và nguồn nhân lực tài chính) giống như tất cả các bộ phận khác của tổ chức. Quản lý được thảo luận trong các chương nguyên tắc liên quan của công nghệ phần mềm.

3.2.4. Quản lý cấu hình phần mềm

IEEE 14.764 mô tả quản lý cấu hình phần mềm như là một yếu tố quan trọng của quá trình bảo trì. Các thủ tục quản lý cấu hình phần mềm sẽ cung cấp cho việc xác minh, xác nhận, và kiểm toán của mỗi bước cần thiết để xác định, ủy quyền, thực hiện và phát hành các sản phẩm phần mềm. Nó không phải là chỉ đơn giản là đủ để theo dõi các yêu cầu sửa đổi hoặc vấn đề báo cáo. Các sản phẩm phần mềm và bất kỳ thay đổi được thực hiện cho nó phải được kiểm soát. Điều khiển này được thiết lập bằng cách thực hiện và thực thi một quá trình quản lý cấu hình phần mềm đã được phê duyệt (SCM). Quản lý cấu hình phần mềm KA cung cấp thông tin chi tiết của SCM và thảo luận về quá trình mà các yêu cầu thay đổi phần mềm trình, đánh giá và phê duyệt. SCM để bảo trì phần mềm là khác nhau từ SCM cho phát triển phần mềm trong số thay đổi nhỏ mà phải được kiểm soát trên phần mềm hoạt động. Quá trình SCM được thực hiện bằng cách phát triển và theo một kế hoạch quản lý cấu hình phần mềm và quy trình hoạt động. Bảo trì tham gia Ban kiểm soát cấu hình để xác định nội dung của việc phát hành / phiên bản tiếp theo.

3.2.5. Chất lượng phần mềm

Nó không phải là chỉ đơn giản là đủ để hy vọng rằng việc gia tăng chất lượng sẽ cho kết quả từ việc bảo trì phần mềm. Bảo trì cần phải có một chương trình chất lượng phần mềm. Nó phải được quy hoạch và quy trình phải được thực hiện để hỗ trợ quá trình bảo trì. Các hoạt động và kỹ thuật để đảm bảo chất lượng phần mềm (SQA), V & V, đánh giá, và kiểm toán phải được lựa chọn trong sự phối hợp với tất cả các quá trình khác để đạt được mức mong muốn về chất lượng. Nó cũng khuyến cáo rằng các nhà bảo trì thích ứng với quá trình phát triển phần mềm, kỹ thuật và phân phôi (ví dụ, tài liệu thử nghiệm), và kết quả xét nghiệm. Thông tin chi tiết có thể được tìm thấy trong các phần mềm chất lượng KA.

4. Các kỹ thuật trong bảo trì phần mềm

Chủ đề này giới thiệu một số kỹ thuật thường được chấp nhận sử dụng trong bảo trì phần mềm.

4.1. Đọc hiểu chương trình

Các lập trình viên dành nhiều thời gian đọc sách và các chương trình sự hiểu biết đáng kể để thực hiện thay đổi. Trình duyệt mã là công cụ quan trọng cho chương trình hiểu và được sử dụng để tổ chức và mã nguồn hiện nay. Tài liệu rõ ràng và súc tích cũng có thể hỗ trợ trong chương trình hiểu.

4.2. Tái cấu trúc

Tái cấu trúc được định nghĩa là việc kiểm tra và thay đổi phần mềm để pha nó trong một hình thức mới, và bao gồm việc thực hiện tiếp theo của hình thức mới. Nó thường không được thực hiện để cải thiện khả năng bảo trì nhưng để thay thế lão hóa phần mềm kế thừa. Tái cấu trúc là một kỹ thuật tái cấu trúc đó nhằm tổ chức lại một chương trình mà không cần thay đổi hành vi của nó. Nó tìm cách cải thiện cấu trúc chương trình và bảo trì của nó. Kỹ thuật refactoring có thể được sử dụng trong quá trình thay đổi nhỏ.

4.3. Kỹ thuật đảo ngược

Kỹ thuật đảo ngược là quá trình phân tích phần mềm để xác định thành phần của phần mềm và liên mối quan hệ của họ và tạo ra các cơ quan đại diện của các phần mềm dưới hình thức khác hoặc ở mức độ trừu tượng cao hơn. Kỹ thuật đảo ngược là thụ động; nó không thay đổi phần mềm hoặc kết quả trong phần mềm mới. Nỗ lực đảo ngược kỹ thuật sản xuất đồ thị cuộc gọi và đồ thị luồng điều khiển từ mã nguồn. Một loại kỹ thuật đảo ngược là tài liệu hóa lại. Một loại khác là phục hồi thiết kế. Cuối cùng, dữ liệu kỹ thuật đảo ngược, nơi schemas logic được thu hồi từ cơ sở dữ liệu vật lý, đã trở nên quan trọng trong vài năm qua. Công cụ là chìa khóa cho kỹ thuật và nhiệm vụ liên quan ngược như tài liệu hóa lại và phục hồi thiết kế.

4.4. Chuyển đổi và tích hợp

Trong suốt cuộc đời của phần mềm, nó có thể phải được sửa đổi để chạy trong môi trường khác nhau. Để di chuyển nó đến một môi trường mới, các nhà bảo trì cần xác định các hành động cần thiết để thực hiện việc chuyển đổi, và sau đó phát triển và tài liệu hóa các bước cần thiết để thực hiện việc di chuyển trong một kế hoạch di cư bao gồm các yêu cầu chuyển đổi, công cụ chuyển đổi, chuyển đổi của sản phẩm và dữ liệu, thực hiện, xác minh, và hỗ trợ. Phần mềm di cư cũng có thể kéo theo một số hoạt động khác như:

- Thông báo về mục định: một tuyên bố về lý do tại sao môi trường cũ không còn được hỗ trợ, theo sau là một mô tả của môi trường mới và ngày sẵn có.
- Các hoạt động song song: làm cho có sẵn các môi trường cũ và mới để người dùng trải nghiệm một chuyển đổi suôn sẻ với môi trường mới.
- Thông báo hoàn thành: khi di chuyển dự kiến được hoàn tất, một thông báo được gửi đến tất cả có liên quan.

- Xem xét sau phẫu thuật: đánh giá về hoạt động song song và tác động của thay đổi môi trường mới.
- Dữ liệu lưu trữ: lưu trữ các dữ liệu phần mềm cũ.

4.5. Nghỉ hưu phần mềm

Một khi phần mềm đã đạt đến kết thúc của cuộc sống hữu ích của nó, nó phải được nghỉ hưu. Một phân tích nên được thực hiện để hỗ trợ trong việc đưa ra các quyết định nghỉ hưu. Phân tích này nên được bao gồm trong kế hoạch nghỉ hưu, trong đó bao gồm các yêu cầu về hưu, tác động, thay thế, lịch trình, và nỗ lực. Tiếp cận của các kho lưu trữ các bản sao của dữ liệu cũng có thể được bao gồm. Nghỉ hưu phần mềm đòi hỏi một số hoạt động tương tự như di cư.

5. Công cụ sử dụng trong bảo trì phần mềm

Chủ đề này bao gồm các công cụ đặc biệt quan trọng trong việc bảo trì phần mềm, nơi phần mềm hiện đang được sửa đổi. Các ví dụ về chương trình hiểu bao gồm:

- Cắt chương trình, mà chỉ chọn các phần của một chương trình bị ảnh hưởng bởi sự thay đổi;
- Phân tích tĩnh, cho phép xem chung và bắn tóm tắt của một nội dung chương trình;
- Phân tích động, cho phép các nhà bảo trì dễ theo dõi con đường thực hiện của một chương trình;
- Phân tích lưu lượng dữ liệu, cho phép các nhà bảo trì dễ theo dõi tất cả các luồng dữ liệu có thể có của một chương trình;
- Tham chiếu xuyên qua, tạo ra các chỉ số thành phần của chương trình;
- Phân tích phụ thuộc, giúp bảo trì phân tích và hiểu được mối quan hệ giữa các thành phần của một chương trình.

Các công cụ kỹ thuật đảo ngược hỗ trợ quá trình này bằng cách làm ngược từ một sản phẩm hiện có để tạo ra các đồ tạo tác như đặc điểm kỹ thuật và thiết kế giới thiệu, sau đó có thể được biến đổi để tạo ra một sản phẩm mới từ một tuổi. Bảo trì cũng sử dụng phần mềm kiểm tra, quản lý cấu hình phần mềm, tài liệu phần mềm, và các công cụ đo lường phần mềm.

Chapter 6:Software Configuration Management

Nhóm 6 :

Nguyễn Văn Hải

Nguyễn Thành Luân

Introduction

Chắc hẳn trong chúng ta đã ít nhất một lần gặp những tình huống như :

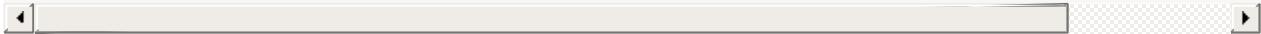
- Một lỗi (bug) nào đó của phần mềm đang xây dựng đã tốn nhiều công sức sửa chữa,bỗng dưng xuất hiện trở lại.
- Một chức năng (function) nào đó của phần mềm đã được phát triển và kiểm tra cẩn thận bỗng thất lạc hoặc biến mất một cách khó hiểu.
- Một chương trình (program) đã được kiểm tra hết sức cẩn thận bỗng nhiên không chạy được nữa.
- Hay một chương trình gồm nhiều đơn thể (module),mà mỗi đơn thể đó là một chức năng và các chức năng lại được chia cho nhiều lập trình viên với mỗi chức năng bao gồm nhiều tập tin mã nguồn (source code) cùng với nhiều phiên bản khác nhau.Khi tích hợp hệ thống và biên dịch,trong hàng chục tập tin source code với hàng trăm version.Làm cách nào để xác định tập tin nào hay version nào là đúng cần phải lấy để tiến hành tích hợp?

Các vấn đề trên sẽ không xảy ra nếu như trong dự án việc **Quản lý cấu hình (QLCH)** được thực hiện nghiêm túc và kiểm soát chặt chẽ. Nói cách khác, **QLCH** bao gồm các công việc về nhận dạng,tổ chức và quản lý các thay đổi đối với những sản phẩm đang được xây dựng bởi một nhóm lập trình viên,từ các sản phẩm trung gian đến các sản phẩm sau cùng.

QLCH tốt sẽ giải quyết được hàng loạt những khó khăn trong các dự án,đặc biệt là các dự án lớn :

Cập nhật đồng thời : khi 2 hay nhiều lập trình viên làm việc cách biệt nhau nhưng trên cùng một chương trình hoặc dự án,những thay đổi mà người này thực hiện có thể sẽ phá vỡ kết quả làm việc của người khác.

Ví dụ: Sản phẩm anh A sử dụng kết quả của anh B, khi sản phẩm của anh B thay đổi có thể sản phẩm của anh A chạy lỗi hoặc không chạy được.



Chia sẻ source code : Trong các hệ thống lớn, khi các chức năng chung bị thay đổi tất cả những người liên quan cần được biết về sự thay đổi đó.

Phiên bản phần mềm(release) : Hầu hết các phiên chương trình hoặc hệ thống lớn được phát triển với nhiều phiên bản release tiến hóa từ thấp đến cao.

Có 2 khái niệm rất cơ bản trong QLCH:

- Configuration Item - CI: định danh này trong QLCH là tên gọi của các sản phẩm, sản phẩm trung gian, một tập tin (file) hoặc nhóm file, tài liệu hoặc nhóm tài liệu trong một dự án mà ta cần phải quản lý và kiểm soát.

Ví dụ: Một file source code, tài liệu về yêu cầu sản phẩm, bản thiết kế v.v.

- Baseline: một điểm “mốc” được thỏa thuận bởi những người liên quan trong một dự án, sao cho sau điểm “mốc” này, mọi thay đổi phải được thông báo tới tất cả những người có liên quan.

Lập kế hoạch QLCH (Configuration Management planning)

Thông thường, việc lập kế hoạch cho QLCH được thể hiện trong tài liệu có tên Kế hoạch quản lý cấu hình (Configuration Management Plan – CMP). Bản kế hoạch này thường bao gồm:

- Ý nghĩa, mục đích và phạm vi áp dụng của bản kế hoạch.
- Vai trò và trách nhiệm của nhóm, cá nhân trong dự án thực hiện các hoạt động khác nhau liên quan đến QLCH.
- Công cụ (tool), môi trường (environment) và cơ sở hạ tầng (infrastructure).
- Phương pháp định danh và thiết lập baseline trên các CI.
- Quy trình xử lý và quản lý các thay đổi (change control process).
- Kiểm kê và báo cáo cấu hình (configuration accounting and reporting).
- Quy trình thủ tục lưu trữ và chép dự phòng (backup and archive).

Định danh/đánh số các CI (Identification of Configuration Items)

Định danh là một trong những hoạt động nền tảng của QLCH. Mục đích của định danh là để xác định tính duy nhất của một CI, cũng như mối quan hệ của nó với các CI khác. Nó bao gồm việc mô tả tên, đánh số, đánh dấu đặc trưng, giúp nhận biết và phân biệt một CI với các CI hay thành phần khác.

Bạn có thể nhận thấy hình thức định danh tương tự trong đời sống thực tế. Ví dụ, người ta đánh số bàn trong nhà hàng nhằm giúp người phục vụ mang đúng thức ăn cho khách.

Mỗi CI phải có một số định danh duy nhất, dạng thức thường thấy là:

```
Ví dụ : PRJ001_REQB_1.0.4_draft_B
Số ID của dự án: PRJ001
Số ID của Item : REQB
Phiên bản: 1.0.4_draft_B
```

Trong một dự án, thường có rất nhiều file source code, quy tắc cơ bản là: các file cùng tạo nên một khối chức năng được gom chung thành một CI.

Kiểm soát phiên bản (Version Control)

Version control là sự kiểm soát các phiên bản (version) khác nhau của một CI (bao gồm việc định danh và sự lưu trữ CI đó).

Hiện có nhiều công cụ trên thị trường hỗ trợ cho việc kiểm soát phiên bản, một số công cụ thông dụng là: Visual Source Safe của Microsoft, ClearCase của Rational, CVS (nguồn mở).

Mỗi phiên bản sẽ có một số ID đầy đủ, và được tăng dần cho mỗi phiên bản mới.

Quản lý baseline (Baseline Management)

Cũng như Version Control, baseline có nhiều cách hiểu khác nhau, Trong thực tế thường gặp các loại baseline sau:

- Chức năng (functional)
- Kế hoạch (planning)
- Yêu cầu (requirements)
- Sản phẩm (product)
- Bản phân phối (release)
- Kiểm tra (test)
- Môi trường hoạt động (environment)

Quản lý baseline bao gồm:

- Chọn các CI cho mỗi loại baseline

- Tiến hành cố định baseline tại thời điểm sau khi các thay đổi đã được chấp thuận và phê chuẩn.

Thông thường, baseline được tiến hành tại điểm kết thúc của mỗi giai đoạn hay tại các “mốc” quan trọng trong dự án.

Kiểm soát thay đổi (Change control)

Khi phát triển hoặc bảo trì một sản phẩm phần mềm, việc thay đổi yêu cầu là không thể tránh khỏi. Mục đích của change control là để kiểm soát đầy đủ tất cả các thay đổi ảnh hưởng đến việc phát triển một sản phẩm. Đôi lúc chỉ một vài yêu cầu thay đổi nhỏ của khách hàng, tất cả các chặng của quy trình phát triển phần mềm từ thiết kế, đến viết code, đến kiểm tra sản phẩm đều phải thay đổi theo.

Nếu các thay đổi này không được kiểm soát chặt chẽ sẽ dẫn đến rất nhiều sai sót.

Ví dụ: 5 lập trình viên cùng làm trong một dự án, nhưng chỉ có 3 được thông báo về vi

Yêu cầu trong kiểm soát thay đổi là mọi sự thay đổi phải được thông báo đến tất cả những người hoặc nhóm làm việc có liên quan.

Các bước cơ bản của kiểm soát thay đổi bao gồm:

- Nghiên cứu, phân tích
- Phê chuẩn hoặc không phê chuẩn
- Thực hiện việc thay đổi
- Kiểm tra việc thay đổi
- Xác lập baseline mới

Trong kiểm soát thay đổi, ta cũng thường gặp khái niệm “nhóm kiểm soát thay đổi” gọi tắt là **CCB (Change Control Board)**, nhóm này được thành lập trong từng dự án. CCB thông thường bao gồm:

- Người QLCH (Configuration Manager)
- Trưởng dự án (Project Manager)
- Trưởng nhóm (Technical Lead)
- Trưởng nhóm kiểm soát chất lượng (Test Lead)
- Kỹ sư chất lượng (Quality Engineer)

Nhiệm vụ của CCB thường là:

- Bảo đảm tất cả các thay đổi được các bộ phận liên quan nhận biết và tham gia
- Xem xét, phê chuẩn hoặc phủ quyết các thay đổi trên các baseline.

- Kiểm tra, xác nhận các thay đổi
- Phê chuẩn các bản phân phối sản phẩm (release) đến khách hàng

Báo cáo tình trạng cấu hình (Configuration Status Accounting)

Công việc này bao gồm việc ghi nhận và báo cáo tình trạng của các CI cũng như yêu cầu thay đổi, tập hợp số liệu thống kê về CI, đặc biệt là các CI góp phần tạo nên sản phẩm.

Kết quả của công việc này được ghi nhận trong một báo cáo mang tên Configuration Status Accounting Report (CSAR). Báo cáo này thường làm rõ những điểm sau:

- Liệt kê tất cả baseline và CI thành phần hoặc có liên quan.
- Làm nổi bật các CI đang được phát triển hoặc vừa bị thay đổi
- Liệt kê các thay đổi còn dang dở hay đang hoàn thành, và các baseline bị ảnh hưởng (bởi sự thay đổi đó)

Báo cáo này được làm thường xuyên và định kỳ, xuyên suốt dự án.

Auditing

Có 3 loại audit thường được thực hiện :

- **CSAR Audit:** Thường được làm sau mỗi lần một CSAR được tạo ra, việc kiểm tra bao gồm
 - Bảo đảm các baseline mới nhất được liệt kê trong CSAR
 - Bảo đảm tất cả CI tạo nên một baseline được liệt kê
 - Kiểm tra các CI đã bị thay đổi từ lần baseline trước đó, so sánh chúng với các yêu cầu thay đổi để khẳng định rằng sự thay đổi trên CI là hợp lý.
- **Physical configuration audit (PCA):** nhằm mục đích khẳng định xem những gì khách hàng yêu cầu có được hiện thực hay không. Gồm 2 việc:
 - Kiểm tra lịch sử trò chuyện để phản ánh tính 2 chiều (traceability) giữa yêu cầu khách hàng và việc hiện thực code trong dự án.
 - Xác định những gì sẽ được phân phối cho khách hàng (executable files, source code, tài liệu đi kèm...) có đáp ứng yêu cầu khách hàng hay không.
- **Functional configuration audit (FCA):** nhằm mục đích khẳng định những gì khách hàng yêu cầu có được kiểm tra chặt chẽ trên sản phẩm tạo ra trước khi giao cho khách hàng hay không. Báo gồm:
 - Kiểm tra vết để phản ánh tính 2 chiều giữa yêu cầu khách hàng và việc kiểm tra sản phẩm.

Quản lý release (Release management)

Trong thực tế, có nhiều định nghĩa khác nhau về khái niệm **Release**. Về cơ bản, chúng ta có thể hiểu: Quá trình phát triển một phần mềm thường qua nhiều lần tích hợp, kết quả của mỗi lần tích hợp là một bản **Build**, trong rất nhiều bản Build đó, một số bản đáp ứng một số yêu cầu đã định hoặc lập kế hoạch trước (theo yêu cầu khách hàng), sẽ được gởi cho khách hàng để kiểm tra hoặc đánh giá. Các bản build này được gọi là **Release**. Công việc tạo ra và phân phối các bản release được gọi là **Công việc Release**. Theo cách hiểu này, sản phẩm sau cùng cũng là một bản release, đôi khi được gọi là **Final release**.

Trong quá trình release, việc quản lý đòi hỏi phải thực hiện các công việc sau:

- Baseline môi trường phát triển sản phẩm và các file, tài liệu (sẽ release).
- Thực hiện báo cáo CSAR
- Thực hiện các audit: PCA và FCA
- Đóng gói file và tài liệu sẽ release
- Xác nhận đã nhận bản release từ khách hàng

Lưu trữ và chép dự phòng (Backup & archive)

Lưu trữ và chép dự phòng là một hoạt động của QLCH và là một trong những hoạt động quan trọng phải có của sản xuất phần mềm. Nó giúp khắc phục các trường hợp rủi ro bị mất mát dữ liệu do thao tác sai, virus, hoặc sự cố phần cứng/ phần mềm. Ở khía cạnh khác, nó hỗ trợ cho hoạt động version control (đã nói ở trên) trong trường hợp ta muốn sử dụng những version khác nhau.

Lưu trữ và chép dự phòng đòi hỏi toàn bộ sản phẩm và sản phẩm trung gian của dự án phải được định kỳ chép dự phòng trên những thiết bị hoặc những nơi khác một cách an toàn.

Và khi dự án kết thúc, các hoạt động sau cần phải thực hiện:

- Lưu trữ toàn bộ dữ liệu của dự án, tuân thủ quy trình lưu trữ đã được thiết lập (định nghĩa bởi dự án hoặc quy định ở cấp công ty).
- Lưu trữ hoặc huỷ bỏ các tài liệu ở dạng giấy.
- Dọn sạch dữ liệu hoặc thông tin của dự án vừa kết thúc, sau khi đã chép lưu trữ.

Chương 7: Quản lý công nghệ phần mềm

Danh sách thành viên:

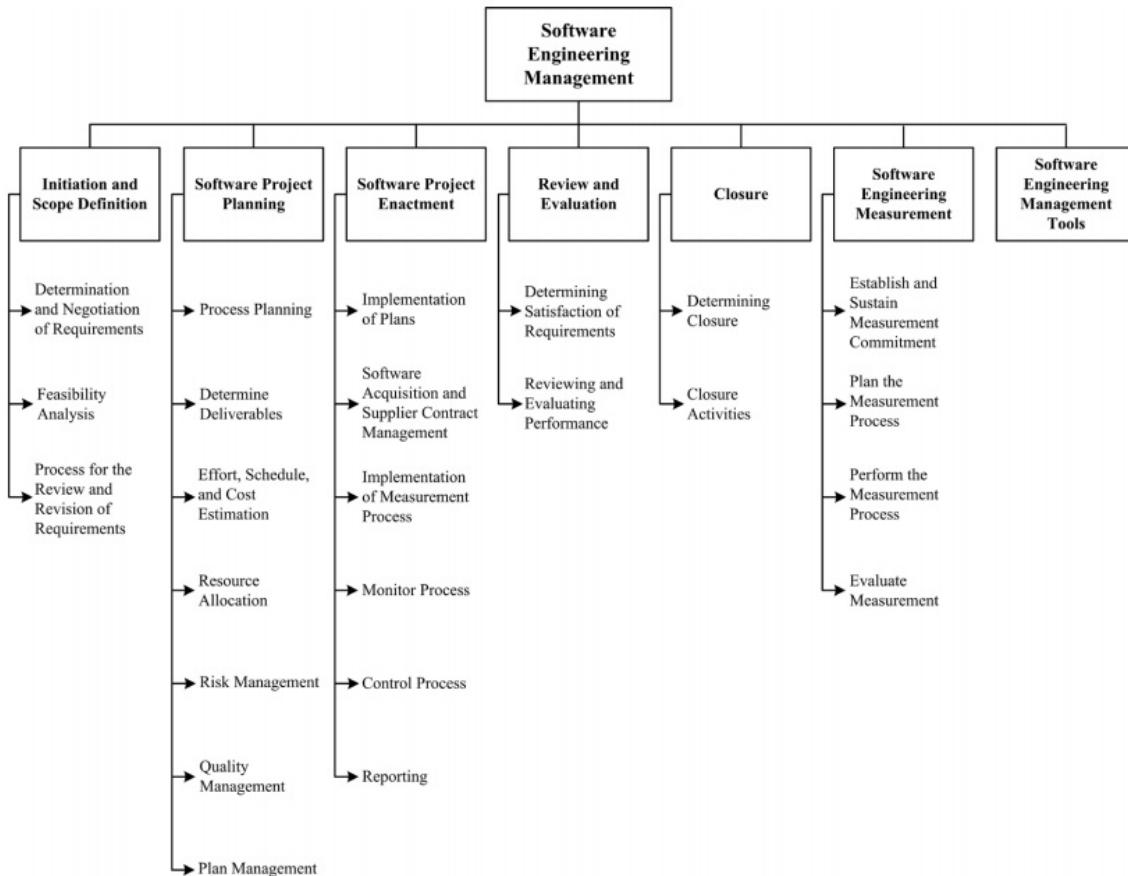
- Lê Thị Bích Liên
- Trần Văn Tiến
- Vũ Thị Thanh

Tổng quan

Quản lý công nghệ phần mềm có thể được định nghĩa là việc áp dụng các kế hoạch hoạt động quản lý, điều phối, đo lường, giám sát, kiểm soát và báo cáo để đảm bảo rằng các sản phẩm phần mềm và dịch vụ công nghệ phần mềm được phân phối hiệu quả, và đảm bảo lợi ích của các bên liên quan. Có những khía cạnh đặc trưng trong dự án phần mềm và chu trình phát triển của phần mềm, nó làm quá trình quản lý trở nên phức tạp bao gồm:

- Khách hàng thường là người không biết những gì là cần thiết, những gì là khả thi.
- Khách hàng thường thiếu sự đánh giá cao cho sự phức tạp vốn có trong công nghệ phần mềm, đặc biệt là về tác động của việc thay đổi.
- Sự hiểu biết và điều kiện thay đổi sẽ tạo ra các yêu cầu phần mềm mới hoặc thay đổi những yêu cầu của phần mềm.
- Phần mềm thường được xây dựng sử dụng một quá trình lặp đi lặp lại chứ không phải là một chuỗi các nhiệm vụ khép kín.
- Phần mềm kỹ thuật nhất thiết phải kết hợp sáng tạo và tính kỷ luật trong quá trình phát triển phần mềm. Duy trì một sự cân bằng thích hợp giữa hai bên là đôi khi là khó khăn.
- Mức độ của sự mới mẻ và phức tạp thường cao.
- Thường có sự thay đổi công nghệ rất nhanh chóng.

Hoạt động quản lý công nghệ phần mềm xảy ra ở ba cấp độ: tổ chức và quản lý cơ sở hạ tầng, quản lý dự án và quản lý các hệ thống đo lường. Trong chương này, sẽ tìm hiểu chi tiết về quản lý dự án và quản lý các hệ thống đo lường. Các vấn đề được trình bày trong chương này bao gồm:



1.Khởi tạo và định nghĩa phạm vi

Mục tiêu của các hoạt động này là xác định hiệu quả các yêu cầu của phần mềm sử dụng các phương pháp khác nhau, và đánh giá tính khả thi của dự án từ nhiều quan điểm. Một khi tính khả thi của dự án đã được xác định, thì nhiệm vụ tiếp theo là đặc tả yêu cầu và lựa chọn quy trình xử lý cho việc ra soát và xem lại các yêu cầu.

Quá trình khởi tạo và định nghĩa phạm vi sẽ theo tuần tự như sau: Xác định và điều chỉnh yêu cầu. Nếu khả thi thì xem xét và sửa đổi các yêu cầu.

- Xác định và điều chỉnh yêu cầu dựa vào:
 - Những yêu cầu được đưa ra.
 - Điều chỉnh yêu cầu với các bên liên quan
 - Dựa vào bản đặc tả yêu cầu SRS.
 - Phần mềm làm ra đáp ứng được nhu cầu của người sử dụng.
- Để đánh giá phần mềm có tính khả thi hay không, dựa vào các tiêu chí sau:
 - Được sử dụng trong thời gian dài mà hầu hết các lỗi ban đầu của nó đã được loại bỏ.
 - Giải pháp phân tích hoạt động của quá trình phát triển phần mềm.
 - Phân tích tài chính trên chu trình phát triển phần mềm.

- Tác động xã hội, chính trị
- Xem xét và sửa đổi yêu cầu bao gồm như:
 - Thay đổi cách quản lý trong việc xem xét và sửa đổi các yêu cầu.
 - Đánh giá khả năng thành công với những yêu cầu thay đổi.

1.1 Đàm phán và xác định các yêu cầu

Đàm phán và xác định các yêu cầu, xác định các công việc định làm. Các hoạt động bao gồm các yêu cầu về sự khám phá, phân tích, đặc điểm kỹ thuật và xác nhận. Các phương pháp và kỹ thuật cần được lựa chọn và áp dụng, xét đến các quan điểm của các bên liên quan. Điều này dẫn đến việc xác định phạm vi dự án trong đơn đặt hàng để đáp ứng mục tiêu và đảm bảo ràng buộc.

1.2 Phân tích tính khả thi

Mục đích của việc phân tích tính khả thi là để phát triển một mô tả rõ ràng về mục tiêu dự án và đánh giá các phương án khác nhau để xác định xem dự án này là một lựa chọn tốt cho những hạn chế của công nghệ, nguồn lực, tài chính, và xã hội / chính trị. Một dự án ban đầu và việc trình bày phạm vi sản phẩm, phân phối dự án, hạn chế thời gian dự án, và ước tính nguồn lực cần thiết phải được chuẩn bị. Tài nguyên bao gồm một số lượng đủ của những người có những kỹ năng cần thiết, cơ sở vật chất, cơ sở hạ tầng, và hỗ trợ (bên trong hoặc bên ngoài). Phân tích tính khả thi thường đòi hỏi ước tính gần đúng của nỗ lực và chi phí dựa trên các phương pháp thích hợp.

1.3 Quy trình về việc xem xét và sửa đổi yêu cầu

Các bên liên quan phải thống nhất về cách thức mà các yêu cầu và phạm vi cần được xem xét và sửa đổi. Phạm vi và yêu cầu sẽ không được “ấn định chính thức” nhưng có thể và nên được xem xét lại. Nếu thay đổi được chấp nhận, sau đó một số hình thức phân tích truy xuất nguồn gốc và phân tích rủi ro nên được sử dụng để xác định tác động của những thay đổi. Có nghĩa là phạm vi yêu cầu không phải luôn luôn được ấn định mà nó có thể thay đổi. Ví dụ, có thể thêm chức năng của phần mềm dựa vào việc phân tích các yêu cầu hoặc yêu cầu của khách hàng hoặc là tại thời điểm này thì ưu tiên công việc nào trước, nó thể thay đổi không cố định. Và nếu như những thay đổi đó được chấp nhận thì phải có phương pháp phân tích rủi ro để đánh giá mức độ rủi ro của việc thay đổi, xem xét xem việc thay đổi đó có hợp lý không.

2 Kế hoạch dự án phần mềm

Giai đoạn này yêu cầu thiết lập phạm vi công việc của dự án, điều chỉnh lại mục tiêu và xác định đường đi tới mục tiêu đó.

Bước đầu tiên trong việc lập kế hoạch dự án phần mềm phải được lựa chọn mô hình chu trình phát triển phần mềm, và có thể thiết kế nó dựa trên phạm vi dự án, yêu cầu dự án, và đánh giá rủi ro.

Trong tất cả các chu trình phát triển phần mềm (SDLC), đánh giá rủi ro phải được lên kế hoạch từ ban đầu dự án. Và danh sách rủi ro phải được đem thảo luận và chấp nhận bởi các bên liên quan. Quy trình và chịu trách nhiệm về việc xem xét và sửa đổi liên tục các kế hoạch dự án và các kế hoạch có liên quan cũng nên được quy định rõ ràng và thống nhất.

2.1 Quá trình lập kế hoạch

Chu trình phát triển phần mềm:



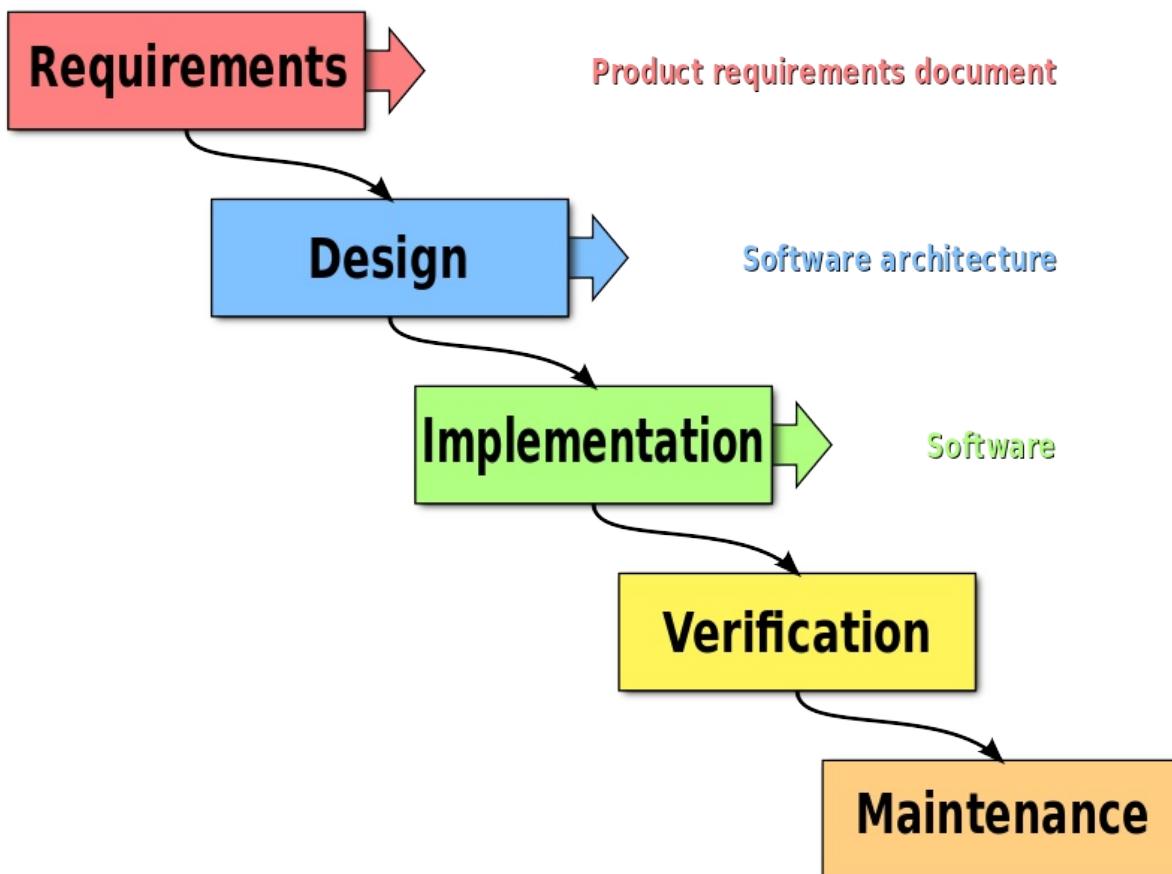
Các

giai đoạn trong chu trình phát triển phần mềm:

1	Giải pháp/Yêu cầu	Thực hiện khảo sát chi tiết yêu cầu khách hàng và tổng hợp vào tài liệu Giải pháp (Phân tích yêu cầu, Đặc tả yêu cầu, Prototype). Tài liệu giải pháp phải mô tả đầy đủ các yêu cầu về chức năng, phi chức năng, giao diện	Tài liệu đặc tả yêu cầu (SRS)/ prototype Tài liệu đặc tả yêu cầu (SRS)/ prototype
2	Phân tích/ Thiết kế	Thực hiện phân tích thiết kế tổng thể, phân tích và thiết kế chi tiết	Tài liệu thiết kế CSDL, chi tiết, workflow, diagrams
3	Lập trình	Lập trình viên thực hiện lập trình theo tài liệu Giải pháp và Thiết kế đã được phê duyệt.	Source code
4	Kiểm thử	Kiểm thử ở các giai đoạn phát triển	Tài liệu kiểm thử
5	Triển khai	Triển khai sản phẩm tới khách hàng	Biên bản nghiệm thu của khách

Các mô hình chu trình phát triển phần mềm: waterfall (mô hình thác nước), incremental (mô hình gia tăng), and spiral models (mô hình xoắn ốc)....

- **Waterfall model:**



Mô hình này bao gồm các giai đoạn xử lý nối tiếp nhau như sau:

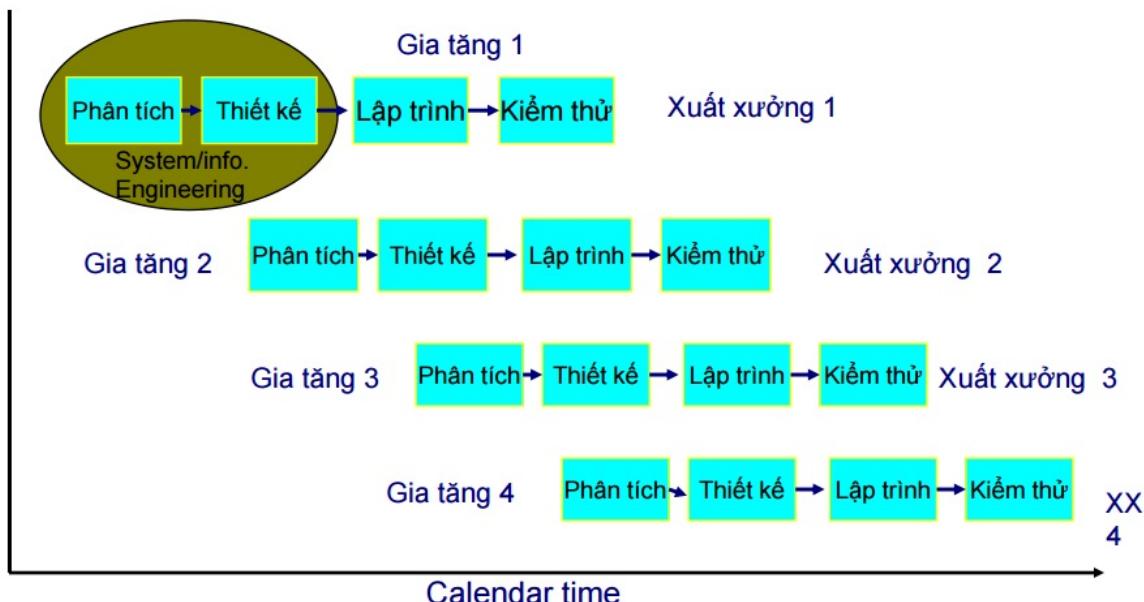
- Phân tích yêu cầu(Requirement Analysis): là giai đoạn xác định những Yêu cầu

liên quan đến chức năng và phi chức năng mà hệ thống phần mềm cần có. Giai đoạn này cần sự tham gia tích cực của khách hàng và kết thúc bằng một tài liệu được gọi là “Bản đặc tả yêu cầu phần mềm”. Tài liệu Đặc tả yêu cầu chính là nền tảng cho các hoạt động tiếp theo cho đến cuối dự án.

- Phân tích hệ thống và thiết kế (System Analysis and Design): là giai đoạn định ra làm thế nào để hệ thống phần mềm đáp ứng những yêu cầu mà khách hàng yêu cầu trong tài liệu SRS.

=> Nhược điểm của mô hình waterfall: Thực tế cho thấy đến những giai đoạn cuối của dự án mới có khả năng nhận ra sai sót trong những giai đoạn trước và phải quay lại để sửa chữa.

- Mô hình gia tăng:**



Ưu điểm.

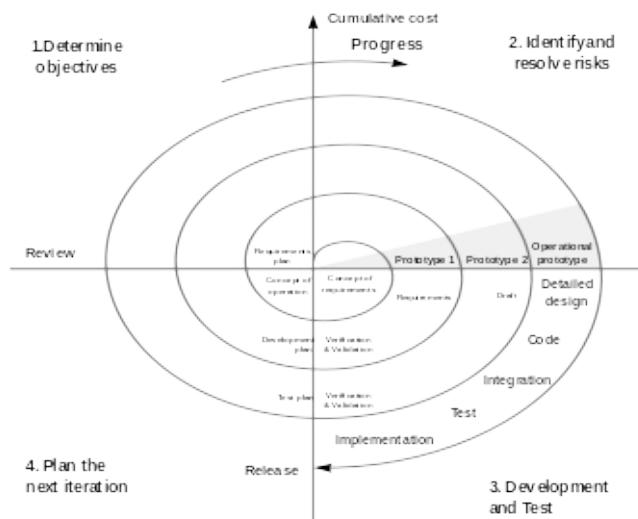
- Có thể sớm tạo ra nguyên mẫu của sản phẩm trong vòng đời phát triển của nó.
- Độ linh hoạt cao hơn và khi thay đổi yêu cầu dự án thì chi phí sẽ ít hơn nhiều.
- Việc phân chia thành các module cũng sẽ làm cho việc test nhẹ nhàng hơn, những lỗi sẽ dễ dàng phát hiện.
- Giảm chi phí cho lần đầu giao sản phẩm.
- Dễ dàng quản lý các rủi ro có thể phát sinh.

Nhược điểm.

- Cần phải có những khả năng thiết kế tốt và phương pháp tốt, để có thể hiểu rõ.
- Chi phí để phát triển theo phương pháp này là rất cao, cao hơn hẳn waterfall.

Khi nào áp dụng mô hình này:

- Áp dụng mô hình này khi yêu cầu của dự án là rõ ràng, đầy đủ, và nắm rõ được các yêu cầu.
- Khi sớm cần có một nguyên mẫu phần mềm để quảng bá, giới thiệu hoặc thử nghiệm.
- Sử dụng mô hình này khi một công nghệ mới được áp dụng.
- Tài nguyên và kỹ năng chuyên môn luôn sẵn sàng.
- Khi có một tính năng hay các mục tiêu có nguy cơ lỗi cao.



• Mô hình xoắn ốc:

- Trong mô hình xoắn ốc, quy trình phát triển phần mềm được thực hiện như một vòng xoáy ốc. Mỗi vòng xoắn ốc biểu diễn một hoạt động trong tiến trình phát triển phần mềm.
- Nó dựa trên ý tưởng là tối thiểu hóa rủi ro, bằng việc phân tích yếu tố rủi ro ở mỗi bước lặp và sử dụng phương pháp làm bản mẫu.
- Quá trình phát triển được chia thành nhiều bước lặp lại, mỗi bước bắt đầu bằng việc lập kế hoạch, phân tích rủi ro, rồi tạo bản mẫu, hoàn thiện và phát triển hệ thống, duyệt lại, và cứ thế tiếp tục.
- Quy trình 4 bước:
 1. *Lập kế hoạch*: xác định mục tiêu, giải pháp và ràng buộc;
 2. *Phân tích rủi ro*: phân tích các phương án, xác định và giải quyết rủi ro;
 3. *Phát triển và kiểm định*: phát triển sản phẩm “mức tiếp theo”. Xây dựng một hay một số biểu diễn của ứng dụng
 4. *Lên kế hoạch cho chu kỳ lặp tiếp theo*: kiểm duyệt tất cả các kết quả của các giai đoạn con xảy ra trước đó và lập kế hoạch cho chu kỳ lặp tiếp theo.
 - Tại mỗi vòng phát triển, nếu rủi ro lớn thì có thể dừng việc phát triển lại. Mô hình này thích hợp cho việc xây dựng những hệ thống lớn.
 - Theo IEEE / EIA Std. 12.207.0-1996, các yếu tố lập kế hoạch dự án bao gồm: Nguồn lực cần thiết để thực hiện các nhiệm vụ; Giao nhiệm vụ, phân công trách nhiệm; Biện pháp kiểm soát chất lượng trong suốt quá trình sử dụng; Cung cấp môi trường và cơ sở hạ tầng.

2.2 Quyết định phân phối

Kế hoạch dự án xác định các phân phối dự án mà có thể bao gồm, không bị giới hạn:

- Các phần mềm hoạt động
- Yêu cầu của khách hàng
- Đặc điểm chức năng
- Thiết kế kỹ thuật
- Tài liệu hướng dẫn thiết kế
- Mã nguồn
- Hướng dẫn sử dụng tài
- Nguyên tắc hoạt động
- Hướng dẫn cài đặt
- Thủ tục bảo trì
- Tài liệu đào tạo

2.3 Ước tính công sức, thời gian và chi phí

Phạm vi ước lượng đòi hỏi sự nỗ lực cần thiết cho một dự án, hoặc các bộ phận của một dự án, có thể được xác định bằng cách sử dụng một mô hình dự toán được hiệu chỉnh dựa trên kích thước và nỗ lực lịch sử dữ liệu (nếu có) và các phương pháp khác có liên quan như phán đoán của chuyên gia và tương tự. Để hoàn thành nhiệm vụ đồng thời và liên tục, các nhiệm vụ có thể được xác định và ghi chép bằng cách sử dụng một biểu đồ ví dụ như biểu đồ Gantt. Đối với dự án dự đoán chu trình phát triển phần mềm, đã cho rằng những công việc với thời gian bắt đầu dự án, khoảng thời gian triển khai dự án, và thời gian kết thúc là thường được đưa ra trong suốt quá trình lập kế hoạch. Đối với dự án chu trình phát triển phần mềm thích ứng, tỷ lệ chung của nỗ lực và tiến độ thường được phát triển từ những hiểu biết ban đầu về các yêu cầu, hoặc nói cách khác, những hạn chế về nỗ lực chung và tiến độ có thể được xác định và được sử dụng để xác định một ước tính ban đầu về số lượng các chu kỳ lặp đi lặp lại và ước tính của nỗ lực và các nguồn lực khác được giao cho mỗi chu kỳ.

Yêu cầu nguồn tài nguyên (ví dụ, con người và những công cụ) có thể được coi là sự ước lượng chi phí. Dự kiến ban đầu của nỗ lực, tiến độ, chi phí và là một hoạt động lặp đi lặp lại rằng nên được đàm phán và điều chỉnh các bên liên quan bị ảnh hưởng cho đến khi đạt được sự đồng thuận về tài nguyên và thời gian dành cho dự án hoàn thành.

2.4 Phân phối nguồn lực tài nguyên

Công cụ, con người, cơ sở vật chất cần được giao cho nhiệm vụ cụ thể. Phân bổ đòi hỏi phải cân bằng, chuyên môn, có đạo đức nghề nghiệp.

Điều chỉnh lịch trình/ chi phí là cần thiết nếu như các nguồn lực trở nên không có sẵn.

Quản lý dự án có thể cần phải thay đổi kích thước nhóm và cơ cấu để hoạt động đồng thời có thể thực thi một cách hiệu quả.

2.5 Quản lý rủi ro

Rủi ro và tính không chắc chắn là những khái niệm có liên quan nhưng khác biệt. Kết quả của tính không chắc chắn từ sự thiếu thông tin. Còn Rủi ro được đặc trưng bởi xác suất của một kết quả sẽ dẫn đến một tác động tiêu cực cộng với một mô tả đặc điểm của các tác động tiêu cực trên một dự án. Rủi ro thường là kết quả của sự không chắc chắn. Trái ngược với rủi ro là cơ hội, được đặc trưng bởi xác suất mà một sự kiện có một kết quả tích cực có thể xảy ra.

Quản lý rủi ro đòi hỏi phải xác định các yếu tố rủi ro và phân tích các khả năng và tác động tiềm năng của mỗi yếu tố rủi ro, ưu tiên các yếu tố nguy cơ, và phát triển các chiến lược giảm thiểu rủi ro để giảm xác suất và giảm thiểu các tác động tiêu cực nếu một yếu tố nguy cơ trở thành một vấn đề. Phương pháp đánh giá rủi ro đôi khi có thể được sử dụng để xác định và đánh giá các yếu tố nguy cơ.

Quản lý rủi ro cần phải được thực hiện không chỉ ở đầu của một dự án, mà còn thực hiện định kỳ trong suốt vòng đời dự án.

Tất cả các hoạt động quản lý dự án có thể được xem như là quản lý rủi ro. Ví dụ, dự toán chi phí giảm thiểu các nguy cơ mất tiền. Các tiêu chuẩn ISO / IEC định nghĩa quản lý rủi ro như:

1. một quá trình có tổ chức để xác định và xử lý các yếu tố nguy cơ.
2. một phương tiện tổ chức xác định và đo lường rủi ro (đánh giá rủi ro) và phát triển, lựa chọn, và các tùy chọn quản lý (phân tích rủi ro) cho giải quyết (xử lý rủi ro) những rủi ro này.
3. tổ chức, quá trình phân tích để xác định những gì có thể gây tổn hại hoặc mất mát (xác định rủi ro); để đánh giá và định lượng các rủi ro được xác định; và để phát triển. Nếu cần thiết, thực hiện một phương pháp thích hợp để ngăn chặn hoặc xử lý gây ra các rủi ro có thể dẫn đến thiệt hại đáng kể hoặc mất mát.

Quá trình xử lý rủi ro bao gồm:



IEEE

- Plan and implement risk management (lập kế hoạch và thực hiện quản lý rủi ro)
- Manage project risk profile (quản lý hồ sơ dự án rủi ro)
- Perform risk analysis (Thực hiện phân tích rủi ro)
- Perform risk treatment (Thực hiện xử lý rủi ro)
- Perform risk monitoringe (Thực hiện giám sát rủi ro)
- Evaluate risk management process (Đánh giá quá trình quản lý rủi ro)

2.6 Quản lý chất lượng

Yêu cầu chất lượng phần mềm nên được thống nhất, có lẽ cả về số lượng và chất lượng cho một dự án phần mềm và các sản phẩm liên quan. Nguồn cho phép đo chất lượng chấp nhận được nên được thiết lập cho từng yêu cầu chất lượng phần mềm dựa trên nhu cầu và kỳ vọng của các bên liên quan. Những thủ tục liên quan đến phần mềm liên tục đảm bảo chất lượng (SQA) và cải tiến chất lượng trong suốt quá trình phát triển, xác minh và xác nhận của các sản phẩm phần mềm chuyển giao, cũng nên được chỉ rõ trong hoạch định chất lượng.

Theo IEEE / EIA Std. 12.207,0-1996, một kế hoạch quản lý bảo đảm chất lượng nên bao gồm:

- Tiêu chuẩn chất lượng, phương pháp, quy trình, và các công cụ để thực hiện chất lượng đảm bảo hoạt động (hoặc tài liệu tham khảo trong các tài liệu chính thức của tổ chức).
- Thủ tục xem xét hợp đồng và phối hợp của chúng.
- Thủ tục xác định, thu thập, lập hồ sơ, bảo trì, và định đoạt chất lượng ghi chép.
- Tài nguyên, lịch trình, và trách nhiệm tiến hành các hoạt động đảm bảo chất lượng.
- Các hoạt động và nhiệm vụ được lựa chọn từ các quá trình hỗ trợ như xác minh, kiểm định, đánh giá chung, kiểm toán và giải quyết vấn đề.

2.7 Quản lý kế hoạch

Kế hoạch và quy trình lựa chọn phát triển phần mềm cần được theo dõi một cách hệ thống, được xem xét, báo cáo, và sửa đổi khi thích hợp. Kế hoạch kết hợp với quy trình hỗ trợ (ví dụ, tài liệu, quản lý cấu hình phần mềm, và giải quyết vấn đề) cũng cần được quản lý. Báo cáo, giám sát và kiểm soát một dự án nên được lựa chọn sao cho phù hợp với chu trình phát triển phần mềm và những thực tế của dự án; các kế hoạch nên giả lập giải thích các sản phẩm khác nhau sử dụng để quản lý dự án.

Khi dự án tiến triển, kế hoạch phải được cập nhật để phản ánh:

- Yêu cầu sửa đổi.
- Lịch trình mở rộng
- Những thay đổi trong thủ tục kiểm tra.
- Chức năng phần mềm thay đổi.

Việc tuân thủ kế hoạch phải được hướng dẫn một cách hệ thống, theo dõi, xem xét, báo cáo, và sửa đổi. Kế hoạch quản lý dự án là các mục cấu hình và là một phần của quá trình phát triển phần mềm.

3 Thực hiện dự án phần mềm

Trong suốt quá trình thực hiện dự án phần mềm, kế hoạch được thực hiện và các quy trình trong kế hoạch cũng được thực hiện. Trong quá trình thực hiện phần mềm không nên tập trung vào việc tuân thủ các quy trình được lựa chọn với một kỳ vọng là nếu tuân thủ theo quy trình này thì sẽ dẫn đến sự hài lòng về yêu cầu của khách hàng và đạt được mục tiêu dự án. Các hoạt động cơ bản của việc thực hiện phần mềm bao gồm các hoạt động để kiểm soát, điều hành và báo cáo.

3.1 Sự thực hiện kế hoạch

Các hoạt động của dự án cần được thực hiện phù hợp với kế hoạch dự án và các kế hoạch hỗ trợ.

Tài nguyên của dự án ví dụ như con người, công nghệ và nhà tài trợ được sử dụng để tạo ra sản phẩm. Sản phẩm của dự án có thể biết đến như thiết kế hệ thống, mã nguồn phần mềm và các ca kiểm thử sẽ được tạo ra trong quá trình thực hiện kế hoạch.

3.2 Thu nhận phần mềm và quản lý hợp đồng nhà cung cấp

Thu nhận phần mềm và quản lý hợp đồng nhà cung cấp là khái niệm có liên quan với các vấn đề về ký kết hợp đồng. Hợp đồng này bao gồm khách hàng của các tổ chức phần mềm với tổ chức cung cấp phần mềm.

Các vấn đề về ký kết hợp đồng bao gồm việc lựa chọn loại hợp đồng thích hợp. Có các loại hợp đồng như hợp đồng fix price, time and materials, ... Hợp đồng fix price là loại hợp đồng mà khách hàng trả cho nhà cung cấp giá và thời gian bàn giao là không đổi. Nhà cung cấp có thể quyết định sử dụng bao nhiêu người để hoàn thành sản phẩm. Ví dụ khách hàng thuê nhà cung cấp làm một phần mềm trong thời gian là 6 tháng với số tiền là 100000000 đồng thì đây là dạng hợp đồng fix price. Trong khi đó hợp đồng time and materials là loại hợp đồng mà khách hàng sẽ trả cho nhà cung cấp với số người là cố định còn số tiền sẽ trả theo thời gian làm không cố định giá trước và thời gian làm xong. Ví dụ khách hàng thuê nhà cung cấp làm một phần mềm với 5 người và số tiền trả cho 1 người trong một ngày là 100 USD thì đây là một loại hợp đồng time and materials

Thỏa thuận giữa khách hàng và nhà cung cấp thường phải định rõ phạm vi công việc và các sản phẩm bàn giao, các hình phạt cho việc bàn giao muộn và không giao được sản phẩm và sở hữu trí tuệ. Đối với các phần mềm được phát triển bởi chính nhà cung cấp, các thỏa thuận thường chỉ ra yêu cầu chất lượng phần mềm và điều kiện chấp nhận được của sản phẩm được bàn giao.

Sau khi thỏa thuận được đưa ra, cần quản lý việc thực hiện dự án sao cho phù hợp với các điều khoản của các thỏa thuận.

3.3 Thực hiện quy trình đo lường

Quy trình đo lường nên được thực hiện trong các dự án phần mềm để đảm bảo dữ liệu hữu ích có liên quan được thu thập. Xem thêm trong mục 6.

3.4 Quy trình giám sát

Việc thực hiện các kế hoạch dự án và các kế hoạch liên quan đến dự án cần được đánh giá liên tục tại các khoảng thời gian được định trước. Ngoài ra kết quả đầu ra và tiêu chuẩn hoàn thiện cho mỗi công việc cũng cần được đánh giá.

Nên phân chia việc đánh giá theo các điều khoản và yêu cầu trong các điều khoản. Các tiêu chuẩn cần được đánh giá:

- Hiệu suất làm việc
- Tuân thủ lịch trình
- Chi phí đã dùng cho đến hiện tại
- Kiểm tra việc sử dụng tài nguyên.
- Rủi ro của dự án cũng nên được xem xét lại
- Đánh giá xem các yêu cầu chất lượng phần mềm đã được đáp ứng hay chưa

Dữ liệu đo lường nên được phân tích và đánh giá. Làm rõ sự sai biệt giữa dự kiến và thực tế sai biệt này có thể xấu hoặc tốt. Một số sai biệt có thể kể đến đó là sai biệt về lịch biểu, sai biệt về chi phí, sai biệt về phạm vi phần mềm. Các hoạt động đánh giá này có thể phát hiện ra vấn đề và các rủi ro có thể xảy ra. Kết quả của các hoạt động này nên được ghi lại.

3.5 Quy trình kiểm soát

Các kết quả của hoạt động giám sát dự án cung cấp cơ sở để đưa ra các quyết định có thể được thực hiện. Khi thấy được rủi ro có thể xảy ra và tác động của các rủi ro đó thì cần thực hiện các thay đổi, điều chỉnh cho dự án. Các hoạt động điều chỉnh diễn ra khi tiến độ dự án không đúng lịch biểu, khi chi phí dự án có nguy cơ tăng, khi chất lượng công việc hoặc chất lượng sản phẩm có nguy cơ giảm. Ứng với mỗi nguyên nhân thì hoạt động cũng phải khác nhau ví dụ như:

Khi dự án diễn ra không đúng lịch biểu thì cần phải điều chỉnh lại lịch biểu, thêm người vào dự án hoặc mua hay thuê thêm thiết bị hoặc phần mềm tốt hơn, cải tiến cách làm việc, tập trung vào các việc chủ chốt và làm thêm giờ.

Khi kinh phí dự án có nguy cơ tăng lên thì nên có các hoạt động sau:

- Hạ thấp yêu cầu chất lượng sản phẩm
 - Thuê lao động giá rẻ
 - Rút ngắn thời gian huấn luyện
- Khi chất lượng công việc hoặc sản phẩm có nguy cơ giảm
- Tăng cường kiểm tra chất lượng
 - Thuê thêm tư vấn
 - Tập trung vào những khâu trọng yếu ảnh hưởng đến chất lượng
 - Kiểm tra chéo

Trong một số trường hợp quy trình kiểm soát có thể bị bỏ quên trong dự án. Điều này là không tốt. Vì vậy trong mọi trường hợp cần thực hiện đầy đủ việc kiểm soát. Các quyết định đưa ra nên được ghi chép lại và thông báo cho các bên liên quan được biết. Kế hoạch cũng nên được xem xét lại và sửa đổi khi cần thiết, dữ liệu liên quan cũng cần được ghi chép lại.

3.6 Báo cáo

Báo cáo là việc cần thiết cho việc giám sát và kiểm soát dự án. Người quản lý dự án có trách nhiệm lập báo cáo cho dự án. Tại thời điểm cụ thể đã được thỏa thuận trước thì các nội dung về thời gian, tiến độ cần được báo cáo cho cả hai trong dự án và các bên liên quan bên ngoài như khách hàng hoặc người sử dụng. Báo cáo cần tập trung vào các thông tin cần thiết, tình trạng dự án và các mục tiêu có thể chưa đạt được.

4 Xem xét và đánh giá

Tại các lần được định trước và khi cần thiết, tiến độ tổng thể hướng tới việc đạt được các mục tiêu đề ra và sự hài lòng về yêu cầu của các bên liên quan như người dùng và khách hàng cần được đánh giá. Tương tự như vậy việc đánh giá về hiệu quả, chất lượng phần mềm, các nhân viên tham gia và các công cụ và phương pháp làm việc cũng cần được thực hiện thường xuyên và xác định bởi hoàn cảnh.

4.1 Xác định sự hài lòng của yêu cầu

Bởi vì đạt được sự hài lòng của các bên liên quan là mục tiêu chính của việc quản lý dự án phần mềm, tiến độ đạt được các mục tiêu này nên được đánh giá theo định kỳ. Tiến độ và kết quả của các mốc chính của dự án cần được đánh giá (ví dụ khi hoàn thành thiết kế kiến trúc cho dự án thì cần đánh giá xem thiết kế kiến trúc đó đã đạt yêu cầu hay chưa) hoặc sau khi hoàn thành một chu kỳ phát triển lặp lại mà kết quả một sản phẩm tăng.

Chênh lệnh từ phần mềm so với yêu cầu cần được xác định và cần thực hiện các hành động phù hợp.

Như trong các hoạt động kiểm soát ở phần 3.5 thì quản lý phần mềm và cấu trúc phần mềm cần được theo sát. Khi có cần phải sửa đổi kế hoạch thì phải báo cho các bên liên quan được biết, viết thành tài liệu.

4.2 Xem xét và đánh giá hiệu suất

Đánh giá hiệu suất định kỳ cho cán bộ dự án có thể cung cấp các thông tin như khả năng tuân thủ kế hoạch và quy trình cũng như lĩnh vực mạnh yếu của từng nhân viên, khả năng làm việc trong nhóm. Cần sử dụng các phương pháp khác nhau, các công cụ kỹ thuật được

áp dụng để đánh giá chính xác hiệu quả làm việc và sự phù hợp của nhân viên. Nên có hệ thống đánh giá định kỳ, tiện ích trong từng bối cảnh dự án. Khi thích hợp, thay đổi phải được thực hiện và quản lý.

5 Kết thúc dự án

Tổng thể dự án hoặc các pha nhỏ hơn của dự án hoặc quá trình phát triển đi tới đích khi tất cả các kế hoạch hoặc tiến trình đều đã được thực hiện và hoàn thành. Các tiêu chuẩn để đánh giá cũng nên được xem xét. Chỉ khi dự án đã kết thúc thì các hoạt động lưu trữ, cải thiện, đánh giá lại mới được thực hiện

5.1 Quyết định kết thúc dự án

Sự khép lại dự án xảy ra khi các nhiệm vụ cụ thể cho từng pha, vòng lặp hoặc dự án được hoàn thành và các kết quả đạt được cũng phải thỏa mãn các tiêu chuẩn hoàn thành được xác nhận từ trước. Các yêu cầu phần mềm có thể được xác nhận là thỏa mãn hay không và mức độ đạt được các mục tiêu có thể được xác định. Tiến trình kết thúc phải bao gồm các bên liên quan và phải có sự xác nhận hoàn thành bằng văn bản của các bên liên quan đó. Mọi vấn đề còn tồn tại cũng phải được đưa vào văn bản.

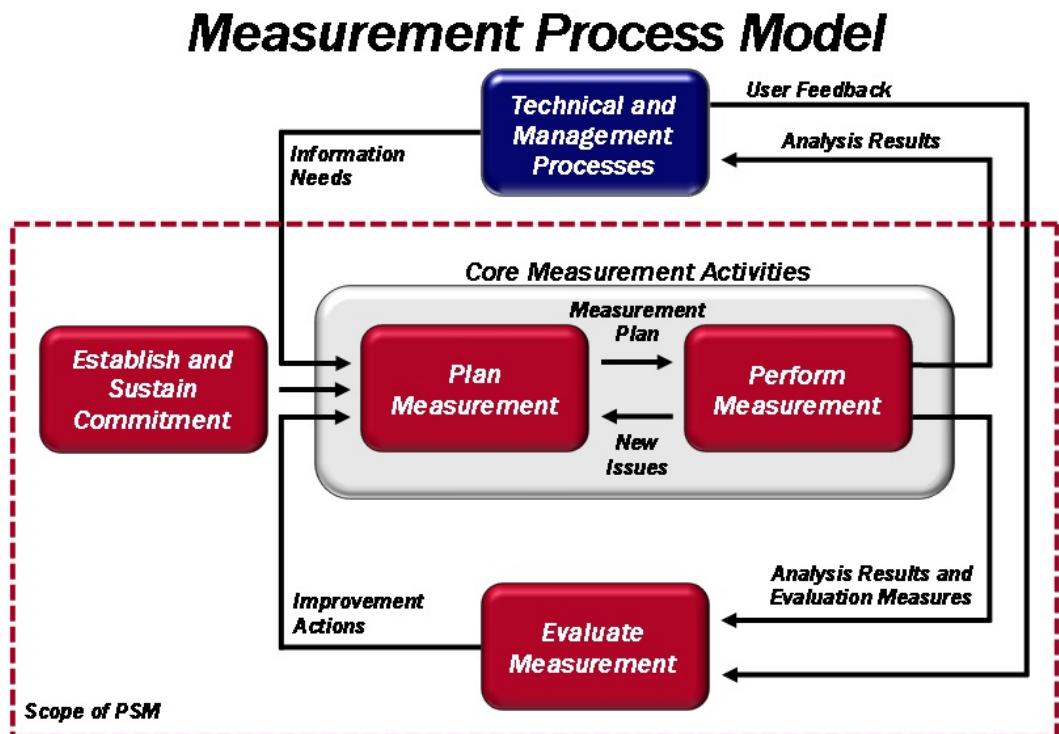
5.2 Các hoạt động kết thúc dự án

Sau khi việc kết thúc dự án được xác nhận, việc lưu trữ các tài liệu dự án cần được thực hiện theo sự thỏa thuận với các bên liên quan về phương thức, địa điểm, thời gian. Nó có thể bao gồm cả việc hủy các thông tin nhạy cảm, phần mềm và các phương tiện đang lưu trữ. Cơ sở dữ liệu đánh giá của tổ chức cũng nên được cập nhật với các dữ liệu liên quan tới dự án. Việc phân tích lại các pha của dự án cũng nên được xem lại sao cho các vấn đề, rủi ro, cơ hội gặp phải có thể được phân tích kỹ càng. Các bài học được rút ra từ việc phân tích dự án nên được đưa vào tri thức của tổ chức.

6 Đo lường công nghệ phần mềm

Sự quan trọng của đo lường và vai trò của nó trong việc quản trị và phát triển phần mềm đã được công nhận rộng rãi. Sự đánh giá hiệu quả phần mềm đã trở thành một trong những nền tảng để đo sự trưởng thành của của tổ chức. Đo lường có thể được áp dụng cho tổ chức, dự án, tiến trình và các sản phẩm trong quá trình làm việc. Ở đây chỉ đề cập đến áp

dụng của đo lường ở mức độ dự án, tiến trình và các sản phẩm tạo ra. **Mô hình đo lường phần mềm cơ bản:**



PSM VERSION 6.2 - 1

6.1 Cam kết thiết lập và duy trì sự đo lường

- **Yêu cầu đo lường:** Sự đo lường phải được thiết lập tuân theo các mục tiêu của tổ chức và được định hướng bởi các yêu cầu của tổ chức và dự án (ví dụ một yêu cầu của tổ chức là “sản phẩm tạo ra đầu tiên trên thị trường”)
- **Phạm vi đo lường:** Các đơn vị thuộc tổ chức mà mỗi yêu cầu đo lường được áp dụng nên được xác định rõ ràng. Nó có thể bao gồm một lĩnh vực chức năng, một dự án, một địa điểm hoặc toàn bộ doanh nghiệp. Ràng buộc thời gian của sự đo lường cũng nên được xem xét cẩn thận vì một số đo lường cần một khoảng thời gian dài ví dụ như để điều chỉnh mô hình dự báo (dự báo nguồn lực, chi phí...)
- Sự cam kết có một nhóm để đo lường phải được thiết lập và phải được hỗ trợ bởi các tài nguyên sẵn có
- **Tài nguyên để đo lường:** Sự cam kết của tổ chức cho việc đo lường rõ ràng là một yếu tố cơ bản nhất của sự thành công trong việc đo lường. Đầu tiên là việc phân bổ tài nguyên cho tiến trình đo lường. Các tài nguyên bao gồm con người, công cụ, tiền bạc, sự huấn luyện...

6.2 Lập kế hoạch cho tiến trình đo lường

- **Đặc tả đơn vị tổ chức:** Đơn vị tổ chức cung cấp bối cảnh để đo lường vì vậy bối cảnh tổ chức cần phải làm rõ ràng bao gồm cả những điều phát hiện ra trong quá trình đo lường. Đặc điểm của tổ chức có thể được ví dụ như các hoạt động doanh nghiệp, công nghệ, cấu trúc doanh nghiệp
- **Xác định các thông tin cần thiết:** Các thông tin cần thiết dựa vào các mục tiêu, ràng buộc, rủi ro và các vấn đề của tổ chức. Nó có thể được rút ra từ các mục tiêu kinh doanh, mục tiêu doanh nghiệp, mục tiêu luật pháp hoặc mục tiêu sản phẩm. Nó cần phải được xác định và ưu tiên. Sau đó các mục tiêu nhỏ hơn có thể được lựa chọn, làm tài liệu và kiểm tra bởi các bên liên quan.
- **Lựa chọn các độ đo:** Các độ đo có thể nên được lựa chọn với những liên kết tới các thông tin cần thiết kể trên. Độ đo nên được lựa chọn dựa trên độ ưu tiên của các thông tin cần thiết kết hợp với các tiêu chí khác như chi phí thu thập dữ liệu, khả năng tiến trình bị hủy bỏ trong quá trình thu thập, sự dễ dàng đạt được độ chính xác...
- **Định nghĩa quy trình thu thập dữ liệu, phân tích và báo cáo:** Việc này bao gồm quy trình thu thập dữ liệu, lưu trữ, kiểm định, phân tích, báo cáo và quản lý dữ liệu
- **Lựa chọn các tiêu chuẩn để đánh giá các sản phẩm thông tin:** Các tiêu chuẩn để đánh giá bị ảnh hưởng bởi các mục tiêu kinh doanh cũng như kỹ thuật của tổ chức. Các sản phẩm thông tin bao gồm những cái liên quan đến sản phẩm đang được làm cũng như những cái liên quan tới các tiến trình đang được sử dụng để quản lý và đo lường dự án
- **Cung cấp tài nguyên cho công việc đánh giá:** Kế hoạch đánh giá phải được xem xét và chấp thuận bởi các bên liên quan bao gồm tất cả các thủ tục thu thập dữ liệu, lưu trữ, phân tích và báo cáo; các tiêu chuẩn đánh giá, lập lịch và các trách nhiệm liên đới. Các tiêu chuẩn để đánh giá các sản phẩm trên phải được thiết lập ở mức độ tổ chức hoặc cao hơn để có thể làm nền tảng cho những công việc việc xem xét này. Những tiêu chuẩn đó nên được tham khảo từ những kinh nghiệm của các dự án trước, sự sẵn sàng của các nguồn lực hiện tại. Sự thông qua biểu thị cho sự đồng thuận tiến trình đánh giá
- **Xác định tài nguyên để thực hiện các tác vụ đã được chấp thuận và lên kế hoạch:** Các tài nguyên sẵn có có thể được cấp dần dần vì có thể sẽ phải thử nghiệm trước khi được áp dụng rộng rãi. Đặc biệt chú ý tới yêu cầu tài nguyên cần thiết để triển khai một tiến trình hoặc độ đo mới.
- **Mua và triển khai các công nghệ hỗ trợ:** Việc này bao gồm sự đánh giá các công nghệ hỗ trợ hiện có, lựa chọn công nghệ phù hợp nhất, mua lại nó và triển khai để sẵn sàng sử dụng

Danh sách các nhóm độ đo dùng để đo lường ví dụ:

Information Category	Measurable Concept	Questions Addressed
Schedule and Progress	Milestone Completion	Is the project meeting scheduled milestones? Are critical tasks or delivery dates slipping?
	Work Unit Progress	How are specific activities and products progressing?
	Incremental Capability	Is capability being delivered as scheduled in incremental builds and releases?
Resources and Cost	Personnel Effort	Is effort being expended according to plan? Is there enough staff with the required skills?
	Financial Performance	Is project spending meeting budget and schedule objectives?
	Environment and Support Resources	Are needed facilities, equipment, and materials available?
Product Size and Stability	Physical Size and Stability	How much are the product's size, content, physical characteristics, or interfaces changing?
	Functional Size and Stability	How much are the requirements and associated functionality changing?
Product Quality	Functional Correctness	Is the product good enough for delivery to the user? Are identified problems being resolved?
	Supportability - Maintainability	How much maintenance does the system require? How difficult is it to maintain?
	Efficiency	Does the target system make efficient use of system resources?
	Portability	To what extent can the functionality be re-hosted

Sau khi có danh sách những độ đo, chọn những độ đo tốt nhất phù hợp tổ chức

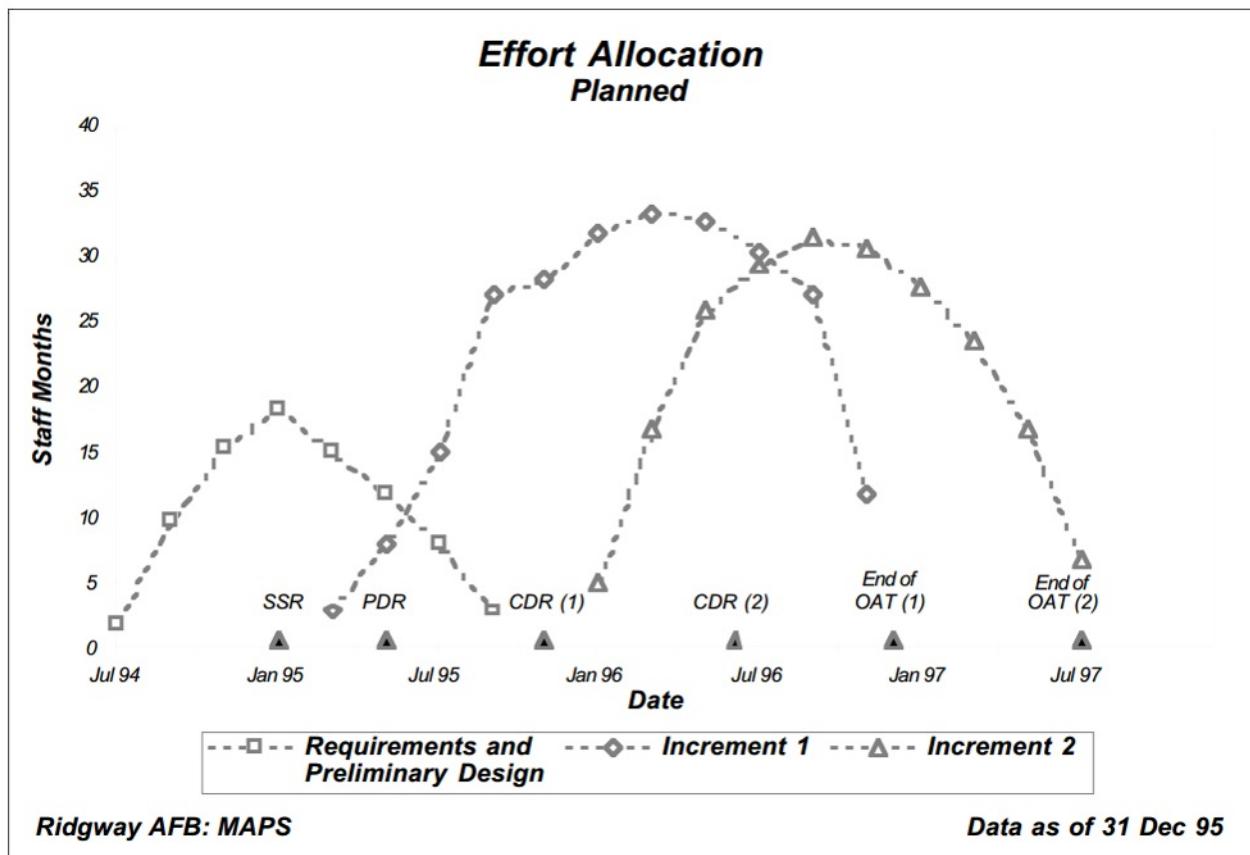
Schedule and Progress

- **Milestone completion**
- **Critical path performance**
- **Work unit progress**
- **Incremental Capability**

Prospective Measures

- Requirements traced
- Requirements tested
- Requirements status
- Problem reports opened
- Problem reports closed
- Reviews completed
- Change requests opened
- Change requests resolved
- Units designed
- Units coded
- Units integrated
- Test cases attempted
- Test cases passed
- Action item opened
- **Action item completed**

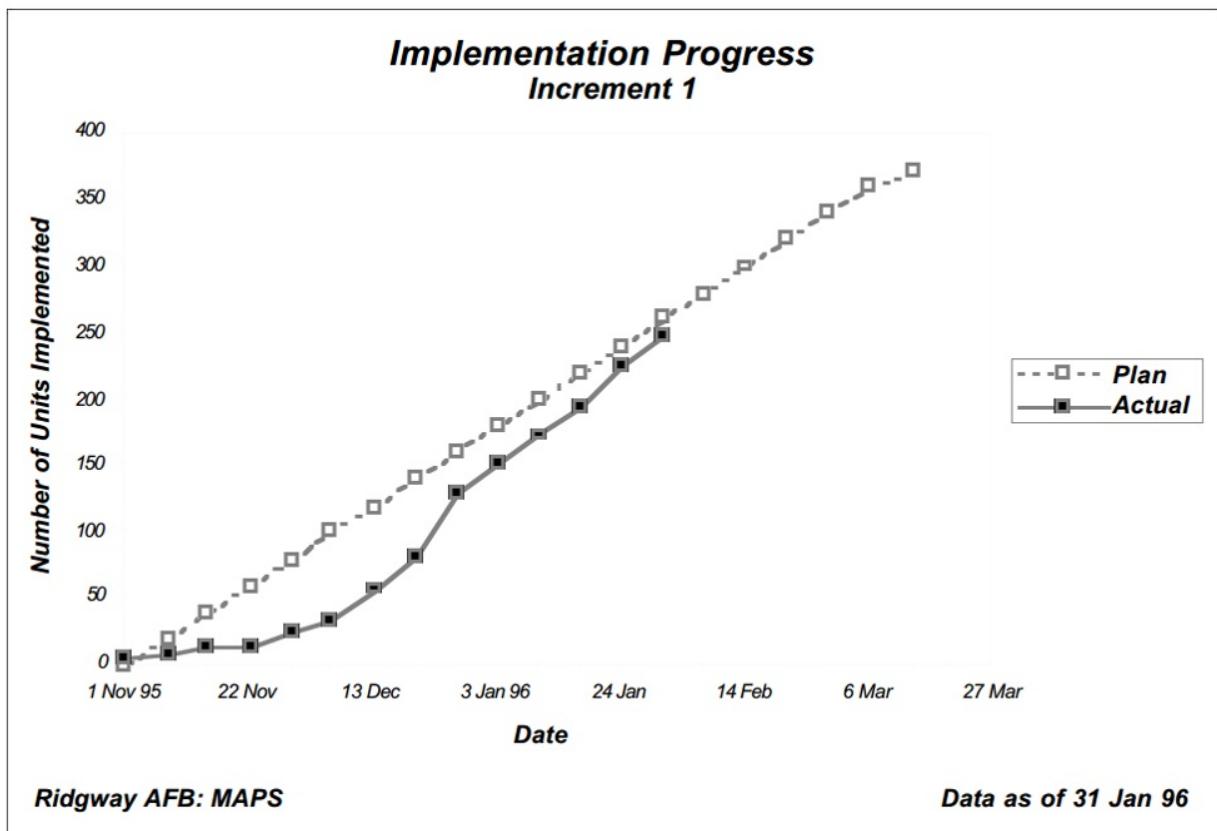
Một ví dụ về biểu đồ dự đoán nhân lực trong bước lập kế hoạch



6.3 Thực hiện tiến trình đo lường

- *Tích hợp thủ tục đo lường với các tiến trình phần mềm liên quan:* Thủ tục đo lường như thu thập dữ liệu, phải được tích hợp vào tiến trình phần mềm nó đang đo lường. Việc này có thể bao gồm thay đổi tiến trình phần mềm hiện tại cho phù hợp với việc thu thập dữ liệu. Nó cũng có thể bao gồm việc phân tích tiến trình phần mềm hiện tại để giảm tối đa công sức và ảnh hưởng tới người sử dụng sao cho tiến trình này được chấp nhận. Việc huấn luyện và hỗ trợ cũng cần phải được cung cấp. Tiến trình phân tích dữ liệu và báo cáo thường được tích hợp vào tiến trình dự án theo một cách tương tự.
- *Thu thập dữ liệu:* Dữ liệu phải được thu thập, kiểm định và lưu trữ. Việc thu thập dữ liệu nhiều khi cũng được tự động hóa bằng các công cụ phần mềm hỗ trợ. Dữ liệu sau đó có thể được tổng hợp, chuyển đổi và lưu trữ như một phần của tiến trình phân tích. Kết quả của quá trình phân tích thường được thể hiện bằng biểu đồ, số hoặc những thứ tương tự, từ đó có thể được gửi tới các bên liên quan. Các kết quả và kết luận thường được xem xét lại bằng một tiến trình định nghĩa bởi tổ chức (hình thức hoặc không hình thức). Người cung cấp dữ liệu và người thực hiện đánh giá nên tham gia vào quá trình xem xét lại này nhằm đảm bảo tính chính xác và có ý nghĩa của dữ liệu thu được.

Ví dụ về biểu đồ sau khi phân tích dữ liệu

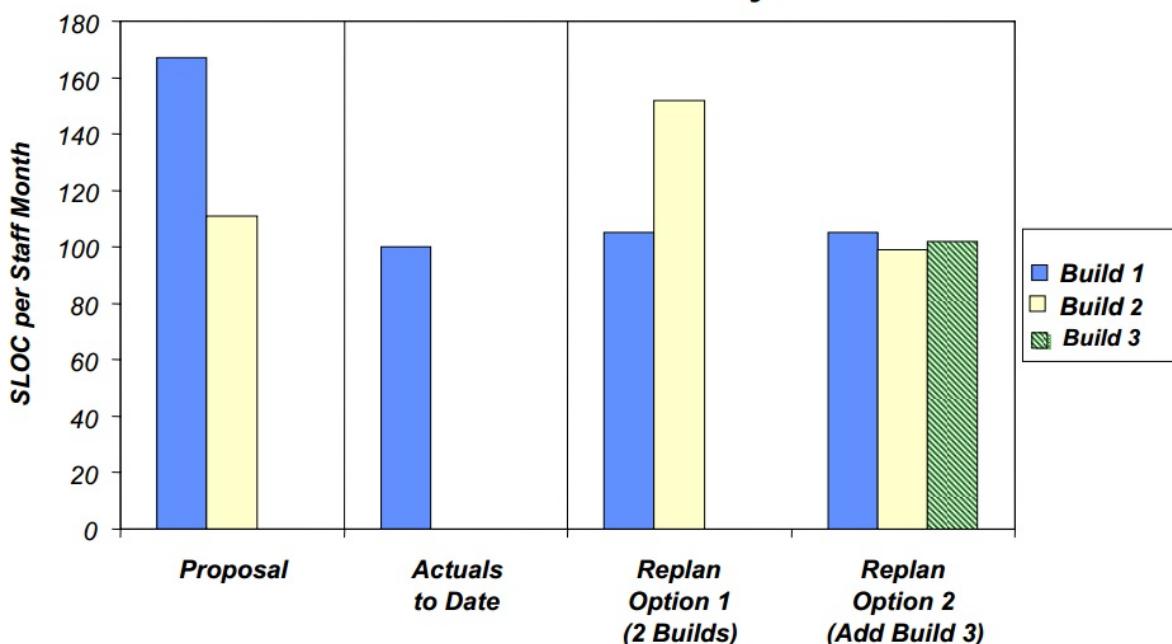


- *Truyền đạt kết quả:* Những sản phẩm thông tin phải được làm tài liệu và truyền đạt lại cho người dùng và các bên liên quan

6.4 Đánh giá sự đo lường

- Đánh giá sản phẩm thông tin và các tiến trình đánh giá với các tiêu chuẩn đánh giá đồng thời xem xét độ mạnh yếu của sản phẩm thông tin tương ứng. Sự đánh giá này có thể được thực hiện bởi các tiến trình nội bộ hoặc các tổ chức bên ngoài. Nó thường bao gồm các thông tin phản hồi từ phía người dùng tham gia đánh giá. Những bài học thu được nên được ghi lại vào cơ sở dữ liệu thích hợp
- Xác định sự cải tiến trong tương lai: Sự cải thiện này có thể trong cách biểu thị, đơn vị đo hoặc phân loại lại các danh mục đánh giá. Chi phí và lợi ích của sự cải thiện này nên được xác định và các hành động cải tiến thích hợp nên được báo cáo lại
- Truyền đạt các cải tiến đề xuất cho người chủ tiến trình và các bên liên quan để xem xét và thông qua nó.

Ví dụ về biểu đồ đánh giá



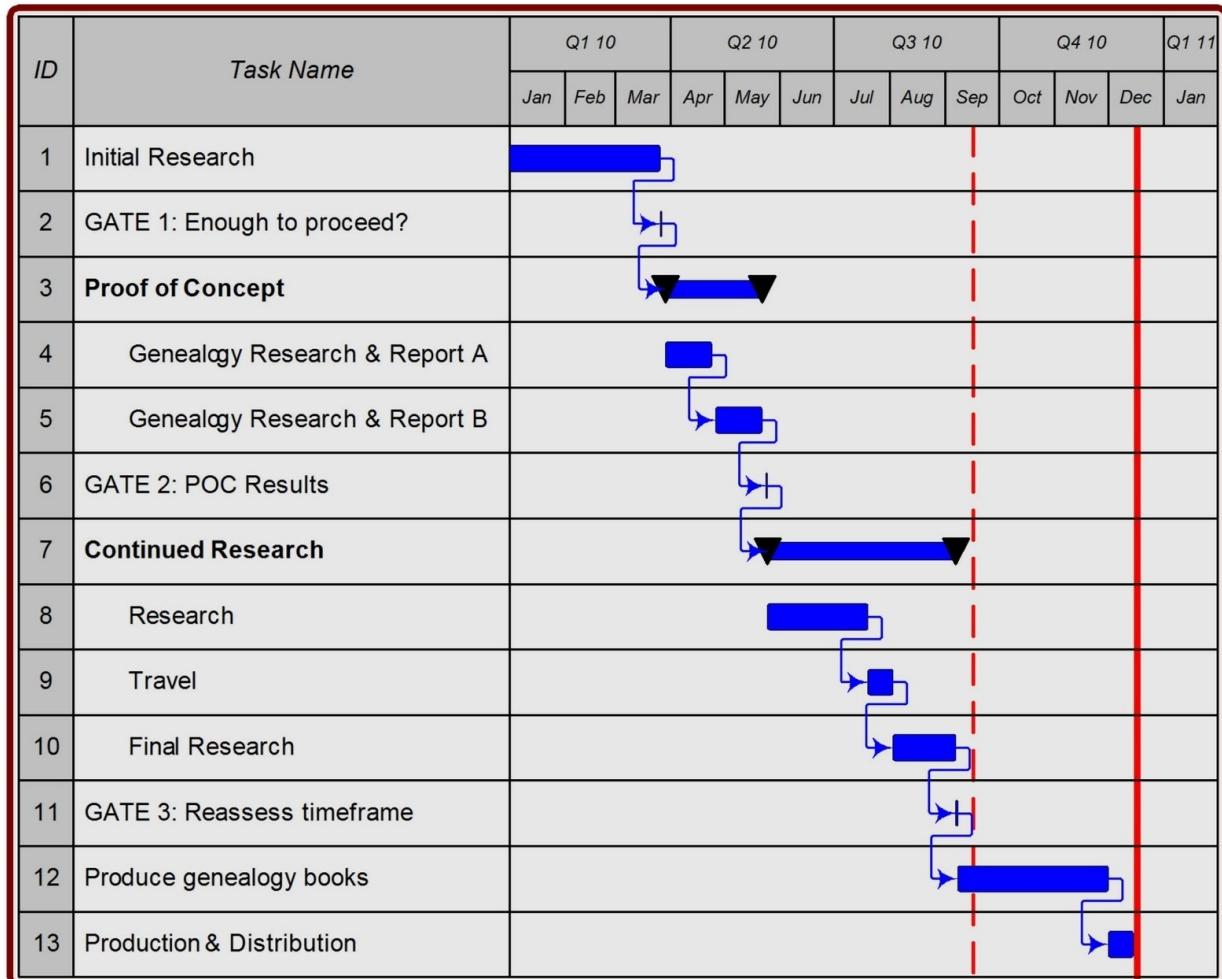
7 Các công cụ quản lý công nghệ phần mềm

Các công cụ quản lý công nghệ phần mềm thường được sử dụng để cung cấp tầm nhìn xa và quản lý tiến trình quản lý công nghệ phần mềm. Một số công cụ tự động và một số công cụ phải viết bằng tay. Các công cụ phục vụ quản lý công nghệ phần mềm có thể được chia thành một số loại như sau:

- Công cụ lập kế hoạch và lần vết:* Công cụ này dùng để ước lượng nhân công dự án, chi phí và để chuẩn bị việc lập kế hoạch dự án. Công cụ lập kế hoạch cũng có thể bao

gồm công cụ lập lịch tự động, các công cụ này phân tích các tác vụ trong bảng phân rã công việc, thời gian dự kiến, mối quan hệ trước sau... để đưa ra một lịch trình công việc dưới dạng biểu đồ Gantt. Công cụ lần lượt có thể được sử dụng để theo dõi các mốc dự án, lên lịch họp và các hành động

Một ví dụ về công cụ lập kế hoạch Microsoft Visio

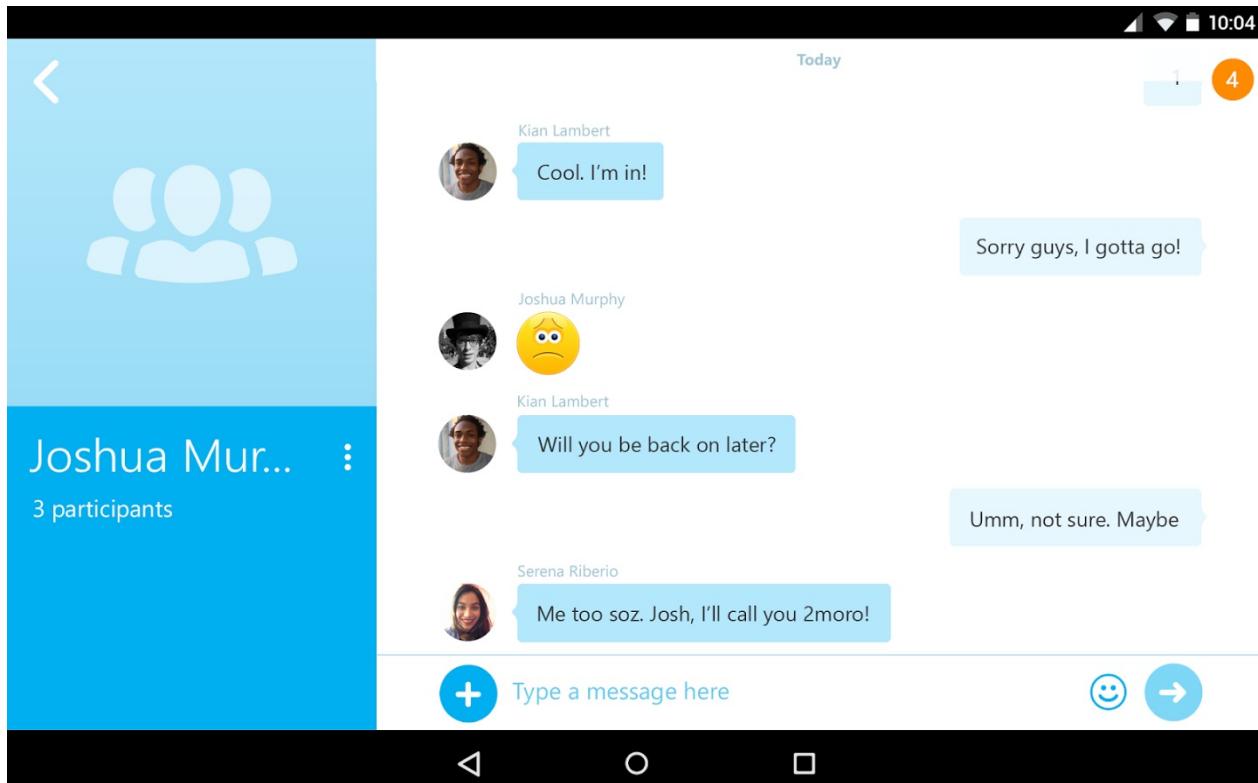


- **Công cụ quản lý rủi ro:** Công cụ này dùng để lẩn tránh sự xuất hiện lỗi, dự đoán lỗi hoặc theo dõi lỗi. Nó sử dụng một trong các phương pháp như mô phỏng hoặc cây quyết định. Công cụ **RiskNav** là một công cụ khá phổ biến để quản lý rủi ro. Nó mô tả không gian lỗi theo hai dạng bảng và đồ họa như hình dưới

Risk Space: <All Risks>									
Risk List Reports Risk Space Filters : Edit Defaults Default Filters Sort Field: Priority									
Show Details Hide Categories									
Risk ID	State	Name	Category	5x5 Color	Priority	Mitigation Status	Impact Date	Risk Manager	
MGT.001 Description	Open	Organizational Interfaces		◆ Red	High/ 0.89	<input type="checkbox"/> White (no plan) Mitigation Analysis	M 16 Sep 2008		
OPS.003 Description	Open	Ground Sampling Collection and Analysis	Operational; Subsystem; Technical	◆ Red	Issue/ 0.84	<input checked="" type="checkbox"/> Green Mitigation	M 19 Jul 2008	Landes, Maxine	
SE.016 Description	Proposed/Pending Review	Technology Readiness for Science Payload CIs	Programmatic; Technical	◆ Red	High/ 0.81	<input checked="" type="checkbox"/> Red Mitigation	M 16 Nov 2008	Landes, Maxine	
PROG.001 Description	Open/Needs Review	Stakeholder and Mission Partner Complexity	Programmatic	◆ Red	High/ 0.79	<input checked="" type="checkbox"/> Red Mitigation	M 02 Oct 2008	Landes, Maxine	
OPS.006 Description	Open	Balloon inflation	Operational; Subsystem	◆ Red	High/ 0.75	<input checked="" type="checkbox"/> Yellow Mitigation	07 Jul 2008	Ramirez, Diego	
MGT.002 Description	Open	WBS	Programmatic	◆ Red	High/ 0.74	<input type="checkbox"/> White (no status)	M 28 Aug 2008	Santos, Andrea	
MGT.003 Description	Proposed	IMS	Programmatic	◆ Yellow	High/ 0.72	<input type="checkbox"/> White (no plan)	M 27 Jul 2008		

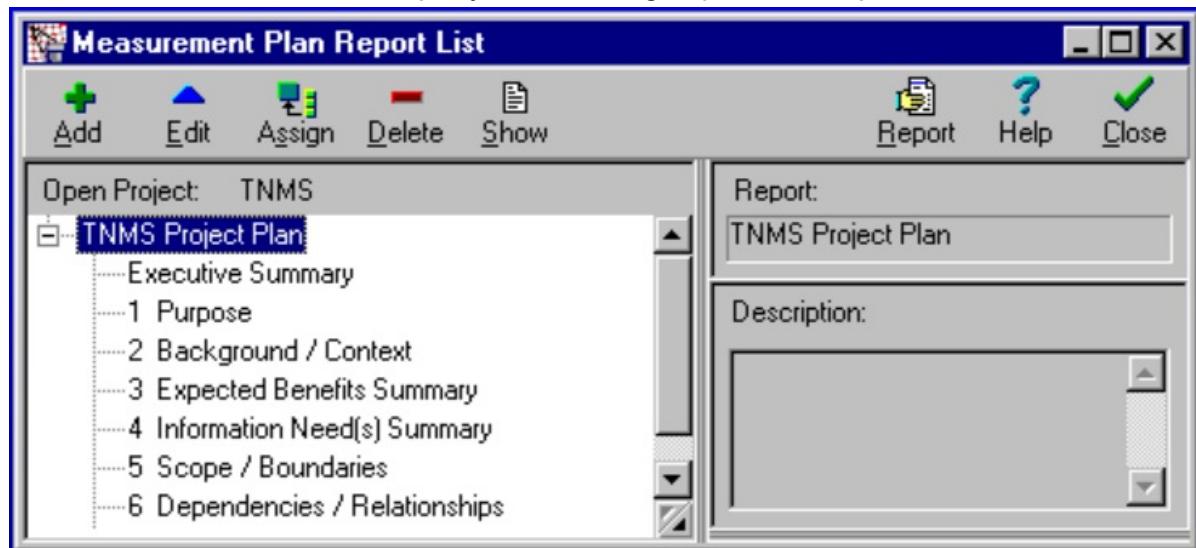
- **Công cụ tương tác:** Công cụ này giúp việc cung cấp thông tin kịp thời, thống nhất đến các đối tượng liên quan tới dự án. Những công cụ này có thể bao gồm những việc như thông báo email và quảng bá đến tất cả thành viên dự án cũng như những người có liên quan. Nó cũng bao gồm việc thông tin về thời gian của kế hoạch dự án, họp đứng, backlog...

Một ví dụ về công cụ tương tác Skype



- **Công cụ đánh giá:** Công cụ này hỗ trợ hoạt động liên quan tới chương trình đánh giá

phần mềm. Nó có rất ít công cụ tự động hỗ trợ việc này. Các công cụ đánh giá được sử dụng để thu thập, phân tích và báo cáo dữ liệu đánh giá dự án có thể dựa trên các bảng tính của các thành viên dự án hoặc của các nhân viên trong tổ chức Công cụ phổ biến dùng để hỗ trợ việc đánh giá dự án có thể kể tới là **PSM Insight**. Đây là một công cụ được tài trợ bởi bộ Tư Pháp Mỹ và vẫn đang tiếp tục được phát triển



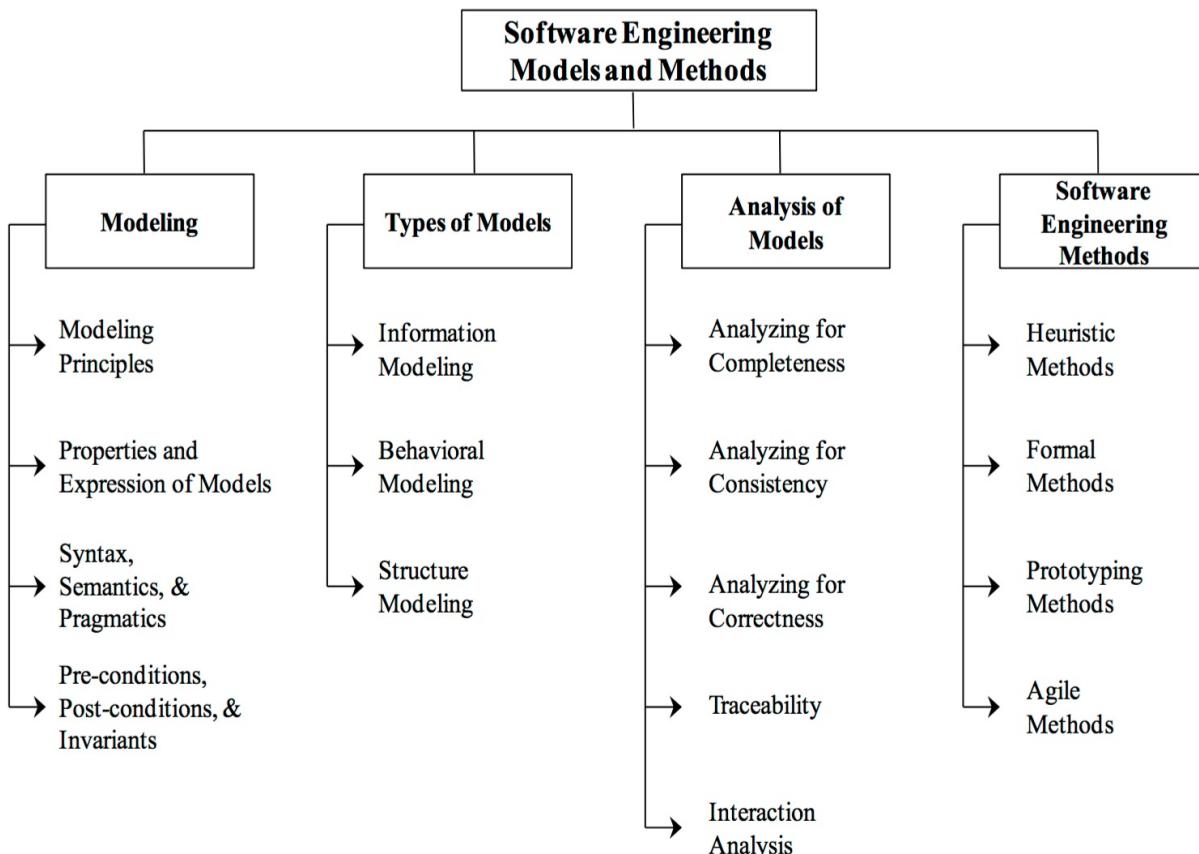
Chapter 9: Software Engineering Models and Methods

Các phương pháp và mô hình công nghệ phần mềm

1. Giới thiệu

Các phương pháp và mô hình công nghệ phần mềm đặt lên cấu trúc trong phần mềm với mục đích làm cho phần mềm hoạt động có hệ thống và vận hành ổn định. Sử dụng mô hình cung cấp cách tiếp cận đến việc giải quyết các vấn đề, ký pháp hoặc quá trình xây dựng và phân tích mô hình. Phương thức cung cấp cách tiếp cận đến những đặc tả hệ thống, thiết kế, xây dựng, kiểm thử và kiểm nghiệm của sản phẩm phần mềm cuối cùng và những sản phẩm có liên quan.

Trong chương này, chúng tôi sẽ đề cập đến bốn chủ đề chính như sau:



2. Mô hình hóa

Mô hình hóa phần mềm đang trở thành một kỹ thuật phổ biến giúp những kỹ sư phần mềm có thể hiểu, bố trí và giao tiếp các vấn đề yêu cầu của phần mềm với các bên liên quan. Các bên liên quan ở đây là những người hay những nhóm có quan tâm đến phần mềm (ví dụ như: người sử dụng, người mua hàng, người thiết kế, người phát triển, kỹ sư phần mềm, v.v.)

Hiện nay, có rất nhiều các ngôn ngữ mô hình hóa, ký pháp, kỹ thuật. Nhưng tất cả chúng đều có những nền tảng và những khái niệm chung. Dưới đây là những nền tảng cơ bản về những khái niệm chung đó.

2.1. Nguyên tắc Mô hình hóa

Mô hình hóa cung cấp cho các kỹ sư phần mềm cách một tiếp cận có tổ chức, có hệ thống để thể hiện những khía cạnh quan trọng của phần mềm đang được phát triển, tạo thuận lợi cho việc ra quyết định về những thành phần của phần mềm và giúp cho trao đổi thông tin với những bên liên quan trở nên dễ dàng hơn. Có ba nguyên tắc chung để mô hình hóa. Đó là:

- *Mô hình hóa những thành phần thiết yếu*: Một mô hình tốt sẽ là mô hình không thể hiện tất cả các thành phần, tính năng của phần mềm dưới mọi điều kiện có thể xảy ra. Mô hình hóa sẽ thường tập trung vào những thành phần quan trọng, trọng tâm. Việc làm này sẽ làm cho mô hình được quản lý dễ dàng hơn và trở nên hữu dụng.
- *Cung cấp dưới nhiều góc nhìn*: Mô hình hóa cung cấp các cách tiếp cận phần mềm bằng cách quy tắc để biểu hiện mô hình. Chúng ta có thể tiếp cận phần mềm dưới nhiều cách khác nhau, đó là: tiếp cận theo cấu trúc, tiếp cận theo hành vi, tiếp cận theo cách tổ chức,... Các thông tin thể hiện ở các cách tiếp cận đó có thể là ký pháp, từ vựng, phương thức hoặc công cụ,...
- *Hỗ trợ trao đổi thông tin hiệu quả*: Mô hình hóa được thể hiện bằng các từ vựng, ngữ nghĩa và các ngôn ngữ mô hình hóa khác nhau. Nhưng phải đảm bảo được khi sử dụng chúng, thì mô hình phải trở nên dễ dàng tiếp cận đối với những bên liên quan.

2.2. Tính chất và cách thể hiện mô hình

Có ba tính chất để mô tả các đặc điểm của mô hình, đó là:

- *Tính toàn vẹn*: thể hiện mức độ mà các yêu cầu phần mềm đã được thực thi và xác nhận bằng mô hình.
- *Tính nhất quán*: thể hiện mức độ của mô hình không tồn tại các trường hợp mâu thuẫn về yêu cầu, ràng buộc hoặc mô tả các thành phần bên trong phần mềm,...
- *Tính đầy đủ*: thể hiện mức độ đúng đắn của mô hình đối với các đặc tả thiết kế, yêu cầu

và không còn tồn tại các chỗ sai (free of defects).

Mô hình được xây dựng để đại diện cho các đội tượng thực tế, và những hành vi của chúng thể hiện cho cách thức phần mềm hoạt động dự kiến. Cách sơ cấp nhất để thể hiện các thành phần của một mô hình đó là sử dụng thực thể. Một thực thể là đại diện cho các sự vật cụ thể như: tiến trình (processor), cảm biến (sensor), robots. Các thực thể sẽ được liên kết với nhau bằng các đường thẳng liên kết, các ký pháp, hình ảnh. Cách tốt nhất để có thể thể hiện các mô hình đó là sử dụng hình ảnh gắn liền với mô hình đó. Như vậy, khi nhìn vào hình ảnh của mô hình, ta có thể biết ngay được ý nghĩa của mô hình là để làm gì.

2.3. Cú pháp, ngữ nghĩa và tính khả dụng.

Mô hình có thể dễ dàng bị hiểu nhầm một cách ngạc nhiên. Thực thể cho thấy rằng, một mô hình là một khái niệm trừu tượng với sự thiếu sót thông tin có thể dẫn người xem có cảm giác sai lệch của việc hoàn toàn hiểu rằng phần mềm là từ một mô hình duy nhất.

Rất khó khăn để có thể hiểu được ý nghĩa chính xác cấu trúc của một mô hình. Các ngôn ngữ mô hình hóa được định nghĩa bởi các quy tắc cú pháp và ngữ nghĩa.

Cú pháp

- Đối với ngôn ngữ văn bản, cú pháp được định nghĩa bằng cách sử dụng các ký hiệu ngữ pháp để định nghĩa các cấu trúc hợp lệ. Ví dụ: ta vẫn thường sử dụng ký hiệu BFS để định nghĩa cho giải thuật Breath First Search chẳng hạn.
- Đối với mô hình hóa bằng đồ họa, cú pháp được định nghĩa bằng cách sử dụng những mô hình hình ảnh để thể hiện, chúng được gọi là siêu mô hình (metamodels). Siêu mô hình này sẽ định nghĩa cách xây dựng lên được một mô hình hợp lệ.

Ngữ nghĩa

Ngữ nghĩa cho các ngôn ngữ mô hình hóa là việc chỉ ra ý nghĩa gắn liền với các thực thể và các mối quan hệ ở trong một mô hình.

Tính khả dụng

Như trong thực tế, lựa chọn ngôn ngữ mô hình hóa để có thể hiểu rõ nhất về ngữ nghĩa của mô hình phần mềm một cách cụ thể và cách mà ngôn ngữ mô hình hóa sử dụng để thể hiện các thực thể và mối quan hệ bên trong các mô hình đó. Ý nghĩa của mô hình vẫn được thể hiện ran gay cả khi thông tin không được đầy đủ. Như vậy, tính khả dụng của mô hình là mô hình cần phải thể hiện được ý nghĩa và đảm bảo sự trao đổi thông tin hiệu quả giữa các kỹ sư phần mềm trong quá trình phát triển

2.4. Tiền điều kiện, hậu điều kiện và khả năng bắt biến

Khi mô hình hóa các hàm hoặc phương thức của một phần mềm, các kỹ sư phần mềm thường bắt đầu với tập các giả định về trạng thái của phần mềm trước, trong và sau khi hàm/phương thức được thực thi. Những giả định này là cần thiết để có thể vận hành đúng chức năng của các hàm/phương thức và chúng được nhóm lại thành ba nhóm đó là: **Tiền điều kiện, hậu điều kiện và bất biến đổi**.

Tiền điều kiện: là tập các điều kiện cần phải được đảm bảo trước khi thực thi các hàm/phương thức để nhận được kết quả đúng. Nếu các điều kiện này không được xem xét trước khi thực thi hàm/phương thức thì sẽ gây ra kết quả không chính xác.

Hậu điều kiện: là tập các điều kiện sẽ đạt được sau khi các hàm/phương thức được thực thi thành công. Hậu điều kiện sẽ thể hiện sự thay đổi trạng thái của phần mềm về các tham số, giá trị,...

Bất biến: là tập các điều kiện không bị thay đổi ngay cả trước và sau khi thực thi hàm/phương thức.

3. Các loại mô hình

Một mô hình (model) tiêu biểu bao gồm tập các mô hình con. Mỗi mô hình con là một mô tả bộ phận và được tạo ra với một mục đích cụ thể, rõ ràng. Nó có thể bao gồm một hay nhiều biểu đồ (diagrams). Tập các mô hình con được thiết kế bằng một hay nhiều loại ngôn ngữ mô hình hóa. Ngôn ngữ mô hình hóa **UML** (Unified Modeling Language) cho phép thiết kế nhiều loại biểu đồ của mô hình. Sử dụng những biểu đồ này cùng với ngôn ngữ mô hình hóa mang tới ba loại mô hình phổ biến: mô hình thông tin (information models), mô hình hành vi (behavioral models), mô hình cấu trúc (structure models).

3.1 Mô hình thông tin

Mô hình thông tin cung cấp một góc nhìn tập trung vào dữ liệu và thông tin. Một mô hình thông tin là một thể hiện trừu tượng mà xác định và định nghĩa tập các khái niệm, đặc tính, mối quan hệ và ràng buộc trên thực thể dữ liệu. Mô hình thông tin ngữ nghĩa hay mô hình thông tin khái niệm thường được dùng để cung cấp vài hình thức và khung cảnh tới phần mềm đang được mô hình hóa như góc nhìn từ mặt vấn đề mà không quan tâm tới việc thực tế mô hình này được ánh xạ tới việc cài đặt phần mềm. Mô hình thông tin ngữ nghĩa hoặc mô hình thông tin khái niệm là một trừu tượng hóa và chỉ gồm định nghĩa, thuộc tính, quan hệ và ràng buộc cần để định nghĩa góc nhìn thật của thông tin. Sau sự biến đổi của mô hình thông tin ngữ nghĩa hoặc mô hình thông tin khái niệm là sự xây dựng mô hình dữ liệu logic và sau đó và mô hình dữ liệu vật lý như sự cài đặt trong phần mềm.

3.2 Mô hình hành vi

Mô hình hành vi xác định và định nghĩa chức năng của phần mềm đang được mô hình hóa. Mô hình hành vi thường gồm ba dạng cơ bản: máy trạng thái, mô hình dòng điều khiển, mô hình dòng dữ liệu. Máy trạng thái cung cấp một mô hình phần mềm như tập của các trạng thái, sự kiện và phép chuyển đổi đã được định nghĩa. Phần mềm chuyển đổi từ một trạng thái tới trạng thái tiếp bởi một sự kiện chắc chắn hay không chắc chắn gây ra mà xuất hiện trong môi trường đã được mô hình. Mô hình dòng điều khiển thể hiện làm sao mà một chuỗi các sự kiện làm cho quá trình được hoạt động hay ngừng hoạt động. Hình vi trong dòng dữ liệu cụ thể là một chuỗi các bước mà dữ liệu chuyển qua quá trình hướng nơi lưu trữ dữ liệu.

3.3 Mô hình cấu trúc

Mô hình cấu trúc thể hiện cấu tạo vật lý hay cấu tạo logic của phần mềm từ nhiều phần của phần mềm. Mô hình hóa cấu trúc thiết lập nên một bao ngoài giữa phần mềm đang được cài đặt hay mô hình với môi trường mà nó thực thi. Vài cấu trúc xây dựng cụ thể được sử dụng trong việc mô hình hóa cấu trúc là tổng hợp, phân tích, tổng quát hóa, cụ thể hóa thực thể; xác định các mối quan hệ và yếu tố liên quan giữa các thực thể; và định nghĩa của quy trình hay giao diện chức năng. Biểu đồ cấu trúc cung cấp bởi UML cho việc mô hình hóa cấu trúc gồm: biểu đồ lớp, biểu đồ thành phần, biểu đồ đối tượng, biểu đồ triển khai, biểu đồ gói (packaging diagram).

4. Phân tích mô hình

Quá trình phát triển mô hình tạo điều kiện cho kỹ sư phần mềm một cơ hội để học, suy luận và hiểu cấu trúc, chức năng, cách sử dụng chức năng và xem như sự liên kết với phần mềm. Phân tích các mô hình đã xây dựng là cần thiết để đảm bảo những mô hình này hoàn thiện, nhất quán và chính xác đủ để phục vụ mục đích của nó cho người yêu cầu. Tiếp theo là mô tả ngắn về các kỹ thuật phân tích thường dùng với mô hình phần mềm để đảm bảo kỹ sư phần mềm và một số người yêu cầu liên quan có được những lợi ích phù hợp từ việc phát triển và sử dụng phần mềm.

4.1 Phân tích sự hoàn thiện

Để có được phần mềm hoàn toàn đáp ứng nhu cầu của người yêu cầu, tính hoàn thiện là cực quan trọng từ quá trình lấy yêu cầu tới việc cài đặt mã nguồn. Tính hoàn thiện là mức độ mà tất cả các yêu cầu đã được chỉ ra được cài đặt và kiểm chứng. Mô hình có thể kiểm tra sự hoàn thiện bởi các công cụ mô hình mà sử dụng kỹ thuật như phân tích cấu trúc và phân tích khả năng đạt tới không gian trạng thái (đảm bảo tất cả các đường đi trong mô hình trạng thái đều đạt được bởi vài tập đầu vào chính xác); mô hình cũng có thể được kiểm tra sự hoàn thiện bằng tay bởi việc sử dụng kỹ thuật duyệt (inspection) hay các kỹ

thuật xem xét lại (review) khác. Lỗi và cảnh báo sinh ra bởi những công cụ phân tích này và được tìm thấy bởi việc duyệt hay xem xét lại thể hiện xác suất cần hành động sửa lại để đảm bảo sự hoàn thiện của mô hình.

4.2 Phân tích sự nhất quán

Sự nhất quán là mức độ mà mô hình chứa các xung đột về yêu cầu, quyết định, ràng buộc, chức năng hay mô tả các thành phần. Điện hình, kiểm tra sự nhất quán là sử dụng một chức năng phân tích tự động với các công cụ mô hình hóa, mô hình cũng có thể được kiểm tra bằng tay sự nhất quán bằng kỹ thuật duyệt hay các kỹ thuật xem xét khác. Cũng như với sự hoàn thiện, lỗi và cảnh báo được sinh ra bởi những công cụ phân tích này và tìm thấy khi duyệt hay xem xét thể hiện cần có hành động sửa chữa những lỗi này.

4.3 Phân tích sự đúng đắn

Sự đúng đắn là mức độ mà một mô hình thỏa mãn tài liệu yêu cầu phần mềm và tài liệu thiết kế, đồng thời cũng không có lỗi và trên hết là đáp ứng được nhu cầu của khách hàng. Phân tích sự đúng đắn gồm việc kiểm tra sự đúng đắn về ngữ pháp của mô hình (là sự sử dụng ngôn ngữ mô hình hóa đúng ngữ pháp và cấu trúc) và kiểm tra sự đúng đắn về ngữ nghĩa của mô hình (là sử dụng ngôn ngữ mô hình hóa để thể hiện đúng đắn ý nghĩa của đối tượng đang được mô hình hóa). Để phân tích mô hình cho cả sự đúng đắn về ngữ pháp và ngữ nghĩa, có thể sử dụng các công cụ tự động (như là sử dụng công cụ mô hình để kiểm tra sự đúng đắn về ngữ pháp của mô hình) hay bằng tay (sử dụng sự kỹ thuật duyệt hay các kỹ thuật xem xét khác) để tìm ra các lỗi có thể và thực hiện sửa lỗi này trước khi phần mềm được đưa ra thực tế sử dụng.

4.4 Khả năng lằn vết (traceability)

Phát triển phần mềm thường gồm việc sử dụng, tạo ra và sửa chữa rất nhiều sản phẩm như tài liệu kế hoạch, tài liệu về quy trình sử dụng, yêu cầu phần mềm, biểu đồ, thiết kế, mã giả, bản viết tay và các công cụ sinh mã nguồn tự động, các test case tự động hay bằng tay cùng các báo cáo, file và dữ liệu. Những sản phẩm này có thể quan hệ, liên kết qua nhiều mối quan hệ phụ thuộc (ví dụ như sử dụng, cài đặt và kiểm thử). Khi phần mềm được phát triển, quản lý, bảo trì hay mở rộng, ta cần sự ánh xạ và điều khiển những mối quan hệ có thể phải lằn vết này để thể hiện yêu cầu phần mềm là nhất quán với mô hình phần mềm và nhiều tài liệu khác. Sử dụng khả năng lằn vết thường cải thiện sự quản lý các sản phẩm phần mềm và qui trình quản lý chất lượng phần mềm. Nó cũng cung cấp sự đảm bảo cho khách hàng rằng tất cả yêu cầu đều được thỏa mãn. Khả năng lằn vết cho phép phân tích sự thay đổi một khi phần mềm đã được triển khai sử dụng thực tế, vì các mối quan hệ giữa các sản phẩm phần mềm có thể dễ dàng lằn ngược để đánh giá mức độ ảnh hưởng. Công cụ mô hình hóa thường cung cấp vài phương tiện tự động hay bằng tay để chỉ rõ và quản lý khả năng lằn vết sự liên kết giữa yêu cầu, thiết kế, mã nguồn và/hoặc các thực thể test như là sự thể hiện trong mô hình và các sản phẩm phần mềm khác.

4.5 Phân tích sự tương tác

Phân tích sự tương tác tập trung vào quan hệ giao tiếp hoặc quan hệ luồng điều khiển giữa các thực thể để hoàn thiện một nhiệm vụ hay một chức năng cụ thể trong mô hình phần mềm. Sự phân tích này đánh giá các hành vi động của tương tác giữa các phần khác nhau của mô hình phần mềm, gồm các tầng phần mềm khác (như lớp hệ điều hành, lớp giữa, lớp ứng dụng). Trong vài ứng dụng phần mềm, việc đánh giá tương tác giữa phần mềm ứng dụng trên máy tính và giao diện người dùng của phần mềm cũng là quan trọng. Vài môi trường mô hình hóa phần mềm cung cấp sự mô phỏng các nhân tố để nghiên cứu các mặt của các hành vi động của phần mềm được mô hình hóa. Qua bước mô phỏng sẽ cung cấp một lựa chọn phân tích cho kỹ sư phần mềm để xem lại thiết kế tương tác và đảm bảo rằng các phần khác nhau của phần mềm làm việc được với nhau và cung cấp những chức năng mong muốn.

5. Các phương pháp phát triển phần mềm

Các phương pháp phát triển phần mềm cung cấp cách tiếp cận hệ thống và tổ chức để phát triển một phần mềm hoàn thiện. Có rất nhiều phương pháp phát triển phần mềm hiện nay, điều quan trọng là các kỹ sư phần mềm phải chọn một hoặc một số phương pháp phù hợp cho nhiệm vụ phát triển cụ thể. Sự lựa chọn này sẽ có ảnh hưởng lớn tới sự thành công của dự án. Việc sử dụng những phương pháp phát triển phần mềm kèm theo các công cụ và nhân sự có kỹ năng cho phép các kỹ sư phần mềm thể hiện trực quan chi tiết của phần mềm cần hoàn thiện, từ đó chuyển hóa thành code và dữ liệu.

Một số phương pháp phát triển phần mềm chọn lọc sẽ được thảo luận dưới đây. Các chủ đề được tổ chức thành 4 nhóm: Nhóm các phương pháp suy nghiệm, Nhóm các phương pháp hình thức, Nhóm các phương pháp tạo bản mẫu, Nhóm các phương pháp Agile.

5.1. Các phương pháp suy nghiệm

Các phương pháp suy nghiệm là những phương pháp phát triển phần mềm dựa vào kinh nghiệm, gồm những phương pháp đã và đang được sử dụng rộng rãi thực tế trong ngành công nghiệp phần mềm.

Nhóm phương pháp này gồm 3 mục lớn: các phương pháp phân tích và thiết kế theo cấu trúc, các phương pháp mô hình hóa dữ liệu và các phương pháp phân tích thiết kế hướng đối tượng.

- *Các phương pháp phân tích và thiết kế theo cấu trúc:* Mô hình phần mềm được phát triển chủ yếu từ góc nhìn chức năng hay hành vi, bắt đầu từ cái nhìn mức độ cao của phần mềm, sau đó dần được phân tách thành các phần nhỏ hơn hoặc tinh lọc thông

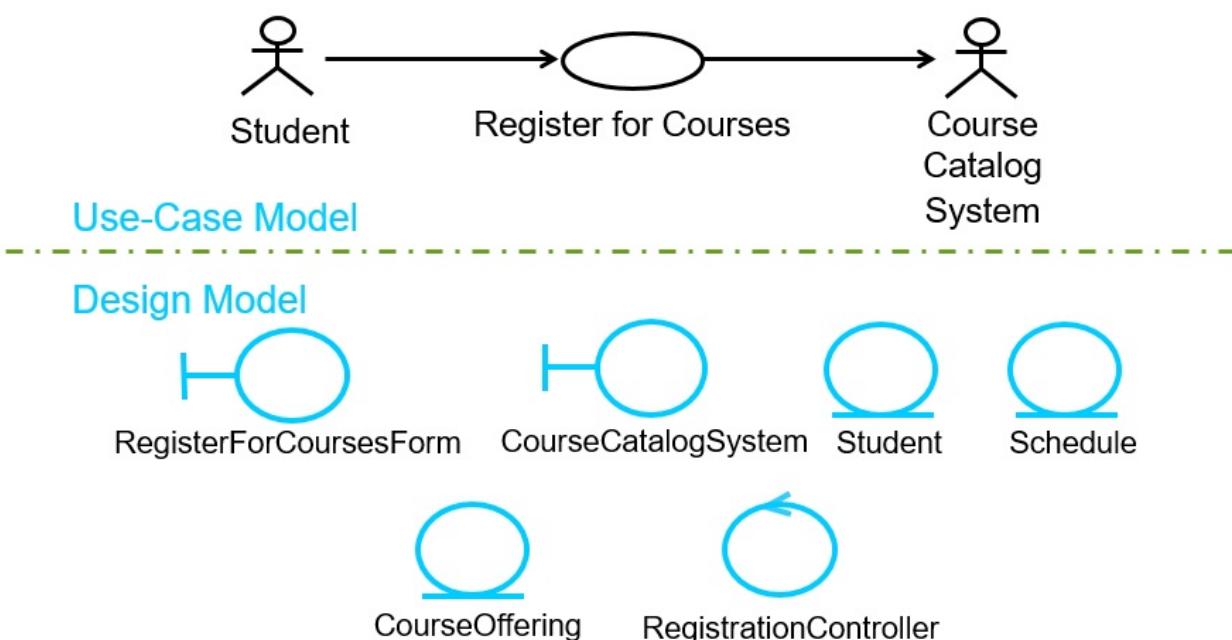
qua các bước thiết kế chi tiết hơn. Thiết kế chi tiết cuối cùng sẽ thể hiện những đặc tả hoặc chi tiết ở mức rất cụ thể của phần mềm. Những thành phần này sau đó sẽ được lập trình, build, kiểm thử và kiểm tra.

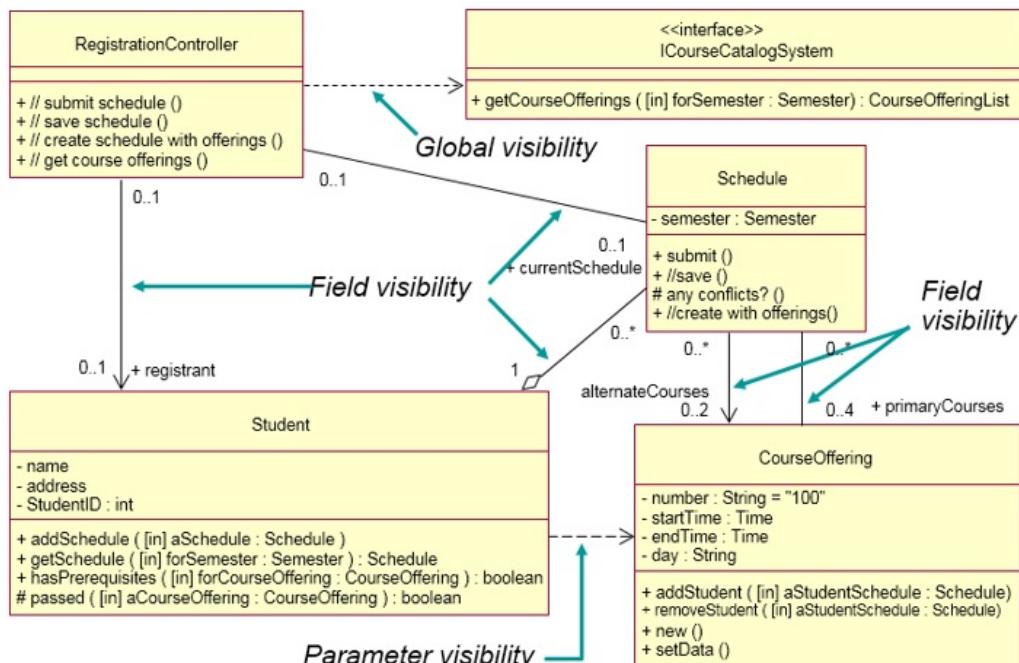
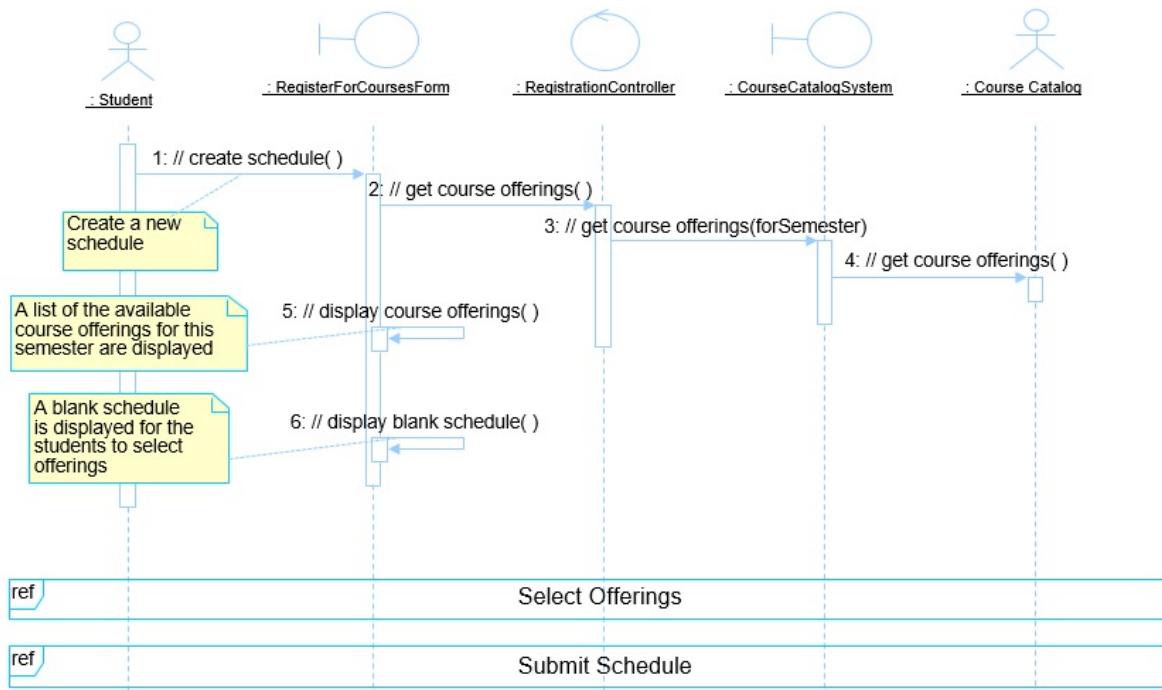
- *Phương pháp mô hình hóa dữ liệu:* Mô hình dữ liệu được xây dựng từ góc nhìn dữ liệu hoặc thông tin được sử dụng. Các bảng và các mối quan hệ dữ liệu định nghĩa ra mô hình dữ liệu. Phương pháp mô hình hóa dữ liệu này được sử dụng trong việc định nghĩa và phân tích yêu cầu dữ liệu, hỗ trợ thiết kế cơ sở dữ liệu hay kho dữ liệu, thường thấy trong những phần mềm mà dữ liệu được quản lý như một tài nguyên hệ thống nghiệp vụ.
- *Phương pháp phân tích thiết kế hướng đối tượng:* Mô hình hướng đối tượng được thể hiện như một tập hợp các đối tượng, chứa dữ liệu và các mối quan hệ, tương tác với nhau thông qua các phương thức. Các đối tượng có thể là các vật thật hoặc ảo. Mô hình phần mềm được xây dựng theo phương pháp này sử dụng các biểu đồ để thể hiện những góc nhìn chọn lọc của phần mềm. Quá trình tinh lọc dần những mô hình tổng quát sẽ tạo ra bản thiết kế chi tiết cho phần mềm. Thiết kế chi tiết sau đó sẽ được lập trình và đóng gói thành những sản phẩm cuối cùng, sử dụng cho việc phát hành và triển khai.

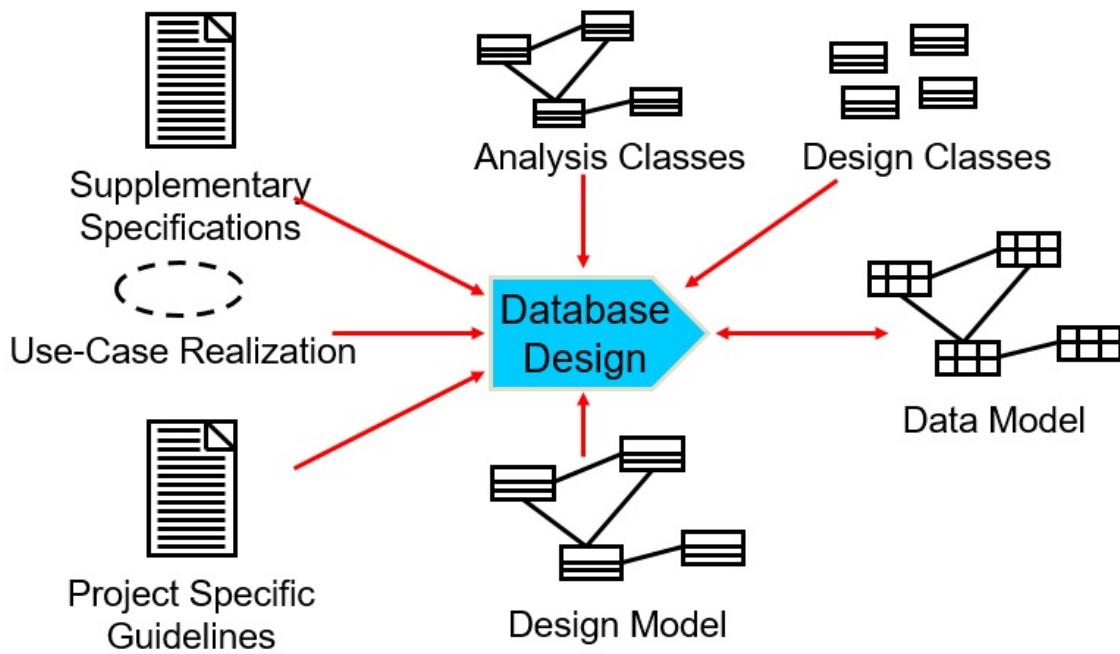
Ví dụ: : phương pháp phân tích thiết kế hướng đối tượng

Yêu cầu: thiết kế chức năng đăng ký môn học

Các biểu đồ thiết kế:







5.2. Các phương pháp hình thức

Các phương pháp hình thức là những phương pháp phát triển phần mềm trong đó việc đặc tả, phát triển và kiểm tra phần mềm được thực hiện thông qua việc ứng dụng ngôn ngữ và ký hiệu toán học một cách chặt chẽ. Thông qua việc sử dụng ngôn ngữ đặc tả, mô hình phần mềm có thể được kiểm tra về tính nhất quán, tính hoàn thiện, tính đúng đắn một cách hệ thống và tự động hoặc bán tự động. Chủ đề này thường liên quan tới mục phân tích hình thức trong khi tìm hiểu yêu cầu phần mềm.

Trong mục này, chúng ta sẽ tìm hiểu về Ngôn ngữ đặc tả, Tinh lọc và chuyển hóa chương trình, Kiểm tra hình thức và Suy diễn logic.

- **Ngôn ngữ đặc tả:** Ngôn ngữ hình thức cung cấp cơ sở toán học cho phương pháp hình thức. Các ngôn ngữ hình thức là ngôn ngữ máy tính bậc cao, ở dạng hình thức, được sử dụng trong suốt giai đoạn đặc tả, phân tích yêu cầu và thiết kế để miêu tả hành vi đầu vào/đầu ra. Ngôn ngữ hình thức không phải là ngôn ngữ có thể chạy trực tiếp, chúng thường bao gồm những ký hiệu, cấu trúc và ngữ nghĩa cho các ký hiệu đó và một tập hợp các mối quan hệ cho phép của các đối tượng.
- **Tinh lọc và chuyển hóa chương trình:** Tinh lọc chương trình là một quá trình tạo ra đặc tả mức thấp, mức chi tiết hơn thông qua một chuỗi biến đổi. Nó là những chuỗi biến đổi liên tục mà từ đó những kỹ sư phần mềm có thể chuyển hóa thành dạng chạy được của chương trình. Những đặc tả có thể sẽ được tinh lọc, thêm chi tiết cho tới khi mô hình có thể được công thức hóa bằng ngôn ngữ lập trình thế hệ thứ 3 hoặc bằng thành phần chạy được trong ngôn ngữ hình thức được chọn. Việc tinh lọc đặc tả được làm

bằng việc định nghĩa những đặc tả với các thuộc tính ngữ nghĩa chính xác; những đặc tả không chỉ phải dựng lên mối quan hệ giữa những thực thể mà còn phải thể hiện ý nghĩa chính xác của những mối quan hệ và quá trình hoạt động đó.

- *Kiểm tra hình thức*: Kiểm tra mô hình là một phương pháp kiểm tra hình thức; nó thường liên quan tới việc phân tích trạng thái để thể hiện rằng thiết kế phần mềm có hoặc đảm bảo những thuộc tính nhất định. Một ví dụ của kiểm tra mô hình là việc phân tích xem hành vi của chương trình có đúng không trong trường hợp đan xen các sự kiện hoặc tin nhắn đến. Việc sử dụng kiểm tra hình thức đòi hỏi mô hình phần mềm và môi trường hoạt động được đặc tả một cách chính xác; mô hình này thường được biểu diễn dưới dạng một máy trạng thái hoặc máy tự động.
- *Suy diễn logic*: Suy diễn logic là một phương pháp thiết kế phần mềm liên quan tới việc đặc tả tiền điều kiện, hậu điều kiện quanh một khối thiết kế, sử dụng logic toán học, phát triển những bằng chứng để thể hiện rằng những tiền điều kiện và hậu điều kiện đó đảm bảo cho tất cả các trường hợp đầu vào. Nó cung cấp cho các kỹ sư phần mềm một cách để dự đoán hành vi mà không cần phải chạy phần mềm.

Ví dụ: ngôn ngữ đặc tả Z-notation

Yêu cầu:

- Initially, the time is midnight, the bell is off, and the alarm is disabled
- Whenever the current time is the same as the alarm time and the alarm is enabled, the bell starts ringing
- The alarm time can be set at any time
- Only when the alarm is enabled can it be disabled
- If the alarm is disabled while the bell is ringing, the bell stops ringing
- Resetting the clock and enabling or disabling the alarm are considered to be done instantaneously

Cách thức mô tả:

$\text{BellStatus} : \{\text{quiet}, \text{ringing}\}$, $\text{AlarmStatus} : \{\text{disabled}, \text{enabled}\}$

Clock

$\text{time}, \text{alarm_time} : N$
 $\text{bell} : \text{BellStatus}$
 $\text{alarm} : \text{AlarmStatus}$

InitClock

ΔClock

$(\text{time}' = \text{midnight}) \wedge (\text{bell}' = \text{quiet}) \wedge$
 $(\text{alarm}' = \text{disabled})$

Tick

ΔClock

$(\text{time}' = \text{succ}(\text{time}))$
 $(\text{alarm_time}' = \text{time}') \wedge (\text{alarm}' = \text{enabled}) \implies (\text{bell}' = \text{ringing})$
 $(\sim ((\text{alarm_time}' = \text{time}') \wedge (\text{alarm}' = \text{enabled})) \implies (\text{bell}' = \text{bell})$
 $(\text{alarm}' = \text{alarm}) \wedge (\text{alarm_time}' = \text{alarm_time})$

SetAlarmTime

ΔClock

$\text{new_time?} : N$

$(\text{alarm_time}' = \text{new_time?})$
 $((\text{alarm_time}' = \text{time}') \wedge (\text{alarm}' = \text{enabled}) \implies (\text{bell}' = \text{ringing})$
 $(\sim ((\text{alarm_time}' = \text{time}') \wedge (\text{alarm}' = \text{enabled})) \implies (\text{bell}' = \text{bell})$
 $(\text{time}' = \text{time}) \wedge (\text{alarm}' = \text{alarm})$

EnableAlarm $\Delta Clock$

$$\begin{aligned}
 (\text{alarm} = \text{disabled}) &\Rightarrow (\text{alarm}' = \text{enabled}) \wedge \\
 ((\text{alarm_time}' = \text{time}') &\implies (\text{bell}' = \text{ringing}) \wedge \\
 (\sim (\text{alarm_time}' = \text{time}')) &\implies (\text{bell}' = \text{bell})) \wedge \\
 (\text{time}' = \text{time}) \wedge (\text{alarm_time}' = \text{alarm_time})
 \end{aligned}$$

DisableAlarm $\Delta Clock$

$$\begin{aligned}
 (\text{alarm} = \text{enabled}) &\Rightarrow (\text{alarm}' = \text{disabled}) \wedge (\text{bell}' = \text{quiet}) \wedge \\
 (\text{time}' = \text{time}) \wedge (\text{alarm}' = \text{alarm}) \wedge (\text{alarm_time}' = \text{alarm_time})
 \end{aligned}$$

5.3. Các phương pháp tạo bản mẫu

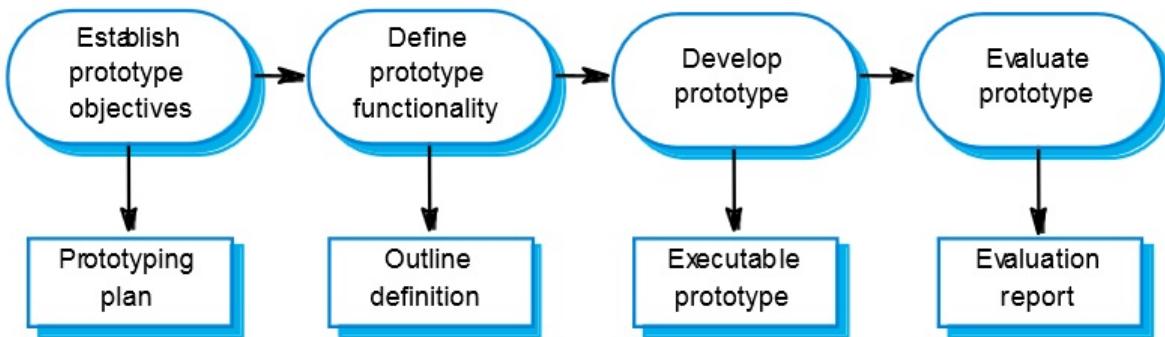
Tạo bản mẫu phần mềm là một hoạt động tạo ra những phiên bản ít chức năng và chưa hoàn thiện của ứng dụng phần mềm, thường được sử dụng để thử nghiệm những tính năng mới xác định, lấy yêu cầu phần mềm hoặc giao diện người dùng, hoặc cao hơn là thăm dò yêu cầu phần mềm, thiết kế phần mềm hoặc những lựa chọn cài đặt và những thông tin hữu ích khác về phần mềm. Kỹ sư phần mềm lựa chọn phương pháp tạo bản mẫu để tìm hiểu những khía cạnh hoặc những thành phần còn chưa được hiểu rõ của phần mềm đầu tiên; cách tiếp cận này ngược với việc phát triển thông thường là phát triển những thành phần đã rõ trước tiên. Về cơ bản, những sản phẩm mẫu sẽ không trở thành sản phẩm phần mềm cuối được nếu không được làm lại hoặc cải thiện một cách đáng kể.

Phần này sẽ thảo luận về các phong cách, mục tiêu và kỹ thuật đánh giá bản mẫu một cách ngắn gọn.

- *Phong cách tạo bản mẫu:* Có nhiều cách tiếp cận để phát triển bản mẫu. Các bản mẫu có thể được phát triển như là code dùng một lần hoặc sản phẩm giấy, như là một sự tiến hóa từ một bản thiết kế đang sử dụng, hoặc như là một đặc tả chương trình có thể chạy được. Những quá trình quản lý vòng đời bản mẫu khác nhau được sử dụng cho các phong cách bản mẫu khác nhau. Việc lựa chọn phong cách bản mẫu có thể dự vào loại kết quả mà dự án cần, chất lượng của kết quả hoặc độ cấp bách của kết quả.
- *Mục tiêu tạo bản mẫu:* Mục tiêu của hoạt động tạo bản mẫu là để tạo ra sản phẩm cụ thể đúng với yêu cầu. Những ví dụ về mục tiêu tạo bản mẫu bao gồm: đặc tả yêu cầu, các thành phần thiết kế kiến trúc, một thuật toán, hoặc một giao diện người dùng.

- **Kỹ thuật đánh giá bản mẫu:** Một bản mẫu có thể được sử dụng hoặc đánh giá theo nhiều cách bởi các kỹ sư phần mềm hoặc những người tham gia dự án khác, theo hướng chủ yếu dựa vào lý do dẫn đến việc làm bản mẫu. Bản mẫu cũng có thể được đánh giá và kiểm thử dựa trên phần mềm đã được cài đặt thật hoặc dựa trên tập hợp những yêu cầu đích.

Mô hình phát triển bản mẫu:



5.4. Các phương pháp Agile

Những phương pháp Agile được tạo ra vào những năm 1990 từ nhu cầu cần giảm những công việc nặng nhọc tại thời điểm bắt đầu dự án của các phương pháp dựa vào kế hoạch vốn được sử dụng trong những dự án phát triển phần mềm quy mô lớn. Các phương pháp Agile được xem như là những phương pháp nhẹ nhàng, đơn giản với đặc điểm bao gồm những vòng phát triển ngắn, lặp lại, những đội tự tổ chức, thiết kế đơn giản hơn, tổ chức code lại, phát triển theo hướng kiểm thử, có sự tham gia thường xuyên của khách hàng và nhấn mạnh vào việc tạo ra những sản phẩm làm việc có thể trình diễn được sau mỗi vòng phát triển.

Có nhiều phương pháp agile được sử dụng hiện nay, trong đó những phương pháp Rapid Application Development (RAD), eXtreme Programming (XP), Scrum và Feature-Driven Development (FDD) là phổ biến hơn cả.

- **RAD:** Phương pháp phát triển phần mềm Rapid được sử dụng chủ yếu trong việc phát triển ứng dụng hệ thống nghiệp vụ, nặng về dữ liệu. Với những công cụ phát triển cơ sở dữ liệu mục đích đặc biệt, những nhà kỹ sư phần mềm có thể phát triển, kiểm thử và triển khai những ứng dụng nghiệp vụ mới hoặc sửa đổi một cách nhanh chóng.
- **XP:** Cách tiếp cận này sử dụng những câu chuyện hoặc những bối cảnh cho những yêu cầu phần mềm, phát triển những bài kiểm thử trước, có sự liên hệ trực tiếp của khách hàng với đội phát triển, sử dụng lập trình theo cặp và cung cấp việc tổ chức code lại và tích hợp một cách liên tục. Những câu chuyện được phân tách nhỏ thành những nhiệm vụ, sắp xếp độ ưu tiên, ước lượng thời gian, phát triển và kiểm thử. Mỗi vòng

phát triển phần mềm được kiểm thử với những bài test thủ công hoặc tự động; một vòng phát triển có thể được phát hành thường xuyên, ví dụ như một vài tuần một lần hoặc gần như vậy.

- **Scrum:** Cách tiếp cận agile này là thân thiện với quản lý dự án hơn cả. Người quản lý Scrum quản lý những hoạt động trong các vòng phát triển dự án; mỗi vòng phát triển được gọi là một sprint và kéo dài dưới 30 ngày. Một danh sách PBI được phát triển mà từ đó những nhiệm vụ được xác nhận, định nghĩa, sắp xếp mức độ ưu tiên và ước lượng thời gian. Mỗi phiên bản làm việc của phần mềm được kiểm thử và phát hành sau mỗi vòng phát triển. Phương pháp này sử dụng những buổi họp hàng ngày để đảm bảo công việc tiến hành đúng như kế hoạch.
- **FDD:** Đây là cách tiếp cận phát triển phần mềm ngắn, theo vòng phát triển, hướng mô hình gồm 5 giai đoạn:
 - Phát triển mô hình sản phẩm để giới hạn miền phát triển
 - Tạo ra danh sách những nhu cầu hay tính năng cần phát triển
 - Xây dựng lên kế hoạch phát triển các tính năng
 - Phát triển thiết kế cho những tính năng theo từng vòng phát triển cụ thể
 - Lập trình, kiểm thử và tích hợp tính năng

FDD vừa tương tự với phương pháp phát triển phần mềm gia tăng (tức phương pháp phát triển phần mềm gồm nhiều vòng phát triển nhỏ, mỗi vòng phát triển thêm các tính năng mới cho sản phẩm) vừa tương tự với phương pháp phát triển phần mềm XP, ngoại trừ việc nhiệm vụ được giao tới từng thành viên chứ không phải từng đội. FDD nhấn mạnh vào cách tiếp cận theo kiến trúc tổng thể của phần mềm, thúc đẩy việc xây dựng tính năng đúng ngay lần đầu hơn là tổ chức lại code thường xuyên.

Có nhiều dạng khác nhau của phương pháp agile trên sách vở cũng như thực tế. Cũng cần chú ý rằng luôn có nhu cầu cho những phương pháp phát triển phần mềm dựa vào kế hoạch. Ngoài ra, có những phương pháp mới được tạo ra từ việc kết hợp phương pháp Agile và phương pháp dựa vào kế hoạch: những người sử dụng cố gắng cân bằng những tính chất tốt nhất trong cả 2 phương pháp này để phục vụ nhu cầu nghiệp vụ tổ chức.

Chapter 11: Chuyên nghiệp trong kỹ nghệ phần

1. Yếu tố chuyên nghiệp

Một kỹ sư phần mềm thể hiện tính chuyên nghiệp thông qua việc tuân thủ các quy tắc đạo đức, ứng xử nghề nghiệp tới các chuẩn và thủ tục được thiết lập bởi cộng đồng chuyên môn. Cộng đồng chuyên môn thường được đại diện bởi một hoặc nhiều tổ chức chuyên môn (hội, nhóm nghề nghiệp). Các tổ chức này công bố các chuẩn về đạo đức và quy tắc ứng xử nghề nghiệp được biết như là các tiêu chuẩn gia nhập cộng đồng. Các tiêu chí này là cơ sở cho việc các hoạt động công nhận, cấp giấy phép và được sử dụng như một thước đo năng lực kỹ thuật.

Một kỹ sư phần mềm thể hiện tính chuyên nghiệp thông qua việc tuân thủ các quy tắc đạo đức, ứng xử nghề nghiệp tới các chuẩn và thủ tục được thiết lập bởi cộng đồng chuyên môn. Cộng đồng chuyên môn thường được đại diện bởi một hoặc nhiều tổ chức chuyên môn (hội, nhóm nghề nghiệp). Các tổ chức này công bố các chuẩn về đạo đức và quy tắc ứng xử nghề nghiệp được biết như là các tiêu chuẩn ra nhập cộng đồng. Các tiêu chí này là cơ sở cho việc các hoạt động công nhận, cấp giấy phép và được sử dụng như một thước đo năng lực kỹ thuật.

1.1. Kiểm định, chứng nhận và giấy phép

1.1.1. Kiểm định

Kiểm định là quá trình chứng nhận năng lực, thẩm quyền hoặc sự tin cậy của một tổ chức. Các chương trình kiểm định được đảm bảo tuân thủ các tiêu chuẩn cụ thể và duy trì chất lượng nhất định. Ở nhiều nước, các kỹ sư tiếp nhận kiến thức thông qua việc hoàn thành một khoá đào tạo kiểm định. Kiểm định kỹ thuật thường được thực hiện bởi các tổ chức chính phủ như bộ giáo dục, ví dụ Trung Quốc, Pháp, Đức, Isarel, Ý và Nga. Ở Việt Nam, để hoàn thành chương trình kiểm định, các kỹ sư phải hoàn thành các khoá học tại các đơn vị đào tạo thuộc bộ giáo dục. Tuy nhiên, ở một số quốc gia khác, quá trình kiểm định tách rời khỏi chính phủ và được thực hiện bởi các hiệp hội tư nhân. Một ví dụ điển hình như ở Hoa Kỳ, kiểm định kỹ thuật được thực hiện bởi tổ chức ABET. Là một thành viên của ABET, CSAB là một tổ chức tiền hành kiểm định và công nhận các chương trình đào tạo. Tuy quá trình kiểm định có thể khác nhau, nhưng ý nghĩa chung là như nhau ở các nước.

1.1.2. Chứng nhận

Chứng nhận dùng để xác nhận các đặc tính đặc biệt của một người. Một loại chứng nhận phổ biến là chứng nhận chuyên môn, chứng nhận xác nhận một người có khả năng nhất định trong việc hoàn thành một hoạt động ở một lĩnh vực cụ thể. Chứng nhận chuyên môn cũng được dùng để xác minh khả năng của chủ sở hữu có thể đáp ứng các tiêu chuẩn chuyên môn và áp dụng các kỹ năng chuyên môn trong việc giải quyết vấn đề. Ngoài ra, các chứng chỉ chuyên môn còn được dùng để xác minh các kiến thức được quy định, kinh nghiệm và mức độ thành thạo các kỹ năng trong nghề nghiệp. Một kỹ sư thường có được chứng nhận chuyên môn bằng cách vượt qua một kỳ thi kết hợp với các tiêu chí dựa trên các kinh nghiệm khác. Những kỳ thi thường được quản lý bởi các tổ chức phi chính phủ, chẳng hạn như các tổ chức chuyên nghiệp. Trong kỹ nghệ phần mềm, các chứng nhận dùng để kiểm tra trình độ chuyên môn của một kỹ sư phần mềm. Ví dụ, IEEE đã ban hành 2 chương trình chứng nhận (CSDA và CSDP) được thiết kế để xác nhận kiến thức của một kỹ sư phần mềm thông qua các chuẩn kỹ năng trong kỹ nghệ phần mềm. Từ tháng 10 năm 2000, chính phủ Nhật Bản đề xuất sáng kiến chuẩn hoá kỹ năng CNTT châu Á. Theo đó, có hai loại hình chứng chỉ là kỹ sư công nghệ thông tin cơ bản (FE) và kỹ sư thiết kế và phát triển phần mềm (SW). Để có được các chứng chỉ này các kỹ sư phải vượt qua một kỳ thi đánh giá chuẩn kỹ năng. Cho đến nay đã có 11 nước công nhận hai chuẩn trên bao gồm Nhật Bản, Ấn Độ, Trung Quốc, Việt Nam, Malaysia, Hàn Quốc, Singapore, Philipin, Thái Lan, Đài Loan và Myanmar.

1.1.3. Giấy phép

Cấp giấy phép là hành động cấp phép cho một người được thực hiện một số loại hoạt động và chịu trách nhiệm về kết quả thực hiện. Giấy phép thể hiện thông qua việc cấp phép và tài liệu ghi nhận sự cấp phép. Để có được giấy phép sử dụng đòi hỏi không chỉ đáp ứng được một số tiêu chuẩn nhất định mà còn đáp ứng các tiêu chuẩn khi thực hiện. Nói chung việc cấp giấy phép nhằm đảm bảo quyền lợi cho các kỹ sư (so với các cá nhân không đủ tiêu chuẩn). Ở một số nước, các kỹ sư không thể thực hiện công việc như một chuyên gia nếu không được cấp giấy phép. Khắt khe hơn, với các doanh nghiệp, các công ty không được phép cung cấp “dịch vụ kỹ thuật” nếu không có ít nhất một kỹ sư được cấp phép.

1.2. Quy tắc đạo đức và ứng xử nghề nghiệp

Quy tắc đạo đức và ứng xử nghề nghiệp bao gồm các giá trị và hành vi thực hiện chuyên môn và các quyết định của một kỹ sư chuyên nghiệp phải thể hiện. Các quy tắc ứng xử nghề nghiệp được các tổ chức, hội nghề nghiệp. Các quy tắc này tồn tại trong một bối cảnh nhất định và được điều chỉnh để được số đông cộng đồng chấp nhận, phù hợp với với chuẩn mực đạo đức xã hội và luật pháp của chính quyền sở tại. Khi được thiết lập, quy tắc đạo đức và ứng xử nghề nghiệp được thực thi bởi các tổ chức chuyên môn hoặc các tổ chức theo luật định. Các hành vi vi phạm đạo đức và quy tắc ứng xử nghề nghiệp có thể kể đến như tiết lộ thông tin bí mật, làm sai lệch thông tin, hoặc xuyên tạc khả năng của một

người. Một số hành vi khác như sự thiếu sót trong việc thông báo những rủi ro hoặc không cung cấp thông tin quan trọng, không miêu tả đầy đủ quyền lợi của khách hàng. Vi phạm các quy tắc đạo đức và ứng xử nghề nghiệp có thể dẫn đến hình phạt (theo quy định) và có thể bị tước bỏ việc công nhận sự chuyên nghiệp. Một quy tắc đạo đức và ứng xử nghề nghiệp cho phần mềm kỹ thuật đã được phê duyệt bởi Hội đồng ACM và IEEE CS vào năm 1999. Theo đó: "Kỹ sư phần mềm sẽ cam kết thực hiện các quá trình trong phát triển phần mềm gồm: phân tích, đặc tả, thiết kế, phát triển, thử nghiệm và bảo trì phần mềm là một nghề mang lại lợi ích và và được tôn trọng. Phù hợp với các cam kết của họ đối với sức khỏe, an toàn và phúc lợi của công chúng, kỹ sư phần mềm phải tuân thủ các nguyên tắc liên quan đến tám nguyên tắc bao gồm sự công khai, khách hàng và sử dụng lao động, sản phẩm, phán quyết, quản lý, nghề nghiệp, đồng nghiệp và bản thân." Sau khi các quy tắc, tiêu chuẩn được đề xuất có thể được giới thiệu, sửa đổi hoặc thay thế, các kỹ sư có trách nhiệm cập nhật và tiếp tục học tập để phù hợp với các quy tắc nghề nghiệp mới.

1.3. Tính chất và vai trò của các hội nghề nghiệp

Các tổ chức nghề nghiệp bao gồm sự kết hợp của người trong nghề và các chuyên gia, các nhà nghiên cứu. Các tổ chức này thực hiện việc xác định, đề xuất và điều tiết các ngành nghề tương ứng của tổ chức, hội. Các hội, tổ chức nghề nghiệp giúp cộng đồng trong nghề thiết lập các tiêu chuẩn chuyên môn cũng như các quy tắc ứng xử và đạo đức nghề nghiệp. Vì lý do này, họ cũng tham gia vào các hoạt động có liên quan, bao gồm:

- Xây dựng và ban hành nội dung kiến thức chung về chuyên môn
- Kiểm định, chứng nhận và cấp giấy phép;
- Ban hành các quy tắc chung cho cộng đồng nghề
- Thúc đẩy mở rộng cộng đồng thông qua qua các hội nghị, đào tạo, và các ấn phẩm.

Tham gia vào các hội nghề nghiệp, hỗ trợ các kỹ sư trong việc duy trì và phát triển kiến thức phù hợp với chuyên môn, đồng thời mở rộng và duy trì mạng lưới nghề nghiệp của mình.

1.4. Tính chất và vai trò của các tiêu chuẩn kỹ nghệ phần mềm

Các tiêu chuẩn trong kỹ nghệ phần mềm bao gồm nhiều chủ đề nổi bật. Các tiêu chuẩn cung cấp các hướng dẫn cho việc thực hiện các pha trong quy trình phát triển phần mềm. Bằng việc thông qua các nội dung thông qua sự đồng thuận của cộng đồng về các kiến thức và kinh nghiệm trong kỹ nghệ phần mềm, các tiêu chuẩn thiết lập các hướng dẫn cơ bản cho kỹ nghệ phần mềm. Các tiêu chuẩn này có thể được tiếp tục phát triển và hoàn thiện phù hợp với các điều kiện cụ thể. Những lợi ích của việc áp dụng các tiêu chuẩn mang lại

rất nhiều cải thiện chất lượng phần mềm, giúp tránh sai sót, bảo vệ cả phía nhà phát triển và người sử dụng phần mềm, gia tăng tính chuyên nghiệp và thuận lợi trong việc chuyển đổi công nghệ.

1.5. Sự tác động của phần mềm đến nền kinh tế

Phần mềm ít nhiều có các tác động đến kinh tế của từng cá nhân, doanh nghiệp và xã hội. Một phần mềm "thành công" có thể được xác định bởi sự phù hợp của một phần mềm và việc được công nhận cũng như hiệu quả của nó khi áp dụng giải quyết một vấn đề nhất định. Ở cấp độ cá nhân, việc làm của một kỹ sư có thể tuỳ thuộc vào khả năng và sự sẵn sàng của họ trong việc hiểu và thực hiện nhiệm vụ trong đáp ứng nhu cầu và mong đợi của khách hàng hoặc người sử dụng lao động. Khách hàng hoặc tình hình tài chính của chủ sử dụng lao động có thể ảnh hưởng tích cực hay tiêu cực tới việc tiêu thụ phần mềm. Ở mức độ doanh nghiệp, phần mềm được áp dụng một cách hợp lý có thể giảm thiểu thời gian làm việc và gia tăng lợi nhuận khi làm việc. Hơn nữa, với các tổ chức sử dụng hoặc cung cấp phần mềm một cách hiệu quả cũng mang lại lợi ích cho xã hội bằng việc tạo công ăn việc làm và góp phần cải thiện chất lượng dịch vụ. Tuy nhiên, các chi phí phát triển và mua phần mềm cũng có thể tương đương với lợi nhuận kinh doanh của doanh nghiệp. Ở cấp độ xã hội, phần mềm sử dụng việc dự đoán tai nạn, cung cấp các loại dịch vụ xã hội. Sự tác động gián tiếp của phần mềm thông qua việc hiệu quả hoặc không hiệu quả của các đơn vị cung cấp hoặc mua phần mềm dẫn đến tăng hoặc giảm năng xuất lao động xã hội. Đồng thời, phần mềm còn ảnh hưởng đến một số lĩnh vực khác của cuộc sống như trật tự xã hội, thậm chí là cả các vấn đề tài chính...

1.6. Hợp đồng lao động

Dịch vụ kỹ thuật phần mềm có thể được cung cấp theo một loạt các mối quan hệ khách hàng-kỹ sư. Các công việc trong kỹ thuật phần mềm có thể được thực hiện thông qua các nhà cung cấp dịch vụ, tư vấn kỹ thuật, thuê hoặc thậm chí thông qua các tình nguyện viên. Trong tất cả các tình huống, khách hàng và nhà cung cấp đạt được sự đồng thuận về một sản phẩm phần mềm hoặc dịch vụ được cung cấp phải thỏa mãn một số tiêu chí nhất định. Trong kỹ nghệ phần mềm, mối lo ngại chung trong hợp đồng lao động là tính bảo mật. Người sử dụng lao thu được lợi nhuận từ lợi thế thương mại của sở hữu trí tuệ, vì vậy họ buộc phải bảo vệ quyền lợi của mình bằng việc đưa ra những quy định về tính bảo mật trong hợp đồng. Vì vậy, các kỹ sư phần mềm thường được yêu cầu ký không tiết lộ hoặc đồng ý với các quy định về sở hữu trí tuệ trong công việc. Một mối quan tâm khác của quyền sở hữu trí tuệ, các tài sản liên quan tới phần mềm bao gồm sản phẩm, sáng chế, phát minh và các ý tưởng có thể được cất giữ bởi người sử dụng lao động, khách hàng hoặc tuân thủ theo các điều khoản đã được quy định trong hợp đồng, nếu những tài sản thu được trong thời hạn của mối quan hệ các kỹ sư phần mềm với người sử dụng lao động

hoặc khách hàng. Cuối cùng, hợp đồng cũng có thể chỉ định trong số các yếu tố khác vị trí mà tại đó công việc sẽ được thực hiện; tiêu chuẩn mà công việc đó sẽ được tổ chức; cấu hình hệ thống được sử dụng để phát triển; hạn chế của các kỹ sư phần mềm và sử dụng lao động trách nhiệm; một ma trận giao tiếp và / hoặc kế hoạch từng bước; và các chi tiết hành chính như lãi suất, bồi thường, giờ làm việc, và điều kiện làm việc.

1.7. Các vấn đề về luật pháp

Các vấn đề pháp lý đáng chú ý liên quan tới kỹ nghệ phần mềm bao gồm các vấn đề liên quan đến tiêu chuẩn, thương hiệu, bằng sáng chế, bản quyền, bí mật kinh doanh, trách nhiệm pháp lý, yêu cầu pháp lý, tuân thủ luật thương mại, và tội phạm công nghệ.

1.7.1. Các chuẩn

Như đã nhắc ở phần 1.4, các tiêu chuẩn trong kỹ nghệ phần mềm cung cấp các hướng dẫn cho việc thực hiện các pha trong quy trình phát triển phần mềm. Tiêu chuẩn có chức năng định ra các yêu cầu và hỗ trợ thực hiện các yêu cầu đó trong việc thực hiện các hoạt động thường xuyên trong kỹ nghệ phần mềm. Các tiêu chuẩn được đưa ra bằng cách liệt kê các đặc tính tối thiểu của các sản phẩm và quá trình thực hiện. Vì vậy, các doanh nghiệp, tổ chức phần mềm thường áp dụng các tiêu chuẩn như là một phần của chính sách tổ chức của họ. Hơn nữa, việc tuân thủ các tiêu chuẩn là một biện pháp để phòng các rủi ro liên quan đến pháp lý và tránh khỏi những cáo buộc phi pháp.

1.7.2. Thương hiệu

Một thương hiệu liên quan đến bất kỳ từ, tên, biểu tượng, hoặc thiết bị được sử dụng trong các giao dịch. Thương hiệu được sử dụng "để chỉ ra nguồn gốc, xuất xứ của hàng hóa". Bảo vệ thương hiệu bảo vệ tên, logo, hình ảnh, và bao bì của doanh nghiệp. Tuy nhiên, nếu tên, hình ảnh, hay tài sản thương hiệu khác sẽ trở thành một thuật ngữ chung, sau đó bảo vệ thương hiệu là vô hiệu. Tổ chức Sở hữu trí tuệ thế giới (WIPO) là cơ quan đặt ra các quy tắc và quy định về thương hiệu, nhãn hiệu hàng hoá. WIPO là cơ quan của Liên Hợp Quốc dành riêng cho việc sử dụng tài sản trí tuệ như một phương pháp thúc đẩy sự đổi mới và sáng tạo.

1.7.3. Bằng sáng chế

Bằng sáng chế bảo vệ các quyền của một nhà nhà sáng chế trong việc tạo ra và bán các ý tưởng của mình. Một bằng sáng chế bao gồm một tập hợp các đặc quyền được cấp bởi một chính phủ có chủ quyền cho một cá nhân, nhóm cá nhân, tổ chức trong một khoảng thời gian nhất định. Tài liệu để được cấp bằng sáng chế đòi hỏi phải được ghi chép một cách cẩn thận trong quá trình dẫn đến các sáng chế. Các nhà sáng chế có thể sử dụng các đại diện sáng chế (luật sư đại diện) để bảo vệ các quyền lợi hợp pháp của mình về các quy

định trong luật bản quyền. Lưu ý rằng, nếu sáng chế được thực hiện trong quá trình của một hợp đồng công nghệ phần mềm, chủ sở hữu có thể thuộc về chủ nhân hay khách hàng hoặc được phối hợp tổ chức, chứ không phải thuộc về các kỹ sư phần mềm.

1.7.4. Bản quyền

Bản quyền là thuật ngữ pháp lý mô tả quyền của người sáng tạo đối với các sản phẩm, tác phẩm văn học và nghệ thuật của họ. Bảo vệ bản quyền là tự động cho dù sản phẩm này có được đăng ký hay không. Ngay khi sản phẩm được viết ra, nó đã được bảo vệ.

1.7.6. Trách nhiệm pháp lý

Các kỹ sư phần mềm phải quan tâm với các vấn đề về trách nhiệm trong chuyên môn. Với một cá nhân cung cấp dịch vụ cho khách hàng hoặc nhà tuyển dụng, điều này có vai trò quan trọng việc tuân thủ các tiêu chuẩn và các quy định chung trong chuyên môn. Do đó, trách nhiệm pháp lý giúp phía người lao động và người sử dụng lao động tránh khỏi những cáo buộc hoặc thủ tục tố tụng hoặc liên quan đến sờ suất, bất cẩn, hoặc thiếu năng lực. Đối với kỹ sư, bao gồm kỹ sư phần mềm, trách nhiệm pháp lý có liên quan đến trách nhiệm sản phẩm.

1.7.7. Yêu cầu pháp lý

Kỹ sư phần mềm phải hoạt động trong phạm vi của các khuôn khổ pháp lý địa phương, quốc gia và quốc tế. Vì vậy, các kỹ sư phần mềm phải lưu ý các yêu cầu pháp lý:

- Đăng ký và cấp giấy phép bao gồm cả kiểm tra, học vấn, kinh nghiệm, và các yêu cầu đào tạo;
- Thoả thuận trong hợp đồng;
- Các vấn đề pháp lý ngoài hợp đồng;
- Các thông tin cơ bản về khuôn khổ luật pháp quốc tế được quy định bởi tổ chức Thương mại quốc tế (WTO)

1.7.8. Tuân thủ luật thương mại

Tất cả kỹ sư phải được nhận thức hạn chế của pháp luật về nhập, xuất, tái xuất sản phẩm dịch vụ và công nghệ trong các phán quyết của mình. Các cân nhắc bao gồm kiểm soát xuất và phân loại, thu hồi giấy phép cần thiết dành cho việc sử dụng phần cứng và phần mềm, dịch vụ và công nghệ. Các kỹ sư cần được tư vấn để được hướng dẫn chi tiết các việc cần tuân thủ.

1.7.9. Tội phạm công nghệ

Một số loại tội phạm công nghệ có thể kể đến như gian lận, truy cập trái phép, spam, nội dung khiêu dâm, xúc phạm, đe dọa, quấy rối, trộm cắp dữ liệu cá nhân nhạy cảm hoặc bí mật thương mại, và sử dụng một máy tính để gây thiệt hại hay xâm nhập vào các máy tính khác trong mạng và hệ thống điều khiển tự động. Theo luật Việt Nam có một số quy định theo các điều 224, 225, 226 Bộ luật Hình sự. Phạm tội thuộc một trong các trường hợp sau đây, thì bị phạt tù từ ba năm đến bảy năm:

- * Có tổ chức;
- * Lợi dụng chức vụ, quyền hạn;
- * Thu lợi bất chính lớn;
- * Gây hậu quả nghiêm trọng;
- * Tái phạm nguy hiểm.

Phạm tội thuộc một trong các trường hợp sau đây, thì bị phạt tù từ năm năm đến mười hai năm:

- * Đối với hệ thống dữ liệu thuộc bí mật nhà nước; hệ thống thông tin phục vụ an ninh, quốc phòng;
- * Đồi với cơ sở hạ tầng thông tin quốc gia; hệ thống thông tin điều hành lưới điện quốc gia;
- * Thu lợi bất chính rất lớn hoặc đặc biệt lớn;
- * Gây hậu quả rất nghiêm trọng hoặc đặc biệt nghiêm trọng.

1.8. Tài liệu

Cung cấp tài liệu hướng dẫn rõ ràng, kỹ lưỡng, và chính xác là trách nhiệm của mỗi kỹ sư phần mềm. Sự đầy đủ của các tài liệu được đánh giá theo các tiêu chí khác nhau dựa trên nhu cầu của các bên yêu cầu liên quan khác nhau. Tài liệu tốt phải đạt được các tiêu chuẩn và hướng dẫn. Đặc biệt, các kỹ sư phần mềm nên làm tài liệu tài liệu:

- Có sự liên kết
- Rủi ro trọng yếu và sự cân bằng, và
- Cảnh báo về những hậu quả không mong muốn hoặc nguy hiểm từ việc sử dụng hoặc sử dụng sai của phần mềm.

2. Tương tác nhóm

2.1. Lợi thế làm việc theo nhóm

Kỹ sư phần mềm sẽ làm việc với đội phát triển dự án, cũng như với các bên liên quan khác để đảm bảo dự án đi đúng hướng và đúng tiến độ. Các bên liên quan gồm:

- Người dùng: Gồm những người sử dụng cuối của hệ thống

- Khách hàng: Gồm những người đặt hàng sản phẩm phần mềm. Họ có quyền đưa sản phẩm của họ lên các cửa hàng phần mềm
- Nhà phân tích thị trường: Gồm những người phân tích nhu cầu thị trường.
- Bên pháp lý: Tùy theo tính chất của sản phẩm phần mềm, sản phẩm cần thỏa mãn các yêu cầu pháp lý cụ thể.
- Nhóm xây dựng phần mềm: Những cá nhân xây dựng sản phẩm phần mềm dựa theo yêu cầu phía khách hàng.

Các thành viên trong nhóm sẽ chia sẻ gánh nặng công việc với nhau, cũng như chia sẻ kiến thức dựa trên phẩm chất tôn trọng và trung thực. Quá trình giao tiếp giữa các thành viên có thể tiến hành qua giao tiếp trực tiếp (họp nhóm, trao đổi cá nhân), qua tài liệu (Ví dụ: mã nguồn). Khi các thành viên trong nhóm phối hợp chặt chẽ với nhau dưới sự điều phối của người quản lý, sản phẩm phần mềm sẽ giảm đáng kể các lỗi phát sinh. Đồng thời, thời gian xây dựng sản phẩm sẽ được đẩy nhanh lên.

2.2. Nhận thức cá nhân

Một điều hiển nhiên đối với kĩ sư phần mềm là họ luôn phải đối mặt với các vấn đề phát sinh bất ngờ và liên tục trong quá trình xây dựng sản phẩm. Đối với những vấn đề đơn giản hoặc đã có hướng giải quyết từ trước, kĩ sư phần mềm dễ dàng vượt qua thử thách này. Tuy nhiên trong thực tế, các vấn đề nảy sinh thường không biết trước cách xử lí hiệu quả ngay lập tức. Bài toán làm sao giải quyết vấn đề một cách hiệu quả là một bài toán khó. Để giải quyết bài toán này, chúng ta sẽ tìm hiểu các yếu tố ảnh hưởng đến nhận thức cá nhân và các phương pháp để cải thiện nhận thức cá nhân.

Các yếu tố rào cản ảnh hưởng đến nhận thức cá nhân gồm:

- Thiếu kiến thức liên quan đến vấn đề phát sinh
- Dữ liệu phân tích từ vấn đề lớn
- Tâm lý lo sợ thất bại
- Thiếu khả năng biểu thị vấn đề
- Ảnh hưởng bởi môi trường làm việc
- Ảnh hưởng bởi trạng thái cảm xúc của cá nhân

Kĩ sư phần mềm cần rèn luyện cho mình khả năng tập trung để giải quyết vấn đề, hoặc nhờ các thành viên trong nhóm xem xét vấn đề cho mình. Bằng cách này, gánh nặng công việc đã vô tình chia sẻ cho các thành viên khác trong nhóm. Bằng các cách nhìn khác nhau với cùng một vấn đề, vấn đề được phân tích tinh tế và thấu đáo.

Ngoài ra, kĩ sư phần mềm có thể giải quyết vấn đề sử dụng nỗ lực cá nhân bằng kĩ thuật giảm tải nhận thức. Một phương pháp phổ biến hay được sử dụng là kĩ thuật chia nhỏ vấn đề. Sau khi vấn đề được chia nhỏ, tại mỗi một thời điểm kĩ sư phần mềm chỉ giải quyết một

bài toán con đó. Tổng hợp các bài toán con lại, kĩ sư phần mềm dễ dàng tìm lời giải cho bài toán lớn.

2.3. Giải quyết vấn đề phức tạp

Đa phần các vấn đề mà kĩ sư phần mềm vấp phải khá phức tạp và khó khăn để tìm hướng giải quyết toàn bộ hoặc giải quyết bởi một nhóm các kĩ sư phần mềm riêng rẽ. Khi những trường hợp tương tự như vậy xảy ra, hướng giải quyết thường dùng được thực hiện là làm việc nhóm và phân rã vấn đề.

Làm việc nhóm cùng nhau giúp giải quyết vấn đề phức tạp và vấn đề lớn bằng cách chia sẻ gánh nặng và nỗ lực dựa trên tri thức và óc sáng tạo của các cá nhân. Khi các kĩ sư phần mềm làm việc theo nhóm, các góc nhìn và khả năng khác nhau của từng cá nhân sẽ bồi sung lẫn nhau và giúp xây dựng phương án khả thi. Ví dụ một vài nhóm điển hình là lập trình theo cặp trong mô hình Agile và đánh giá mã nguồn trong SQA.

2.4. Tương tác với các bên liên quan

Sự thành công của một nỗ lực ngành kĩ thuật phần mềm phụ thuộc vào các tương tác tích cực với các bên liên quan. Họ cung cấp sự hỗ trợ, thông tin và phản hồi ở mọi giai đoạn trong vòng đời phát triển phần mềm. Ví dụ, trong thời kì giai đoạn đầu, khá quan trọng để nhận diện các bên liên quan và xác định sản phẩm ảnh hưởng đến họ như thế nào. Sau đó, sự xác định đầy đủ về yêu cầu của các bên liên quan sẽ bắt được đúng và hoàn hảo.

Trong suốt sự phát triển, các bên liên quan có thể phản hồi về đặc tả và/hoặc các phiên bản sớm của phần mềm, sự thay đổi độ ưu tiên, cũng như làm rõ các yêu cầu phần mềm mới hoặc chi tiết các yêu cầu phần mềm. Cuối cùng, trong pha bảo trì phần mềm cho tới khi kết thúc hẵn dự án, các bên liên quan cung cấp phản hồi về sự tiến hóa hoặc về các yêu cầu mới cũng như các thông báo để phần mềm có thể kế thừa và cải tiến.

2.5. Giải quyết các vấn đề không chắc chắn và mơ hồ

Cũng như các kĩ sư ở lĩnh vực khác, các kĩ sư phần mềm thường phải đối phó và giải quyết sự không chắc chắn và mơ hồ trong khi hỗ trợ dịch vụ và phát triển sản phẩm. Các kĩ sư phần mềm này phải tấn công và giảm thiểu hoặc loại trừ bất kì sự thiếu rõ ràng vốn là một vật cản để thực hiện công việc.

Thông thường, sự không chắc chắn đơn giản là sự phản ánh của thiếu kiến thức. Trong trường hợp này, sự nghiên cứu các nguồn tài nguyên tới các nguồn chính thức như sách, tạp chí chuyên ngành, phỏng vấn với các bên liên quan, hoặc tham khảo các thành viên trong nhóm có thể giúp vượt qua vấn đề.

Khi sự không chắc chắn và sự mơ hồ không thể vượt qua dễ dàng, các kỹ sư phần mềm hoặc tổ chức có thể coi vấn đề này là một hiểm họa dự án. Trong trường hợp này, các ước lượng công việc hoặc chi phí được điều chỉnh để giảm bớt chi phí đoán trước được sao cho giải quyết vấn đề đó.

2.6. Đối phó với môi trường đa văn hóa

Môi trường đa văn hóa có thể có những tác động đến hành vi một nhóm. Điều này đặc biệt đúng khi nhóm bị phân chia bởi môi trường địa lý hoặc giao tiếp không thường xuyên. Quá trình giao tiếp thậm chí khá khó khăn nếu các thành viên có múi giờ khác nhau khiến cho tần suất giao tiếp càng ít hơn.

Môi trường đa văn hóa thấy khá nhiều trong kỹ nghệ phần mềm, có thể hơn cả các ngành nghề khác liên quan đến kỹ thuật. Nguyên nhân bởi vì xu hướng mạnh mẽ về outsourcing xuyên quốc gia và sự dễ gửi các thành phần phần mềm ngay lập tức trên khắp toàn cầu. Ví dụ, một cách thường thấy trong dự án phần mềm là chia nhỏ thành các vấn đề dựa trên biên giới văn hóa và dân tộc. Cũng khá thường thấy các thành viên trong một nhóm gồm các thành viên đến từ những nơi có nền tảng văn hóa khác nhau.

Để một dự án phần mềm thành công, các thành viên trong nhóm cần học sự khoan dung, công nhận những quy tắc dựa trên tiêu chuẩn xã hội. Hầu hết các giao tiếp bao gồm gặp mặt trực tiếp có thể giảm thiểu sự phân chia về mặt địa lý và văn hóa, đồng thời thúc đẩy sự kết nối và gia tăng năng suất. Hơn nữa, cuộc trò chuyện có thể sử dụng ngôn ngữ mẹ đẻ của đối phương sẽ mang lại lợi ích rất nhiều để kết nối các cá nhân với nhau.

3. Kỹ năng giao tiếp

Đối với kỹ sư phần mềm, kỹ năng giao tiếp vô cùng quan trọng gồm kỹ năng nói, kỹ năng đọc và kỹ năng viết. Sự thành công của dự án bị ảnh hưởng trực tiếp bởi thông tin trao đổi giữa kỹ sư phần mềm, khách hàng, người giám sát, đồng nghiệp và nhà hỗ trợ. Giải quyết vấn đề một cách tối ưu có thể đạt được qua bước nghiên cứu, suy ngẫm và tóm tắt thông tin. Sự chấp nhận sản phẩm khách hàng và sử dụng sản phẩm an toàn phụ thuộc vào sự chuẩn bị của các khóa đào tạo và tài liệu liên quan. Sự thành công trong sự nghiệp của một kỹ sư phần mềm bị ảnh hưởng bởi khả năng giao tiếp bằng lời nói và khả năng viết sao cho hiệu quả.

3.1. Đọc, hiểu và tóm tắt

Các kỹ sư phần mềm có thể đọc và hiểu các tài liệu kỹ thuật. Các tài liệu kỹ thuật gồm sách tham khảo, hướng dẫn sử dụng, bài báo và mã nguồn. Kỹ năng đọc không phải là phương pháp chính để cải thiện kỹ năng, nhưng đó là phương pháp để thu thập các thông tin cần

thiết để đạt được mục tiêu dự án. Một kĩ sư phần mềm chọn lọc thông tin qua lượng thông tin rất lớn, lọc lấy các mảnh thông tin mà họ cảm thấy hữu ích. Khách hàng có thể yêu cầu kĩ sư phần mềm tóm tắt kết quả của những thông tin đã được tập hợp lại cho họ, đơn giản hóa thông tin hoặc giải thích các thông tin. Mục đích của hành động này nhằm giúp phía khách hàng có thể đưa ra quyết định cuối cùng.

3.2. Kĩ năng viết

Kĩ sư phần mềm có thể xây dựng sản phẩm tài liệu được yêu cầu từ phía khách hàng. Những sản phẩm tài liệu này gồm mã nguồn, kế hoạch dự án, tài liệu yêu cầu dự án, phân tích hiểm họa, tài liệu thiết kế phần mềm, báo cáo kĩ thuật, biểu đồ, v.v. Viết sao cho rõ ràng và súc tích là một điều quan trọng vì đó thường là phương pháp chính để giao tiếp giữa các bên liên quan. Trong mọi trường hợp, sản phẩm được viết bởi kĩ sư phần mềm cần được xây dựng sao cho dễ hiểu, rõ ràng đối với người đọc.

3.3. Giao tiếp nhóm

Kĩ năng giao tiếp hiệu quả giữa các thành viên trong nhóm là cần thiết để đạt được mục tiêu dự án. Các bên liên quan sẽ được tham khảo để lấy ý kiến, các quyết định sẽ được tạo ra, và kế hoạch phải được xây dựng. Càng đông thành viên trong một nhóm, yêu cầu giao tiếp càng trở nên quan trọng. Một vài giao tiếp có thể thực hiện được qua kĩ thuật viết. Giao tiếp qua tài liệu phần mềm là một cách thay thế phổ biến cho tương tác trực tiếp. Giao tiếp qua email là cách khác rất hữu ích nhưng không đủ. Nguyên nhân bởi vì, khi một bên gửi quá nhiều tin nhắn, thì bên nhận rất khó để xác định đâu là thông tin quan trọng. Các tổ chức thường dùng các phần mềm cộng tác để chia sẻ thông tin.Thêm vào đó, việc sử dụng cửa hàng thông tin điện tử là khả thi đối với các thành viên trong nhóm. Thông tin được lưu trên cửa hàng gồm các chính sách của tổ chức, các chuẩn, các thủ tục kĩ nghệ phổ biến, thông tin về các dự án riêng biệt.

3.4. Kĩ năng trình bày

Các kĩ sư phần mềm dựa vào kĩ năng trình bày của họ trong suốt quá trình phát triển dự án. Ví dụ, trong quá trình xây dựng tài liệu đặc tả, các kĩ sư phần mềm sẽ cùng làm việc với bên khách hàng và đồng đội qua tài liệu đặc tả, qua xem xét yêu cầu. Khả năng truyền đạt của kĩ sư phần mềm trong một bài thuyết trình bị ảnh hưởng bởi sự hỗ trợ từ phía khách hàng, sự quản lý và sự chấp nhận sản phẩm. Ngoài ra, yêu tố này còn bị ảnh hưởng bởi khả năng phân tích của các bên liên quan và sự hỗ trợ của họ trong nỗ lực tạo ra sản phẩm. Những tri thức này cần được tài liệu hóa trong các mẫu slide, bài báo hoặc bất kì tài liệu nào khác tiên lợi cho việc tạo tri thức.

Chapter 10: Software Quality

Group 10:

- Nguyễn Gia Dũng
- Nguyễn Ngọc Khánh

1. Software Quality Fundamentals

Software Engineering Culture and Ethics

Value and Costs of Quality

Models and Quality Characteristics

Software Quality Improvement

Software Safety

1.1 Software Engineering Culture and Ethics

Một nền văn hóa phần mềm lành mạnh bao gồm nhiều đặc điểm, bao gồm:

- Có sự cam kết về chất lượng của người lập trình, đảm bảo cân bằng giữa chi phí, tiến độ, chất lượng.
- Một sản phẩm phần mềm tốt là sản phẩm được báo cáo chính xác thông tin và các kết quả liên quan đến chất lượng.
- Do yêu cầu đảm bảo chất lượng phần mềm nên 1 số tổ chức đã tạo ra các bộ chuẩn về các nguyên tắc trong chất lượng phần mềm

Ví dụ: IEEE Computer Society and the ACM.

1.2. Value and Costs of Quality

CoSQ: Cost of software Quality (Chi phí chất lượng phần mềm) • Chuẩn tập hợp 1 bộ các phép đo đánh giá về giá trị kinh tế trong quá trình phát triển phần mềm và bảo trì. • Four cost of quality categories: (4 loại chi phí)

- prevention
- appraisal
- internal failure
- external failure

1.2.1. Prevention: Chi phí phòng ngừa

- Bao gồm các khoản đầu tư vào các nỗ lực cải tiến quy trình phần mềm, chất lượng cơ sở hạ tầng, các công cụ chất lượng, đào tạo, kiểm tra, đánh giá và quản lý.
- Các chi phí này thường không cụ thể cho một dự án

1.2.2. Appraisal: Chi phí thẩm định

- Phát sinh từ các hoạt động dự án mà tìm thấy lỗi.
- Phân loại: chi phí đánh giá thiết kế và chi phí kiểm thử (phần mềm kiểm tra đơn vị, tích hợp phần mềm, kiểm tra hệ thống cấp, kiểm tra chấp nhận)
- chi phí thẩm định sẽ được mở rộng cho các nhà cung cấp phần mềm hợp đồng phụ

1.2.3. Internal failure: Chi phí thất bại nội bộ

Là chi phí được phát sinh để sửa chữa những lỗi tìm thấy trong quá trình hoạt động thẩm định và phát hiện ra trước khi phân phối sản phẩm cho khách hàng

1.2.4. External failure: Chi phí thất bại bên ngoài

Bao gồm các hoạt động ứng phó với các vấn đề phần mềm phát hiện sau khi giao hàng cho khách hàng.

Summary: Kỹ sư phần mềm có thể sử dụng phương pháp CoSQ để xác định cấp độ chất lượng phần mềm và cũng có thể trình bày lựa chọn sao cho cân bằng giữa chất lượng và chi phí của sản phẩm

1.3 Models and Quality Characteristics:

Mô hình và đặc điểm của chất lượng

- Thuật ngữ mô tả đặc tính chất lượng phần mềm (hoặc mô hình của chất lượng phần mềm)
- Mỗi mô hình có các mức phân cấp và đặc tính khác nhau.

Ví dụ: ISO/IEC 25.010: 2011 bao gồm

- identifying software and system requirements (Yêu cầu của phần mềm và hệ thống)
- validating the comprehensiveness of a requirements definition (Tính toàn diện của hệ thống)
- identifying software and system design objectives (Yêu cầu thiết kế theo mục tiêu của phần mềm và hệ thống)
- identifying software and system testing objectives (Yêu cầu kiểm thử theo mục tiêu)
- identifying quality control criteria as part of quality assurance (Yêu cầu về tiêu chí quản lý chất lượng – là 1 phần của việc đảm bảo chất lượng)

- identifying acceptance criteria for a software product and/or software-intensive computer system (Các tiêu chí ở mức chấp nhận được cho 1 sản phẩm phần mềm)
- establishing measures of quality characteristics in support of these activities (Các biện pháp hỗ trợ cho đặc tính chất lượng)

1.4 Software Quality Improvement

Cải thiện chất lượng phần mềm

- Thông qua các quy trình phòng ngừa hoặc một quá trình lặp đi lặp lại các cải tiến liên tục, đòi hỏi có sự kiểm soát quản lý, điều phối, và thông tin phản hồi từ nhiều tiến trình đồng thời:
 - (1) các quá trình trong vòng đời phần mềm.
 - (2) quá trình lỗi / khiếm khuyết phát hiện, loại bỏ, và phòng ngừa.
 - (3) các quá trình cải tiến chất lượng.

Ví dụ:

- Xây dựng chất lượng sản phẩm thông qua việc phát hiện phòng ngừa sớm các lỗi, cải tiến liên tục giữa các bên liên quan trong quá trình phát triển sản phẩm.
=> Các chuyên gia đã khẳng định: Chất lượng của một sản phẩm được liên kết trực tiếp đến chất lượng của quá trình phát triển và sử dụng.
- PDCA (Plan- Do- Check- Act): Lên kế hoạch – thực hiện – kiểm tra – chỉnh sửa •
- Evolutionary delivery: Khả năng tiến hóa
-
- QFD (quality function deployment): Cải tiến chất lượng triển khai •
=> Cung cấp các kỹ thuật để xác định mục tiêu chất lượng sản phẩm và xác định xem họ được đáp ứng như thế nào.

1.5 Software Safety

Đảm bảo An toàn phần mềm

- Thực tế: Một lỗi hệ thống trong 1 sản phẩm có thể gây tổn hại cho đời sống con người - các loài sinh vật khác, cấu trúc vật lý, hoặc môi trường.
•
=> Các phần mềm trong các hệ thống này coi trọng an toàn

Ví dụ:

Hệ thống quản lý bay, nhà máy sản xuất hóa chất, các thiết bị y tế. Một lỗi trong hoạt động của hệ thống phần mềm trong các ngành này có thể có tác động rất lớn, thậm chí gây nên thảm họa.

Phân loại

- Direct (Trực tiếp) có thể là phần mềm nhúng trong một hệ thống, chẳng hạn như máy tính điều khiển chuyến bay của công ty quản lý bay •
- Indirect (Gián tiếp) bao gồm các ứng dụng phần mềm được sử dụng trong các môi trường phát triển công nghệ phần mềm và môi trường kiểm tra phần mềm. •

Ví dụ: chuẩn DO - 178C của tổ chức RTCA

- Catastrophic (mức thảm họa) Thất bại có thể gây nguy hiểm tính mạng cho rất nhiều người (thường là trong lĩnh vực hàng không) •
- Hazardous (nguy hiểm) Tác động lớn đến sự an toàn, có thể gây nguy hiểm đến tính mạng của 1 số ít người •
- Major: không làm giảm đáng kể sự an toàn, tuy nhiên có thể tăng lượng công việc phải làm, nhiều khi dẫn tới sự khó chịu. •

Ví dụ: một sự thay đổi kế hoạch bay thường lệ .

- No Effect: Nếu xảy ra lỗi thì cũng không ảnh hưởng hoặc tác động đến sự an toàn và vận hành của hệ thống cũng như khối lượng công việc. •

2. Software Quality Management Processes (SQM)

- Quản lý chất lượng phần mềm là tập hợp các quy trình để đảm bảo rằng các sản phẩm phần mềm, dịch vụ, và vòng đời của quá trình triển khai sản phẩm đạt được sự hài lòng của các bên liên quan.
-
- SQM định nghĩa các quy trình, chủ sở hữu quá trình, các yêu cầu đối với các quá trình, các phép đo của các quá trình và kết quả đầu ra của họ, và các kênh thông tin phản hồi trong suốt toàn bộ chu kỳ đời phần mềm . •

SQM comprises four subcategories (4 loại)

- Software quality planning (SQP) Lập kế hoạch (xác định các tiêu chuẩn chất lượng sẽ được sử dụng, xác định mục tiêu chất lượng cụ thể •
- Software quality assurance (SQA): Đảm bảo chất lượng phần mềm •
- Software quality control (SQC): Kiểm soát chất lượng phần mềm •
- Soft- ware process improvement (SPI): Cải tiến quy trình •

2.1. Software quality planning (SQP) Lập kế hoạch

- Bao gồm việc xác định chất lượng tiêu chuẩn sẽ được sử dụng, xác định mục tiêu chất lượng cụ thể, và ước lượng tiến độ của các hoạt động phát triển phần mềm.
- Trong một số trường hợp, lập kế hoạch chất lượng phần mềm cũng bao gồm việc xác định

các quy trình chất lượng phần mềm được sử dụng. •

2.2. Software quality assurance (SQA): Đảm bảo chất lượng phần mềm

- Xác định và đánh giá sự phù hợp của các quy trình phần mềm.
- Đảm bảo các sản phẩm phần mềm có chất lượng phù hợp cho mục đích, dự định của họ.
-

2.3. Software quality control (SQC): Kiểm soát chất lượng phần mềm

- Xác định chất lượng phần mềm phải tuân thủ các tiêu chuẩn được thiết lập cho các dự án (bao gồm cả các yêu cầu, ràng buộc, thiết kế, hợp đồng, và các kế hoạch).
-
- SQC đánh giá trung gian sản phẩm cũng như các sản phẩm cuối cùng. •

2.4. Software process improvement (SPI): Cải tiến quy trình

- Cải thiện chất lượng phần mềm bằng hoạt động khắc phục - phòng ngừa, tìm cách nâng cao hiệu quả quá trình • Mặc dù SPI có thể được thực thi trong ba loại trước, tuy nhiên do yêu cầu cao của 1 số sản phẩm mà SPI được tổ chức thành một thể loại riêng biệt. •

2.5. Verification & Validation

Verification & Validation (Xác minh và xác thực)

Mục đích:

- Giúp phát triển và xây dựng chất lượng vào hệ thống trong vòng đời của phần mềm.
- Cung cấp một đánh giá khách quan của sản phẩm và các quá trình trong suốt vòng đời.
-
- Các quy trình V&V xác định xem các sản phẩm phát triển có phù hợp với nhu cầu sử dụng.
- Verification: đảm bảo rằng các sản phẩm được xây dựng chính xác, theo nghĩa là những sản phẩm đầu ra của một hoạt động đáp ứng các yêu cầu kỹ thuật đặt ra. •
- Validation: đảm bảo rằng các sản phẩm phải được xây dựng có nghĩa là, các sản phẩm đáp ứng đúng mục đích cụ thể của nó. •

Cả hai quá trình xác minh và xác nhận quá trình bắt đầu từ sự phát triển và duy trì đến hết giai đoạn.

Cung cấp tính năng kiểm tra sản phẩm trong mối quan hệ với giai đoạn trước đó và ngay lập tức cập nhật các thông số kỹ thuật để được giải quyết

2.6. Reviews and Audits

Nhận xét và kiểm toán bao gồm việc

- Người quản lý sẽ đánh giá kết quả dự án thực tế so với kế hoạch

- Đánh giá, kiểm tra về mặt kỹ thuật của sản phẩm.
- Đảm bảo qui trình kiểm toán.
- Đảm bảo sản phẩm sau khi kiểm toán.

3.Practical Considerations

(Xem xét thực tế)

- 3.1. Software Quality Requirements (Yêu cầu chất lượng phần mềm) •
- 3.2. Defect Characterization (Các điểm khiếm khuyết) •
- 3.3. Software Quality Management Techniques (Kỹ thuật quản lý chất lượng phần mềm) •
- 3.4. Software Quality Measurement (Đo lường chất lượng phần mềm) •

3.1. Software Quality Requirements

(Yêu cầu chất lượng phần mềm)

- 3.1.1. Influence Factors (Các yếu tố ảnh hưởng) •
- 3.1.2. Dependabilit (Độ đáng tin cậy) •
- 3.1.3. Integrity Levels of Software (Mức toàn vẹn của phần mềm) •

3.1.1. Influence Factors

Các yếu tố ảnh hưởng đến việc lập kế hoạch, quản lý, và lựa chọn các hoạt động quản lý chất lượng phần mềm, bao gồm:

- Môi trường vật lý của các hệ thống phần mềm •
- Hệ thống và phần mềm chức năng và yêu cầu chất lượng của chúng •
- Các thành phần thương mại hoặc tiêu chuẩn được sử dụng trong hệ thống •
- Các tiêu chuẩn kỹ thuật phần mềm cụ thể được áp dụng •
- Các phương pháp và các công cụ phần mềm được sử dụng để phát triển và bảo trì •
- Ngân sách, nhân viên, tổ chức, kế hoạch, dự án, và lịch trình của tất cả các quá trình •
- Những người sử dụng và có ý định sử dụng hệ thống •
- Mức toàn vẹn của hệ thống •

3.1.2. Dependability

- Độ đáng tin cậy bao gồm các đặc điểm như: •
 - +Tính sẵn sàng •
 - +An toàn •
 - +Bảo mật •
 - +Có lỗi hay không •
 - ... •

3.1.3. Integrity Levels of Software

- Xác định mức toàn vẹn là một phương pháp quản lý rủi ro. •
- Mức toàn vẹn của phần mềm là một loạt các giá trị phức tạp đại diện cho các phần mềm như: •
 - mức độ rủi ro •
 - mức độ an toàn •
 - mức độ bảo mật •
 - hiệu suất mong muốn •
 - độ tin cậy •
 - ... •

(Các mức toàn vẹn phần mềm giao có thể thay đổi khi các phần mềm phát triển)

3.2. Defect Characterization

- Đánh giá chất lượng phần mềm là các kỹ thuật tìm các khiếm khuyết, và các lỗi. •
- Theo phương pháp thiết kế mới và các ngôn ngữ phát triển, cùng với những tiến bộ trong phần mềm các khiếm khuyết mới xuất hiện. •
- Khi theo dõi các khiếm khuyết, các kỹ sư phần mềm quan tâm đến không chỉ số lượng của các khiếm khuyết mà còn các loại. Thông tin một mình, không có một số phân loại, có thể không đủ để xác định nguyên nhân của các khiếm khuyết. •
- Một số loại khiếm khuyết: •
 - Lỗi tính toán
 - Lỗi do người sử dụng
 - Các khiếm khuyết
 - Lỗi trong mã nguồn

3.3. Software Quality Management Techniques

- Kỹ thuật quản lý chất lượng phần mềm có thể được phân loại theo nhiều cách, nhưng cách tiếp cận đơn giản là gồm hai loại: tĩnh và động. •
- Kỹ thuật động liên quan đến việc thực hiện các phần mềm •
- Kỹ thuật tĩnh liên quan đến việc phân tích tài liệu và mã nguồn, nhưng không thực hiện các phần mềm. •

3.4. Software Quality Measurement

- Những kỹ thuật đo lường bao gồm: •
 - Thống kê
 -
 - Thủ nghiệm thống kê •

- Phân tích xu hướng •

- Dự báo •

4. Software Quality Tools

- Công cụ chất lượng phần mềm bao gồm các công cụ phân tích tĩnh và động. •
- Công cụ phân tích tĩnh đầu vào mã nguồn, thực hiện phân tích cú pháp và ngữ nghĩa mà không thực hiện mã này, và kết quả hiện nay cho người dùng. •
- Công cụ phân tích động sẽ thực hiện các chương trình và phân tích quá trình đó.

1. Định nghĩa Quy trình phần mềm

Quản lý quy trình phần mềm

Cơ sở hạ tầng quy trình phần mềm

1. Vòng đời của phần mềm

Các loại quy trình phần mềm

Những mô hình vòng đời phần mềm

Điều chỉnh quy trình phần mềm

Liên hệ trong thực tế

1. Đánh giá và cải thiện quy trình phần mềm

Những mô hình đánh giá quy trình phần mềm

Những phương pháp đánh giá quy trình phần mềm

Những mô hình cải thiện quy trình phần mềm

Xếp hạng nối tiếp và giai đoạn quy trình phần mềm

1. Cách đo lường phần mềm

Cách đo lường quy trình phần mềm và sản phẩm

Chất lượng của kết quả sự đo lường

Những mô hình thông tin phần mềm

Những kỹ thuật đo lường quy trình phần mềm

1. Những công cụ quy trình kỹ thuật phần mềm

1. Định nghĩa Quy trình phần mềm

Quy trình công nghệ phần mềm có liên quan với các hoạt động công việc hoàn thành bởi các kỹ sư phần mềm để phát triển, duy trì và vận hành phần mềm, chẳng hạn như yêu cầu, thiết kế, xây dựng, thử nghiệm, quản lý cấu hình, và các quy trình công nghệ phần mềm khác. Để dễ hiểu, "quy trình công nghệ phần mềm" sẽ được gọi là "quy trình phần mềm". Quy trình phần mềm là một cấu trúc, bao gồm tập hợp các hoạt động cần thiết và các kết quả tương ứng, sử dụng trong việc phát triển, sản xuất ra một sản phẩm phần mềm. Quy trình phần mềm được đặc tả vì một số lý do: để tạo điều kiện cho sự hiểu biết của con

người, thông tin liên lạc và phối hợp; để hỗ trợ quản lý dự án phần mềm; để đo lường và nâng cao chất lượng sản phẩm phần mềm một cách hiệu quả; để hỗ trợ quá trình cải tiến; và để cung cấp một cơ sở cho việc hỗ trợ tự động thực hiện quy trình.

Định nghĩa đầy đủ về quy trình phần mềm bao gồm việc hỗ trợ thực hiện quy trình phần mềm

- Môi trường làm việc hỗ trợ việc thực hiện quy trình phần mềm.
 - Đặc tả vai trò và năng lực mỗi đơn vị trong tổ chức.
 - Có biện pháp đánh giá hiệu quả việc thực hiện quy trình phần mềm. Ví dụ bằng chỉ số KPI
- Quy trình phần mềm có thể mô tả bằng ngôn ngữ tự nhiên, hoặc bằng các công cụ như [UML](#), [BPMN](#); [IDEF0](#) hoặc sơ đồ.

1.1 Quản lý quy trình phần mềm

Quản lý quy trình phần mềm nhằm nhận ra hai mục tiêu: Nhận ra năng suất và hiệu quả của việc áp dụng quy trình phần mềm vào trong dự án hoặc tổ chức. Việc thay đổi hoặc thêm mới một quy trình phần mềm nhằm mục đích cải tiến hiệu quả và năng suất hoạt động của quy trình, từ đó cải thiện kết quả của quá trình làm ra sản phẩm. Thay đổi hoặc cải tiến quy trình có liên quan chặt chẽ tới việc thay đổi mô hình tổ chức hoặc cơ sở hạ tầng của tổ chức. Mục tiêu của sự thay đổi này nhằm mục đích cải thiện giá thành sản phẩm, thời gian xây dựng sản phẩm hoặc chất lượng của sản phẩm phần mềm. Thay đổi quy trình của một tổ chức thường kéo theo thay đổi nhiều thứ, ví dụ như mô hình của tổ chức đó, và ngược lại.

1.2 Cơ sở hạ tầng quy trình phần mềm

Thiết lập, thực hiện và quản lý các quá trình phần mềm và các mô hình vòng đời phần mềm thường xảy ra ở cấp độ các dự án phần mềm. Tuy nhiên, ứng dụng có hệ thống các quy trình phần mềm và các mô hình vòng đời phần mềm qua một tổ chức có thể đem lại lợi ích cho tất cả các phần mềm làm việc trong tổ chức. Một cơ sở hạ tầng quy trình phần mềm có thể cung cấp các định nghĩa quy trình, chính sách để giải thích và áp dụng các quy trình, và mô tả các thủ tục được sử dụng để thực hiện các quy trình. Ngoài ra, một cơ sở hạ tầng quy trình phần mềm có thể cung cấp kinh phí, công cụ, đào tạo, và các nhân viên đã được phân công trách nhiệm cho việc thiết lập và duy trì cơ sở hạ tầng quy trình phần mềm.

Cơ sở hạ tầng quy trình phần mềm tùy thuộc vào kích thước và độ phức tạp của tổ chức và các dự án thực hiện trong tổ chức. Các tổ chức và các dự án nhỏ, đơn giản có nhu cầu cơ sở hạ tầng đơn giản. Các tổ chức và các dự án phức tạp có cơ sở hạ tầng quá trình phần mềm lớn hơn và phức tạp hơn. Trong trường hợp đó, đơn vị tổ chức khác nhau có thể được thiết lập (chẳng hạn như một quá trình nhóm kỹ sư phần mềm hoặc một ban chỉ đạo) để giám sát việc thực hiện và cải tiến các quy trình phần mềm.

Một nhận thức sai lầm phổ biến là việc thiết lập một cơ sở hạ tầng quy trình phần mềm và thực hiện các quy trình phần mềm lặp lại sẽ thêm thời gian và chi phí để phát triển phần mềm và bảo trì. Có chi phí cần thiết liên quan đến việc giới thiệu hoặc cải thiện một quá trình phần mềm; Tuy nhiên, kinh nghiệm cho thấy rằng việc thực hiện cải cách hệ thống các quy trình phần mềm có xu hướng dẫn đến chi phí thấp hơn thông qua cải thiện hiệu quả, tránh làm lại, và phần mềm đáng tin cậy hơn và giá cả hợp lý hơn. Hiệu suất của quy trình do đó ảnh hưởng đến chất lượng sản phẩm phần mềm.

1. Vòng đời phần mềm

Vòng đời phát triển phần mềm (SDLC) là những quy trình dùng để đặc tả và chuyển đổi các yêu cầu thành sản phẩm có thể chuyển giao.

Vòng đời sản phẩm (SPLC) bao gồm vòng đời phát triển phần mềm cộng với:

- Những quy trình phần mềm cung cấp cho việc triển khai, hỗ trợ, bảo trì, nâng cấp và ngưng sử dụng.
- Các quy trình khác: Quản lý cài đặt phần mềm, quy trình quản lý chất lượng phần mềm.

Mô hình vòng đời phát triển phần mềm

- Nhấn mạnh vào tính phụ thuộc và quan hệ lẫn nhau về thời gian (temporal) và logic của những quy trình trong một mô hình.
- Bao gồm các cơ chế kiểm soát đối với việc áp dụng tiêu chuẩn đầu vào và đầu ra. Đầu ra của một quy trình phần mềm thường cung cấp đầu vào cho quy trình phần mềm khác.

2.1 Phân loại quy trình phần mềm

Quy trình cơ bản

- Những quy trình phần mềm dành cho việc phát triển, thi hành và bảo trì phần mềm

Quy trình hỗ trợ

- Quy trình hỗ trợ được áp dụng liên tục trong suốt vòng đời sản phẩm phần mềm để hỗ trợ quy trình cơ bản.
- Bao gồm những quy trình như quản lý cấu hình phần mềm, đảm bảo chất lượng, kiểm tra và xác nhận.

Quy trình tổ chức

- Quy trình đào tạo.
- Quy trình phân tích đo lường.
- Quy trình quản lý cơ sở hạ tầng, quản lý danh mục và tái sử dụng.

Quy trình cho Cross-Project

- Reuse, product-line, domain engineering, liên quan đến nhiều dự án trong tổ chức.

Quy trình phần mềm bổ sung cho những quy trình trên:

- Quy trình quản lý dự án bao gồm các quá trình lập kế hoạch và lập dự toán, quản lý tài nguyên, đo lường và kiểm soát, lãnh đạo, quản lý rủi ro, quản lý các bên liên quan, phối hợp chính, hỗ trợ, tổ chức, và các quy trình cross-project của các dự án phát triển phần mềm và bảo trì.

Quy trình phần mềm có thể được phát triển cho một số nhu cầu đặc biệt: ví dụ một dự án cần bảo mật thông tin thì quy trình phần mềm cũng cần sửa đổi để tuân theo nhu cầu bảo mật thông tin.

2.2 Mô hình vòng đời phần mềm

Mô hình tuyến tính: Các giai đoạn phát triển phần mềm được thực hiện tuần tự

Mô hình lặp lại: Phần mềm được phát triển từng bước tăng chức năng trên chu kỳ lặp đi lặp lại

Mô hình Agile

- Phát triển phần mềm linh hoạt là một cơ chế thực hiện các dự án phần mềm khuyến khích các thay đổi mang tính tiến hóa trong toàn bộ vòng đời của dự án.
- Thường xuyên thuyết minh phần mềm với một đại diện khách hàng hoặc người dùng.
- Trong ngắn hạn lặp đi lặp lại chu kỳ sản xuất các giai đoạn nhỏ của phần mềm

Mô hình phát triển tuyến tính thường phát triển một bộ hoàn chỉnh các yêu cầu phần mềm, đến mức có thể, trong thời gian bắt đầu và lập kế hoạch dự án. Các yêu cầu về phần mềm này sau đó được kiểm soát một cách nghiêm ngặt. Thay đổi các yêu cầu phần mềm được xử lý bởi một ban kiểm soát thay đổi.

Mô hình gia tăng sản xuất gia tăng liên tiếp của bộ làm việc, phần mềm chuyển giao dựa trên phân vùng của các phần mềm yêu cầu phải được thực hiện trong mỗi giai đoạn. Các yêu cầu về phần mềm có thể được kiểm soát một cách chặt chẽ, như trong một mô hình tuyến tính, hoặc có thể có một số linh hoạt trong việc sửa đổi các yêu cầu phần mềm. Mô hình Agile có thể xác định phạm vi sản phẩm và tính năng cao cấp ban đầu; Tuy nhiên, các mô hình Agile được thiết kế để tạo thuận lợi cho quá trình tiến hóa của các yêu cầu phần mềm trong dự án.

2.3 Điều chỉnh quy trình phần mềm

- Điều chỉnh (adaptation) quy trình sẽ tùy vào bối cảnh tổ chức, sự đổi mới công nghệ, kích thước dự án, yêu cầu quản lý, thực tế ngành công nghiệp phần mềm ở địa phương.
- Quá trình điều chỉnh mô hình vòng đời phần mềm có thể bao gồm: thêm thông tin vào quy trình phần mềm hoặc loại bỏ những quy trình không áp dụng được

2.4 Liên hệ với thực tế

Những yếu tố được xem xét khi định nghĩa và thiết kế một mô hình vòng đời phần mềm bao gồm các yêu cầu phù hợp với các tiêu chuẩn, hướng dẫn và chính sách; nhu cầu của khách hàng; tầm quan trọng của các sản phẩm phần mềm.

Những quy trình phần mềm (ví dụ như quản lý cấu hình, xây dựng và thử nghiệm) có thể được điều chỉnh để tạo thuận lợi cho hoạt động hỗ trợ, bảo trì, chuyển đổi và kết thúc của phần mềm.

1. Đánh giá và cải thiện quy trình phần mềm

Chủ đề này nói về đánh giá quy trình phần mềm, phương pháp đánh giá quy trình phần mềm, quá trình cải thiện mô hình phần mềm, và cách thức xếp hạng. Đánh giá quy trình phần mềm được sử dụng để đánh giá các hình thức và nội dung của một quy trình phần mềm, mà có thể được xác định bởi một bộ tiêu chuẩn hóa các tiêu chí. Trong một số trường hợp, các điều khoản "quy trình thẩm định" và "đánh giá khả năng" được sử dụng thay vì đánh giá quy trình

Hiệu suất đánh giá thường được thực hiện trong một tổ chức để xác định các quy trình phần mềm cần cải thiện. Đánh giá quy trình được thực hiện ở các cấp độ của toàn bộ tổ chức, đơn vị tổ chức trong các tổ chức, và các dự án cá nhân. Đánh giá có thể liên quan đến các vấn đề như đánh giá liệu các tiêu chí quy trình phần mềm đầu vào, đầu ra được đáp ứng, để xem xét các yếu tố rủi ro và quản lý rủi ro, hoặc để rút ra các bài học kinh nghiệm. Đánh giá quy trình được thực hiện bằng cách sử dụng cả một mô hình đánh giá và phương pháp đánh giá. Mô hình này có thể cung cấp một chuẩn mực để so sánh điểm chuẩn giữa các dự án trong một tổ chức và giữa các tổ chức.

Quy trình kiểm tra khác với quy trình đánh giá phần mềm. Việc đánh giá phần mềm sẽ xác định được tính khả thi và sự thay đổi của phần mềm để biết được phần mềm cải thiện như thế nào. Kiểm tra phần mềm để xác định rằng các tiêu chuẩn có hợp lý hay không. Đây là quá trình thu thập bằng chứng và đánh giá bằng chứng về các hoạt động của hệ thống thông tin đã được tổ chức. Việc đánh giá các bằng chứng phải đảm bảo rằng hoạt động của hệ thống ấy phải bảo mật, chính trực, hữu hiệu, và hiệu quả nhằm đạt được các mục tiêu của tổ chức đã đề ra.

3.1 Những mô hình đánh giá quy trình phần mềm

Việc đánh giá mô hình quy trình phần mềm mang lại những kinh nghiệm thực tiễn tốt. Nó có thể áp dụng thêm trong những việc bảo trì phần mềm, quản lý dự án phần mềm, quản lý kỹ thuật hệ thống, hoặc quản lý nguồn nhân lực.

3.2 Những phương pháp đánh giá quy trình phần mềm

- 1 phương pháp đánh giá quy trình phần mềm có thể dựa vào việc định tính hoặc định lượng. Định tính được xác định qua các đánh giá của các chuyên gia có kinh nghiệm. Định lượng là dựa vào các chỉ số đo và tính toán được. Lấy ví dụ là dựa vào các thông số như số lỗi, số issue được tạo ra, thời gian trung bình cần thiết để hiệu chỉnh 1 lỗi, vv....
- Một phương pháp điển hình của việc đánh giá quy trình phần mềm bao gồm lập kế hoạch, tìm hiểu thực tế (bằng cách thu thập dữ liệu thông qua bảng câu hỏi, phỏng vấn và quan sát thực tế công việc), phân tích và báo cáo.
- Chất lượng của các kết quả đánh giá phụ thuộc vào phương pháp đánh giá, tính toàn vẹn và chất lượng của dữ liệu thu được, khả năng của các nhóm đánh giá và tính khách quan của thông tin mà họ đánh giá được, và các thông số trong quá trình kiểm tra đánh giá. Kết quả mong muốn của việc đánh giá quá trình phần mềm là đạt được cái nhìn toàn vẹn rằng phần mềm đang ở trạng thái như thế nào và cung cấp cơ sở cho việc cải tiến quy trình (nếu có)

3.3 Những mô hình cải thiện quy trình phần mềm

Cải thiện quy trình phần mềm nhấn mạnh sự cải thiện liên tục để phần mềm tiến tới trạng thái hoàn thiện. Việc cải tiến quy trình phần mềm có thể bao gồm các quy trình con về tính toán, phân tích, và thay đổi. Mô hình Plan-Do-Check-Act là một cách tiếp cận nổi tiếng để cải tiến quy trình phần mềm. Các hoạt động cải tiến bao gồm việc xác định và ưu tiên những điểm cần cải tiến trước; đề xuất các phương án để cải tiến, đánh giá sự cải thiện so với kết quả quá trình trước đó hoặc so với 1 mô hình mẫu nào đó. Mô hình Plan-Do-Check-Act có thể được áp dụng cho việc phòng ngừa các lỗi xảy ra

3.4 Xếp hạng quy trình phần mềm

Để đánh giá khả năng và sự đổi mới (tiến bộ, thay đổi tích cực) của 1 quy trình phần mềm thì sử dụng bảng đánh giá gồm 5 hoặc 6 cấp độ

Bảng thể hiện các cấp độ đánh giá quy trình phần mềm

Trong bảng 8.1,

- Mức 0 chỉ ra rằng một quy trình phần mềm được thực hiện không đầy đủ hoặc không thể thực hiện được.
- Cấp độ 1, một quy trình phần mềm đang được thực hiện.

- Ở cấp độ 2, một quy trình phần mềm đang được thực hiện và nó có khả năng cho ra các sản phẩm trung gian mà có thể hiển thị, thấy được
- Ở cấp độ 3, những quy trình đã hoàn thành ở cấp độ 2 được xác định rõ (xác định thông qua các chính sách và thủ tục của tổ chức). Cấp độ 3 thì quy trình phần mềm tiếp tục được cải thiện và nó cung cấp một cơ sở cho việc cải tiến quy trình, vì ở cấp độ này các chuyên gia đã có thể đánh giá và đo đạc được các thông số chất lượng quy trình phần mềm
- Cấp độ 4, các biện pháp định lượng, phân tích thống kê có thể được áp dụng và được sử dụng để đánh giá quy trình
- Cấp độ 5, các cơ chế cho việc cải tiến quy trình phần mềm liên tục được áp dụng.

1. Cách đo lường phần mềm

Chủ đề này nói về quy trình phần mềm và đo lường sản phẩm, chất lượng của các kết quả đo lường, mô hình thông tin phần mềm, và kỹ thuật đo lường quy trình phần mềm. Trước khi một quy trình mới được thực hiện hoặc một quy trình hiện tại được sửa đổi, kết quả đo lường cho tình hình hiện nay nên được thu lại để cung cấp một đường cơ sở để so sánh giữa tình hình hiện tại và tình hình mới.

4.1 Cách đo lường quy trình phần mềm và sản phẩm

Đo lường sản xuất phần mềm có liên quan với việc xác định hiệu quả và hiệu suất của một quy trình phần mềm, hoạt động, hoặc tác vụ. Hiệu quả của một quá trình phần mềm, hoạt động, hoặc tác vụ là tỷ lệ của các nguồn tài nguyên thực sự tiêu thụ / tài nguyên dự kiến trong hoàn thành một quy trình phần mềm, hoạt động, hoặc công việc.

Chi phí tương đương là các biện pháp chính tính tài nguyên cho hầu hết các quy trình phần mềm, hoạt động và nhiệm vụ; nó được đo bằng đơn vị như person-hours, person-days, person-hours, person-days, staff-weeks, or staff-months effort hoặc tương đương đơn vị tiền tệ-chẳng hạn như Euro hoặc đô la.

Hiệu quả là tỷ lệ các sản lượng thực tế của đầu ra dự kiến sản xuất bằng quy trình phần mềm, hoạt động hoặc nhiệm vụ; Ví dụ, số lượng thực tế của phát hiện lỗi và sửa chữa trong phần mềm thử nghiệm để dự kiến số lỗi được phát hiện và sửa chữa dựa trên các dữ liệu lịch sử cho dự án tương tự. Lưu ý rằng đo lường tính hiệu quả quy trình phần mềm yêu cầu đo lường của các thuộc tính của các sản phẩm có liên quan; Ví dụ, đo lường của lỗi phần mềm được phát hiện và sửa chữa trong thời gian kiểm thử phần mềm.

Nguyên nhân của hiệu quả thấp một quy trình phần mềm, hoạt động, hoặc công việc được thực hiện có thể bao gồm một hoặc nhiều vấn đề sau: thiếu đầu vào công việc sản phẩm, nhân viên thiếu kinh nghiệm, không đầy đủ các công cụ và cơ sở hạ tầng, học một quy trình mới, một sản phẩm phức tạp, hoặc không quen sản phẩm. Hiệu quả và hiệu suất của quy

trình phần mềm thực hiện cũng bị ảnh hưởng (tích cực hoặc tiêu cực) bởi yếu tố chẳng hạn như doanh thu trong nhân sự phần mềm, thay đổi lịch trình, một đại diện khách hàng mới, hoặc một chính sách mới tổ chức.

Tiêu chuẩn hóa các định nghĩa và tính quy tắc cho các đo lường của quy trình phần mềm và các sản phẩm làm việc là cần thiết để cung cấp kết quả đo lường tiêu chuẩn qua các dự án trong một tổ chức, để có một kho lưu trữ của dữ liệu lịch sử có thể được phân tích để xác định quy trình phần mềm cần phải được cải thiện, và để xây dựng các mô hình tiên đoán dựa trên tích lũy dữ liệu.

4.2 Chất lượng của kết quả sự đo lường

Chất lượng của các kết quả đo lường sản xuất chủ yếu được xác định bởi độ tin cậy và tính hợp lệ của các kết quả đo. Đo đạc không đáp ứng các tiêu chí chất lượng có thể dẫn đến không đúng cách diễn giải và sáng kiến cải thiện quá trình phần mềm bị lỗi. Các thuộc tính mong muốn của các phép đo phần mềm bao gồm một bộ sưu tập, phân tích, và trình bày cùng với một sự tương quan lớn giữa nguyên nhân và hiệu quả.

4.3 Những mô hình thông tin phần mềm

Mô hình thông tin phần mềm cho phép mô hình hóa, phân tích và dự đoán của quy trình phần mềm và thuộc tính sản phẩm phần mềm để cung cấp câu trả lời cho câu hỏi có liên quan và đạt được mục tiêu cải thiện quá trình và sản phẩm.

Dữ liệu cần thiết có thể được thu thập và giữ lại trong một kho lưu trữ; dữ liệu có thể được phân tích và mô hình có thể được xây dựng. Xác nhận và sàng lọc các mô hình thông tin phần mềm xảy ra trong thời gian dự án phần mềm và sau khi dự án được hoàn thành để đảm bảo mức độ chính xác. Mô hình thông tin phần mềm cũng có thể được phát triển cho các ngôn ngữ khác hơn so với dự án phần mềm. Ví dụ, một mô hình thông tin phần mềm có thể được phát triển cho quá trình áp dụng cho một tổ chức, chẳng hạn như quản lý cấu hình phần mềm hoặc đảm bảo chất lượng quy trình phần mềm ở cấp độ tổ chức.

Một mô hình thông tin phần mềm được phát triển bằng cách thiết lập một biến đổi đưa ra giả thuyết của các biến đầu vào thành đầu ra mong muốn. Ví dụ, sản phẩm kích thước và độ phức tạp có thể được chuyển đổi thành ước tính effort cần thiết để phát triển một sản phẩm phần mềm bằng cách sử dụng một phương trình hồi quy phát triển từ các dữ liệu quan sát từ các dự án trong quá khứ. Một mô hình được kiểm định bằng cách điều chỉnh các tham số trong mô hình để phù hợp với các kết quả quan sát từ các dự án trong quá khứ.

- Một mô hình được đánh giá bằng cách so sánh các kết quả tính đến các kết quả thực tế cho một tập khác nhau của dữ liệu tương tự. Hiện có ba kết quả có thể đánh giá:

- 1. Kết quả tính toán cho một tập dữ liệu khác nhau rất khác nhau từ những kết quả thực tế cho bộ dữ liệu, trong trường hợp các mô hình có nguồn gốc không áp dụng cho thiết lập các dữ liệu mới và không nên áp dụng để phân tích hoặc đưa ra dự đoán cho các dự án trong tương lai;
- 2. Kết quả tính toán cho một tập dữ liệu mới đang gần kết quả thực tế cho bộ dữ liệu, trong đó trường hợp điều chỉnh nhỏ được thực hiện điều chỉnh các thông số của mô hình để cải thiện sự thỏa thuận (improve agreement).
- 3. Kết quả tính toán cho dữ liệu mới và thiết lập bộ dữ liệu tiếp theo là rất gần và không có điều chỉnh để các mô hình là cần thiết.
- Các đánh giá liên tục của các mô hình có thể chỉ ra một nhu cầu cho điều chỉnh theo thời gian vì bối cảnh trong đó các mô hình là ứng dụng thay đổi.

Phương pháp mục tiêu/câu hỏi/số liệu, Goals/Questions/Metrics (GQM) ban đầu được dự định để thiết lập đo lường hoạt động, nhưng nó cũng có thể được sử dụng để hướng dẫn phân tích và cải thiện quy trình phần mềm.

Nó có thể được sử dụng để hướng dẫn hướng phân tích xây dựng mô hình thông tin phần mềm; kết quả thu được từ phần mềm thông tin mẫu có thể được sử dụng để hướng dẫn quá trình cải tiến.

Ví dụ sau minh họa các ứng dụng của phương pháp GQM:

Goal: Reduce the average change request processing time by 10% within six months. •

Question 1-1: What is the baseline change request processing time?

- Metric 1-1-1: Average of change request processing times on starting date
- Metric 1-1-2: Standard deviation of change request processing times on starting date
- Question 1-2: What is the current change request processing time?
- Metric 1-2-1: Average of change request processing times currently
- Metric 1-2-2: Standard deviation of change request processing times currently

4.4 Những kỹ thuật đo lường quy trình phần mềm

Kỹ thuật đo lường quy trình phần mềm được sử dụng để thu thập dữ liệu quy trình và dữ liệu sản phẩm, chuyển đổi dữ liệu vào thông tin hữu ích và phân tích thông tin để xác định các hoạt động quy trình là ứng cử viên cho việc cải thiện. Trong một số trường hợp, quy trình phần mềm mới có thể là cần thiết.

Quá trình đo lường kỹ thuật cũng cung cấp thông tin cần thiết để đo lường tác động của quá trình cải tiến sáng kiến. Quá trình đo lường kỹ thuật có thể được sử dụng để thu thập dữ liệu định lượng và chất lượng.

- Các công cụ đơn giản để định lượng: check sheets, Pareto diagrams, histograms, scatter diagrams, run charts, control charts, and cause-and-effect diagrams
- Dữ liệu được thu thập bằng cách sử dụng định lượng quá trình đo lường kỹ thuật cũng có thể được sử dụng như đầu vào để mô phỏng các mô hình. Các mô hình này có thể được sử dụng để đánh giá tác động của phương pháp tiếp cận khác nhau để cải thiện quá trình phần mềm.
- Phân loại khiếm khuyết trực giao (ODC) có thể được sử dụng để phân tích định lượng quá trình đo lường dữ liệu.
- Định tính quá trình đo lường kỹ thuật — bao gồm các cuộc phỏng vấn, câu hỏi, và ý kiến chuyên gia - có thể được sử dụng để làm tăng thêm kỹ thuật đo lường định lượng quy trình. Kỹ thuật sự đồng thuận nhóm, có thể được sử dụng để có được sự đồng thuận giữa các nhóm của các bên liên quan

5 Những công cụ quy trình kỹ thuật phần mềm BPMN, IDEF0 diagrams, Petri nets, and [UML](#) activity diagrams, spreadsheet. Computer-Assisted Software Engineering (CASE) tools

Chú giải

CIA

Confidentiality, Integrity, and Availability (Độ tin cậy, Tính toàn vẹn, tính tiện lợi)

1. Chương 1: Yêu cầu phần mềm

DAG

Directed Acyclic Graph (Đồ thị có hướng và không có chu trình)

1. Chương 1: Yêu cầu phần mềm

FSM

Functional Size Measurement (Thước đo chức năng)

1. Chương 1: Yêu cầu phần mềm

INCOSE

International Council on Systems Engineering ()

1. Chương 1: Yêu cầu phần mềm

SysML

Systems Modeling Language (Ngôn ngữ mô hình hóa hệ thống)

1. Chương 1: Yêu cầu phần mềm

UML

Unified Modeling Language (Ngôn ngữ mô hình hóa)

- 1. Chương 1: Yêu cầu phần mềm
- 8. Chapter 9: Software Engineering Models and Methods
- 3. Chương 3: Xây dựng phần mềm 11. Chương 8: Quy trình công nghệ phần mềm