TRƯỜNG ĐẠI HỌC AN GIANG

KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

ThS. NGUYỄN THỊ MỸ TRUYỀN

AN GIANG, 1-2022

LỜI NÓI ĐẦU

Ngày nay, ngành công nghiệp phần mềm trên thế giới đang rất phát triển. Nhiều người đã thống nhất rằng công nghiệp phần mềm đã trở thành một mũi nhọn để con người tiến nhanh vào nền kinh tế tri thức. Tuy nhiên, việc tạo ra một sản phẩm phần mềm không phải là một công việc đơn giản và dễ dàng. Để làm ra một sản phẩm phức tạp và ít hữu hình này đòi hỏi những người tham gia phát triển phải có những phần kiến thức chuyên môn sâu - đủ để tạo ra được một sản phẩm phần mềm chất lượng, đáp ứng yêu cầu người dùng và mang lại hiệu quả kinh tế cao.

Vì vậy kiến thức về công nghệ phần mềm là một phần rất cần thiết đối với những người bắt đầu tiếp cận với lĩnh vực công nghệ thông tin nói chung và đặc biệt là đối với sinh viên ngành Công nghệ thông tin và Kỹ thuật phần mềm nói tiêng tại Đại học An Giang. Tài liệu nhập môn công nghệ phần mềm được biên soạn nhằm mục tiêu trang bị kiến thức nền tảng giúp sinh viên tiếp cận với chuyên môn được tốt hơn. Nội dung tài liệu giới thiệu tổng quan kiến thức về các giai đoạn trong qui trình phát triển phần mềm như lập kế hoạch, quản lý, phân tích, thiết kế, viết mã, kiểm thử,… dựa trên những phương pháp, kỹ thuật đã được thực hiện trong lĩnh vực phát triển phần mềm. Đồng thời, kiến thức của môn học này được dùng làm cơ sở để sinh viên học tiếp các môn chuyên ngành ở các học kì tiếp theo. Nội dung tài liệu bao gồm 5 chương:

Chương 1 trình bày về quá trình phát triển của công nghệ phần mềm và giới thiệu các mô hình phát triển phần mềm đã và đang áp dụng.

Chương 2 giới thiệu các kiến thức cơ bản về quản lý dự án phần mềm và các hoạt động được thực hiện trong việc quản lý dự án phần mềm.

Chương 3 giới thiệu các hoạt động phân tích yêu cầu và giới thiệu một số mô hình được thực hiện trong quá trình phân tích phần mềm.

Chương 4 giới thiệu các hoạt động thiết kế và mã hóa (code).

Chương 5 giới thiệu các khái niệm, phương pháp và kỹ thuật trong kiểm thử phần mềm, trong đó giới thiệu một số ví dụ kiểm thử theo phương pháp white box dùng kỹ thuật đồ thị dòng chảy để thiết kế ca kiểm thử.

Do lần đầu tiên biên soạn tài liệu nên không thể tránh khỏi những hạn chế và thiếu sót. Rất mong nhận được sự góp ý chân thành từ các bạn sinh viên, các đồng nghiệp và các chuyên gia trong chuyên môn để tài liệu được hoàn thiện tốt hơn.

An Giang, tháng 3 năm 2017

Người biên soạn

ThS. Nguyễn Thị Mỹ Truyền

MỤC LỤC

Trang

CHƯƠNG 1
TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

Trong cuộc sống hiện đại ngày nay, phần mềm có mặt khắp nơi để trợ giúp con người dưới hình thức của chiếc máy tính (computer) hay các thiết bị điện tử. Phần mềm đã góp phần làm cho cuộc sống của chúng ta hiện đại hơn, nhiều công việc được trở nên dễ dàng và đơn giản hơn. Để hiểu hơn về lĩnh vực phát triển phần mềm, chương tổng quan này sẽ trình bày vai trò quan trọng của phần mềm trong cuộc sống và sự phát triển của lĩnh vực phần mềm. Nội dung chương này sẽ tập trung giới thiệu một số khái niệm và các mô hình phát triển phần mềm phổ biến. Kiến thức về các mô hình phát triển phần mềm có thể giúp sinh viên hiểu được các mô hình phát triển phần mềm cơ bản, phân biệt được sự khác nhau giữa các mô hình, nắm được ưu và nhược điểm của từng mô hình và có thể xác định loại hệ thống nào nên áp dụng mô hình phát triển phần mềm nào cho phù hợp.

1.1 GIỚI THIỆU TỔNG QUAN

1.1.1 Sự cần thiết của công nghệ phần mềm

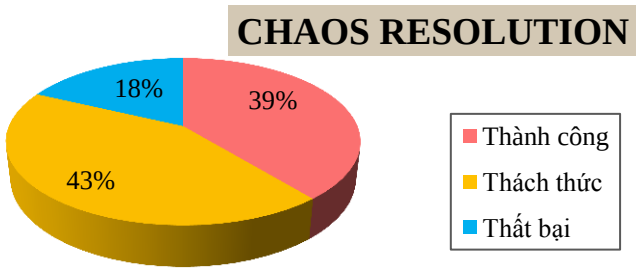
Ngày nay, phần mềm giữ một vai trò rất quan trọng trong các hệ thống máy tính. Phần mềm vừa là nền tảng của nhiều hoạt động xã hội và tổ chức, vừa góp phần tạo ra phong cách làm việc hiện đại, hiệu quả và tăng năng suất lao động đáng kể cho con người. Ngày càng nhiều hệ thống được phần mềm điều khiển, trợ giúp. Ứng dụng phần mềm có mặt trong mọi lĩnh vực xã hội, kinh tế, chính trị, quân sự…. Đến thế kỉ thứ 21, phần mềm đã trở thành nhân tố quyết định về những cơ hội mới trong mọi lĩnh vực từ giáo dục cơ bản cho đến công nghệ phát sinh.

Tuy nhiên, sự phát triển của phần mềm luôn phải đối đầu với nhiều khó khăn và thách thức. Trước nhất là phần cứng và hệ điều hành đã ảnh hưởng không nhỏ đến sự phát triển của phần mềm. Ngoài ra, yếu tố kinh tế luôn là vấn đề quan trọng trong quá trình phát triển phần mềm. Để xây dựng một hệ thống phần mềm, chúng ta thường phải đầu tư một khoản ngân sách khá lớn. Theo nhiều thống kê cho thấy, chi phí cho việc xây dựng phần mềm chiếm một phần đáng kể của GNP (chỉ tiêu kinh tế) ở tất cả các nước phát triển. Để giảm bớt khó khăn này nhà phát triển phần mềm thường dùng giải pháp lựa chọn kỹ thuật thực hiện nhanh hơn để giảm giá thành. Mặc khác, kỹ thuật mới mang lại không ít khó khăn mới cho các công ty phần mềm như khó bảo trì, thời gian huấn luyện dài, thiếu kinh nghiệm làm việc với kỹ thuật mới,… Theo kết quả thực hiện của hơn 9000 dự án hoàn thành trong năm 2004, tỉ lệ thành công một cách trọn vẹn chỉ chiếm khoảng 29% số lượng các dự án phần mềm.

Cuộc khủng hoảng phần mềm bắt đầu từ năm 1970. Vào thời điểm này rất nhiều dự án phần mềm vượt quá kế hoạch và ngân sách cho phép, một số phần mềm hư hỏng nặng (Ngô Trung Việt & Nguyễn Kim Ánh, 2003). Tiêu biểu là sự cố Y2K (time bomb), ước tính quá trình xử lý sự cố Y2K đã làm tiêu tốn của nước Anh 50 tỉ $, đồng thời cần tới 300.000 người để giải quyết sự cố; hay lỗi phần mềm làm phá

hủy tên lửa Patriot của Mỹ trong cuộc chiến vùng vịnh năm 1991; Tên lửa đẩy Ariane 5-ESCA thế hệ mới của Cơ quan Vũ trụ châu Âu, mang theo hai vệ tinh, đã vỡ tan và rơi xuống biển trong 3 phút; Sân bay London tê liệt vì lỗi phần mềm máy tính có từ thập niên 1960: Phần mềm điều khiển các chuyến bay này bị sự cố một giờ mới khôi phục lại, đã ảnh hưởng đến hoạt động cất, hạ cánh ở 5 sân bay của Anh. Hàng loạt chuyến bay bị hoãn, ảnh hưởng đến hơn 10.000 hành khách, …

Thống kê về tỷ lệ thành công của dự án phần mềm (CHAOS Manifesto, 2013) chỉ ra rằng trong năm 2012 chỉ có 39% dự án phần mềm thành công xét trên ba ràng buộc: chi phí, thời gian và phạm vi. Trong khi đó có 43% dự án phần mềm gặp thách thức như không hoàn thành đúng kế hoạch, hoặc là vượt kinh phí của dự án hoặc thiếu các chức năng của phần mềm như đã đề ra từ ban đầu. Số dự án thất bại chiếm 18%, nghĩa là những dự án này bị hủy trước khi kết thúc, hoặc sản phẩm không được chuyển giao hoặc đã chuyển giao nhưng không sử dụng được.



Hình 1. Tỉ lệ thành công, thách thức và thất bại năm 2012

Theo CHAOS năm 2013 thì số liệu thống kê tỉ lệ thành công, thách thức và thất bại của các dự án phần mềm như bảng 1 (CHAOS Manifesto, 2013)

Bảng 1. Tỉ lệ thành công, thách thức và thất bại của các dự án

Năm	Thành công	Thất bại	Thách thức
1998	26%	28%	46%
2000	28%	23%	49%
2004	29%	18%	53%
2006	35%	19%	46%
2008	32%	24%	44%
2010	37%	21%	42%
2012	39%	18%	43%

Đối với những hệ phần mềm lớn như hệ DOS của IBM hay hệ mềm dùng cho chương trình không gian của Mĩ, thì số lượng nhân sự tham gia một dự án phần mềm không phải vài người thực hiện trong vài tháng mà là hàng trăm người làm trong nhiều năm, có khi kéo dài đến mười năm. Hoặc với chương trình điều khiển tổng đài điện thoại hiện bán khắp nơi trên thế giới đã được phát triển trong vòng 5 năm, bao gồm hàng chục nghìn đơn vị người làm. Điều đó cho thấy rằng việc quản lý nhân sự và thời gian của những dự án phần mềm là những khó khăn có thể ảnh hưởng lớn đến sự thành công hay thất bại của dự án.

Một phần mềm máy tính được xem là thành công khi nó đáp ứng tốt tất cả các yêu cầu của người dùng, hoạt động tốt trong suốt thời gian dài, phần mềm dễ sửa lỗi, dễ sử dụng, có thể nâng cấp, phát triển dễ dàng… Nhưng khi phần mềm hỏng hoặc xảy ra lỗi thì rất khó sửa chữa, thậm chí rất khó sử dụng tiếp, mọi điều tệ hại nhất đều có thể xảy ra.

Sự thất bại của những dự án phần mềm trong giai đoạn này xuất phát từ nhiều nguyên nhân khách quan và chủ quan như: Tính chuyên nghiệp trong phát triển phần mềm chưa cao, chất lượng phần mềm chưa đáp ứng tốt yêu cầu của người sử dụng, rủi ro và phức tạp trong phát triển phần mềm là rất lớn, công nghệ thay đổi liên tục, thiết bị ngày càng hiện đại, nhu cầu con người ngày càng cao và cấp bách…

Trên thực tế, phần mềm được làm ra rất nhiều, tuy nhiên không phải tất cả các phần mềm đó đều đáp ứng đúng và đủ yêu cầu của người dùng. Bởi vì nhu cầu sử dụng phần mềm là rất lớn ở nhiều lĩnh vực ngành nghề, với nhiều qui mô và cấp độ khác nhau. Ngoài ra, các vấn đề chất lượng phần mềm, tính thoái hóa của phần mềm cũng ảnh hưởng không ít đến phần mềm,…Có quá nhiều yếu tố chi phối đến phần mềm. Do đó, để tạo được sự thành công trong xây dựng phần mềm, chúng ta cần phải có phương pháp, nguyên tắc trong thiết kế và xây dựng, cần phải có một phương pháp tiếp cận thật sự khoa học trong việc phát triển phần mềm. Công nghệ phần mềm đã ra đời đáp ứng được nhiều yêu cầu đặt ra trong lĩnh vực phát triển phần mềm với những mục tiêu sau:

- Nâng cao chất lượng sản phẩm và tính chuyên nghiệp trong phát triển phần mềm.
- Mang lại phần mềm có chất lượng cao trong khoảng thời gian và chi phí hợp lý.
- Xây dựng kiến trúc phần mềm tốt đối với người phát triển.
- Đáp ứng kịp thời nhu cầu phần mềm trong cuộc sống hiện đại.

Đối với các chuyên ngành liên quan đến công nghệ thông tin, kết quả đầu ra của các chương trình này đòi hỏi sinh viên phải có kiến thức về quá trình phát triển phần mềm và có khả năng tham gia thực hiện các giai đoạn từ phân tích, thiết kế, viết code,… nhằm tạo ra một sản phẩm phần mềm đáp ứng yêu cầu thực tế. Để có thể tạo ra được một sản phẩm phần mềm thì sinh viên phải học qua nhiều môn học từ cơ bản đến nâng cao. Trong đó học phần nhập môn công nghệ phần mềm là một trong những môn mở đầu về kiến thức chuyên môn giúp sinh viên có được cái nhìn tổng thể về tất cả các hoạt động trong qui trình phát triển phần mềm, thấy được mối quan hệ giữa các giai đoạn trong qui trình. Cụ thể hơn, môn học giúp sinh viên hiểu được mỗi giai đoạn sẽ thực hiện công việc gì, sử dụng phương pháp, công cụ nào để thực hiện. Thông qua môn học này, sinh viên hiểu được sự cần thiết phải học tốt các môn học tiếp theo như ngôn ngữ lập trình, phân tích, thiết kế, kiểm thử, bảo trì, quản lý dự án phần mềm,… và hiểu được những kiến thức đó sẽ góp phần vào giai đoạn nào của

qui trình phát triển phần mềm mà học phần nhập môn công nghệ phần mềm đã giới thiệu.

1.1.2 Đặc điểm của phần mềm

❖ ***Chi phí phần mềm***

Công nghiệp phần mềm đã trở thành ngành khổng lồ bởi vì chi phí phần mềm thường chiếm phần lớn chi phí của cả hệ thống máy tính. Chi phí phần mềm trên máy tính cá nhân thường lớn hơn nhiều so với chi phí phần cứng. Ví dụ một đơn vị sử dụng 100 máy tính (chi phí phần cứng là mua 100 máy tính và các thiết bị khác), tất cả các máy tính đều cài hệ điều hành bản quyền. Giả sử mỗi máy tính cài 20 phần mềm ứng dụng thường dùng phải trả tiền bản quyền trên mỗi năm + mua bản quyền phần mềm đặc thù của đơn vị. Như vậy chi phí phần mềm cực lớn so với phần cứng.

❖ ***Phần mềm không mòn cũ nhưng thoái hóa***

Đặc trưng của phần mềm là **không mòn cũ** nhưng **thoái hóa** theo thời gian. Trên thực tế, môi trường sử dụng luôn có những thay đổi, nhu cầu người dùng ngày càng cao, hoặc trong quá trình sử dụng phần mềm cần được nâng cấp, thay đổi nhiều lần dẫn đến lỗi phát sinh tăng quá mức kiểm soát, do đó phần mềm trước đây trở nên lỗi thời, thoái hóa, không dùng đến. Ví dụ phần mềm hệ điều hành Windows đang sử dụng thì trước đây người ta dùng Windows 98, Windows 2000, Windows XP.

❖ ***Sự phức tạp và tính thay đổi để thích ứng***

Một đặc trưng khác thuộc về bản chất của phần mềm là **sự thay đổi**. Thế giới thực luôn thay đổi và phát triển theo thời gian dẫn đến những nghiệp vụ thay đổi, nhu cầu con người thay đổi thì phần mềm phải được thay đổi theo để thích ứng với những thay đổi đó, thích ứng với môi trường vận hành. Quá trình phát triển về mặt công nghệ có sự thay đổi phần cứng, phần mềm như hệ điều hành thay đổi từ Windows XP qua Windows 7 hay Linux thì các phần mềm nghiệp vụ cũng phải thay đổi theo để tương thích với hệ thống mới.

❖ ***Phần mềm không được tạo ra tự động***

Phần mềm **không được lắp ráp, sản xuất từ mẫu sẵn có**. Người ta không thể cho ra đời hàng loạt các sản phẩm phần mềm một cách máy móc như các sản phẩm khác. Một sản phẩm phần mềm là kết quả làm việc trí óc, là kết tinh từ chất xám của những nhóm người làm việc chuyên nghiệp.

❖ ***Phần mềm được phát triển theo nhóm***

Đặc thù của việc phát triển sản phẩm phần mềm là phải **làm việc theo nhóm**. Sản phẩm của ngành công nghiệp này có thể giúp con người thực hiện một số chức năng trong công việc một cách hiệu quả và tin cậy.

Thực tế không có một phương pháp công nghệ phần mềm đủ tính phổ quát để áp dụng thích hợp cho nhiều hệ thống phần mềm khác nhau và những công ty khác nhau. Tuy nhiên có 4 vấn đề liên quan ảnh hưởng đến các loại phần mềm như sau

Tính không đồng nhất ngày càng tăng, cụ thể là hệ thống phải vận hành tốt trong môi trường mạng với đặc tính của hệ thống phân tán trên nhiều thiết bị khác nhau như máy tính, điện thoại và các thiết bị khác. Để chạy tốt trên các thiết bị đòi hỏi phải tích hợp phần mềm mới với hệ thống kế thừa trước đó được viết trên nhiều ngôn ngữ lập trình khác nhau. Thách thức ở đây là phải phát triển các kỹ tuật xây dựng phần mềm đáng tin cậy, đủ linh hoạt để thích ứng với sự không đồng nhất đó.

Sự thay đổi nghiệp vụ và xã hội quá nhanh đặc biệt là sự phát triển của các nền kinh tế mới nổi và sự sẵn sàng của các công nghệ mới. Nên họ cần thay đổi các phần mềm hiện tại và nhanh chóng phát triển các phần mềm mới. Trong khi nhiều công nghệ phần mềm truyền thống đã tiêu tốn quá nhiều thời gian và quá lâu so với kế hoạch để chuyển giao hệ thống mới. Họ cần có sự tiến hoá để giảm thời gian chuyển giao phần mềm đến khách hàng.

Bảo mật và tin cậy, Khi phần mềm đan xem vào tất cả các mặt trong đời sống của chúng ta thì độ tin cậy của phần mềm là rất quan trọng. Điều này càng đúng hơn đối với các hệ thống truy cập từ xa thông qua giao diện web. Khi đó cần đảm bảo rằng những người dùng độc hại không thể tấn cộng được vào phần mềm và thông tin của hệ thống phải được duy trì bảo mật cao.

Quy mô phần mềm được phát triển trên phạm vi rất rộng, từ những hệ thống nhúng rất nhỏ trên các thiết bị cầm tay hay các thiết bị đeo đến các hệ thống trên internet, nền tảng đám mây nhằm phục vụ cộng đồng toàn cầu.

Để giải quyết các thách thức này, chúng ta cần đến các công cụ, kỹ thuật mới cũng như dùng những cách sáng tạo trong việc kết hợp và sử dụng các phương pháp công nghệ phần mềm đang tồn tại hiện nay.

1.2 ĐỊNH NGHĨA

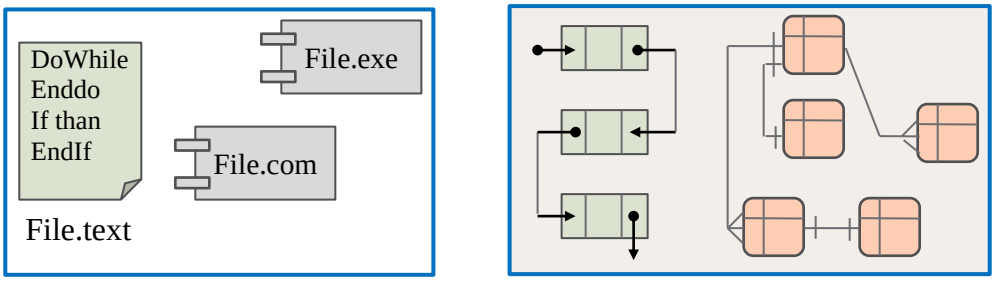
1.2.1 Định nghĩa phần mềm

Phần mềm hay chương trình máy tính - là một tập hợp những câu lệnh được viết bằng một hoặc nhiều ngôn ngữ lập trình theo một trật tự xác định nhằm tự động thực hiện một số chức năng hoặc giải quyết một bài toán nào đó.

Đối tượng chính của công nghệ phần mềm là sản xuất ra các sản phẩm phần mềm. **Sản phẩm phần mềm** là sản phẩm đạt được sau một tiến trình phát triển phần mềm, là các phần mềm được phân phối cho khách hàng cùng với các tài liệu mô tả, phương thức cài đặt và cách thức sử dụng chúng. Gồm 3 phần (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009):

a. Tập các lệnh (c*hương trình máy tính* - program) trên máy tính khi được thực hiện sẽ tạo ra các dịch vụ và đem lại những kết quả mong muốn cho người dùng, thường gồm các tập tin chứa mã lệnh có thể thực thi được.

b. *Các cấu trúc dữ liệu* làm cho chương trình thao tác hiệu quả với các thông tin thích hợp và nội dung thông tin được số hóa – nằm bên trong của chương trình máy tính.



Hình 2. Chương trình và cấu trúc dữ liệu

c. *Các tài liệu liên quan* để mô tả thao tác, cách sử dụng và bảo trì phần mềm

- Tài liệu phát triển: Tài liệu đặc tả, thiết kế, kiểm thử, kế hoạch quản lý.
- Tài liệu hướng dẫn sử dụng (support).
- Tài liệu tham khảo kỹ thuật.

1.2.2 Định nghĩa công nghệ phần mềm

Công nghệ phần mềm là một **nguyên tắc công nghệ** (Engineering discipline) tập trung vào **tất cả các khía cạnh sản xuất phần mềm** từ các giai đoạn đầu của đặc tả hệ thống đến giai đoạn bảo trì hệ thống (sau khi được đưa vào sử dụng).

Nguyên tắc công nghệ Các kỹ sư phần mềm phải áp dụng những lý thuyết, phương pháp và công cụ một cách thích hợp, có lựa chọn và luôn cố gắng để tìm những giải pháp giải quyết vấn đề kể cả trong những tình huống không thể áp dụng được những lý thuyết và phương pháp.

Tất cả các khía cạnh sản xuất phần mềm Công nghệ phần mềm không chỉ tập trung vào các quá trình công nghệ của phát triển phần mềm, mà nó còn bao gồm cả những hoạt động khác như quản lý dự án phần mềm, phát triển công cụ, phương pháp và các lý thuyết hỗ trợ phát triển phần mềm.

Một cách tổng quát, Công nghệ phần mềm áp dụng hướng tiếp cận có hệ thống và có tổ chức để đáp ứng quá trình phát triển phần mềm, đây là cách hiệu quả nhất để tạo ra phần mềm có chất lượng cao. Tuy nhiên, yếu tố công nghệ ở đây là tất cả sự chọn lựa phương pháp phù hợp nhất để giải quyết tốt các tình huống, vì vậy có thể áp dụng hướng tiếp cận sáng tạo hơn, ít gò bó hơn để phát triển hiệu quả hơn cho một số loại phần mềm. Cụ thể là một quá trình phần mềm linh hoạt hơn, có khả năng thay đổi nhanh chóng thì đặc biệt phù hợp cho sự phát triển những hệ thống phần mềm trên nền web tương tác, ứng dụng mobie, các loại này yêu cầu phải có sự kết hợp việc phát triển phần mềm với kỹ năng thiết kế đồ hoạ.

Công nghệ phần mềm quan trọng bởi 2 lý do sau

Ngày càng nhiều cá nhân và tổ chức dựa vào các hệ thống phần mềm tiên tiến nên chúng ta phải tạo ra các hệ thống tinh cậy đúng đắn một cách nhanh chóng và kinh tế.

Sử dụng các phương pháp, công nghệ để tạo ra phần mềm chuyên nghiệp với giá thành thấp và có thời gian sống lâu thì tốt hơn là tạo ra những dự án, chương trình mang tính cá nhân. Bởi vì những thất bại, sai sót trong việc sử dụng các phương pháp, công nghệ sẽ dẫn đến giá thành cao, chi phí kiểm thử, đảm bảo chất lượng, bảo trì hệ thống tăng cao.

Bốn hoạt động chủ yếu của quy trình phần mềm

- Đặc tả phần mềm, khách hàng cùng với nhà phát triển xác định tất cả những gì cần được tạo ra và những ràng buộc trong quá trình vận hành của hệ thống
- Phát triển phần mềm là thiết kế và lập trình cho phần mềm
- Xác thực phần mềm là kiểm tra lại để đảm bảo tất cả đúng với yêu cầu của khách hàng
- Tiến hoá phần mềm là sửa đổi phần mềm thoả mãn những yêu cầu thay đổi của khách hàng và thị trường.

Bauer năm 1969 định nghĩa **"Công nghệ phần mềm** (Software Engineering - SE) là việc **thiết lập và sử dụng các nguyên lý công nghệ đúng đắn** để xây dựng phần mềm một cách kinh tế, vừa tin cậy vừa làm việc hiệu quả trên các máy thực".

"Công nghệ phần mềm là bộ môn tích hợp cả **qui trình**, các **phương pháp**, các **công cụ** để phát triển phần mềm máy tính" - Pressman định nghĩa năm 1995.

Qui trình phát triển và quản lý nhằm xác định trình tự thực hiện các công việc phát triển phần mềm bao gồm: xác định các tài liệu, sản phẩm cần bàn giao và cách thức thực hiện. Ngoài ra cần xác định các mốc thời gian (millestones) và sản phẩm đưa ra (theo các chuẩn).

Phương pháp là cách làm cụ thể để xây dựng phần mềm. Mỗi giai đoạn đều có phương pháp riêng như:

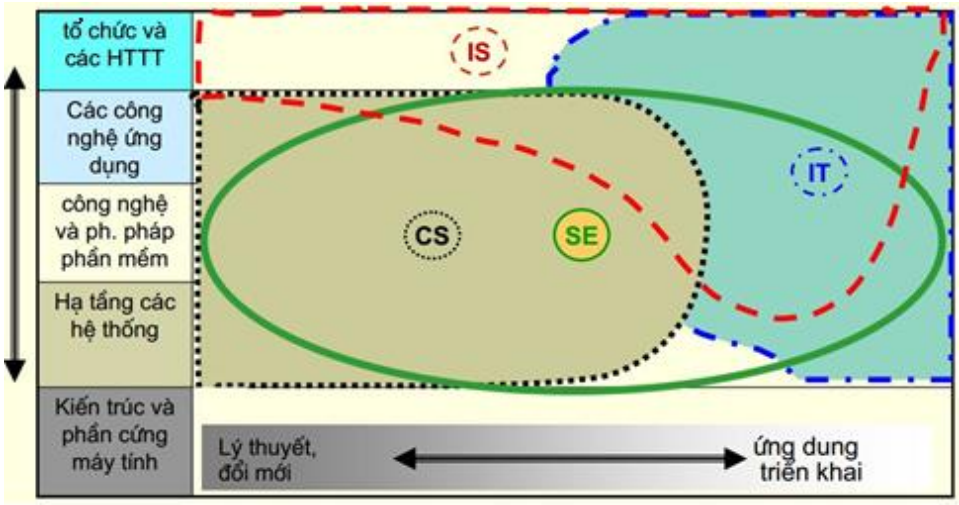
- ***Phân tích*** (xác định, đặc tả yêu cầu)
- ***Thiết kế*** (đặc tả kiến trúc, giao diện, dữ liệu, thủ tục).
- ***Lập trình*** (cấu trúc, hướng đối tượng)
- ***Kiểm thử*** (hộp đen, hộp trắng, hồi quy, luồn sợi)
- ***Quản lý dự án*** (PERT, GANTT, COCOMO)

Công cụ để trợ giúp một cách tự động hoặc bán tự động cho phương pháp. Ví dụ như: **C**omputer **A**ided **S**oftware **E**ngineering -> ***CASE*** các công cụ trợ giúp các

công đoạn khác nhau cho tiến trình phát triển phần mềm; các ngôn ngữ lập trình, công cụ sinh giao diện (Java, C#, C Builder, …); hỗ trợ phân tích, thiết kế (Modeler Oracle Designer, Rational Rose, UML…); trợ giúp lập trình: Compiler, debugger; trợ giúp quản lý: Project management

Theo từ điển **Larousse** (1996) định nghĩa chi tiết hơn: Công nghệ phần mềm là tập hợp các phương pháp, mô hình, kỹ thuật, công cụ và thủ tục liên quan đến các giai đoạn xây dựng một sản phẩm phần mềm. Các giai đoạn đó là: Đặc tả (specification), thiết kế (design), lập trình (programming), thử nghiệm (testing), sửa sai (debugging), cài đặt (setup) để đem vào ứng dụng (application), bảo trì (maintenance) và lập hồ sơ (documentation).

Công nghệ phần mềm có liên quan mật thiết đến những phần kiến thức về khoa học máy tính, hệ thống thông tin, công nghệ thông tin và các qui trình, công cụ, phương pháp,… Đó là lý do công nghệ phần mềm là ngành phức tạp và quan trọng. Hình sau thể hiện mối quan hệ giữa các chuyên ngành có liên quan đến công nghệ phần mềm. Trong đó, công nghệ phần mềm chiếm phần lớn ở trung tâm của các ngành có liên quan (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009).



Hình 3. Các lĩnh vực chuyên môn có liên quan

Theo hình trên, mỗi lĩnh vực tính toán được giới hạn bằng một vùng biên sau:

CS: Khoa học máy tính (-----)

IS: Hệ thống thông tin (- - - -)

IT: Công nghệ thông tin (_ . _ . _ . _)

SE: Công nghệ phần mềm (_____)

1.2.3 Phân loại phần mềm

Có 3 cách phân loại phần mềm: Hoàn thiện, chức năng và ứng dụng.

❖ ***Phân loại theo mức độ hoàn thiện***

Gồm 2 loại: Chương trình và sản phẩm phần mềm.

Chương trình do một người viết cho một người dùng. Mục đích của chương trình là để thu thập và xử lý số liệu, có thể chỉ được dùng một lần. Loại này không có tài liệu và không cần kiểm thử triệt để.

Sản phẩm phần mềm được nhiều người viết cho nhiều người dùng. Thông thường sản phẩm có độ phức tạp cao hơn chương trình và đòi hỏi mức độ đồng bộ và an toàn cao. Những người có kinh nghiệm viết chương trình nhỏ khó có thể tham gia tốt trong phát triển sản phẩm phần mềm lớn.

❖ ***Phân loại theo chức năng thực hiện***

Phần mềm hệ thống: Điều hành hoạt động máy tính, thiết bị và chương trình; trợ giúp các tiện ích (chương trình tổ chức file, nén, dọn đĩa, tool,…)

Phần mềm nghiệp vụ: Đáp ứng các công việc thực tế, các nghiệp vụ công việc, cá nhân. Số lượng phần mềm loại này rất lớn, đa dạng và phong phú. Có thể phân thành 2 loại theo cách làm và cách tiếp cận khác nhau: Sản phẩm đặt hàng và sản phẩm đại trà.

Sản phẩm theo đơn đặt hàng (Bespoke Product hoặc Customised Product): Được phát triển cho khách hàng đơn lẻ với yêu cầu, đặc thù riêng, giá thành thường rất cao. Ví dụ như những hệ thống phần mềm chuyên dụng, hỗ trợ nghiệp vụ cho một doanh nghiệp riêng lẻ …

Sản phẩm đại trà (Generic Product) được phát triển đáp ứng yêu cầu chung cho số lớn người dùng và bán rộng rãi, giá thành không cao như sản phẩm đặt hàng. Những sản phẩm phần mềm thuộc loại này thường là những phần mềm dành cho máy tính cá nhân. Ví dụ như phần mềm MS Office, photoshop, phần mềm diệt virus, tự điển,…

Phần mềm công cụ: Trợ giúp cho quá phát triển phần mềm. Ví dụ như các ngôn ngữ lập trình (soạn thảo, dịch, gỡ rối,…), công cụ trợ giúp một hoặc nhiều giai đoạn phát triển (phân tích, thiết kế, quản lý dự án, kiểm thử,… như: Developer2000, Powerdesigner, Microsoft Project Management, Eclipse…)

❖ ***Phân loại theo lĩnh vực ứng dụng***

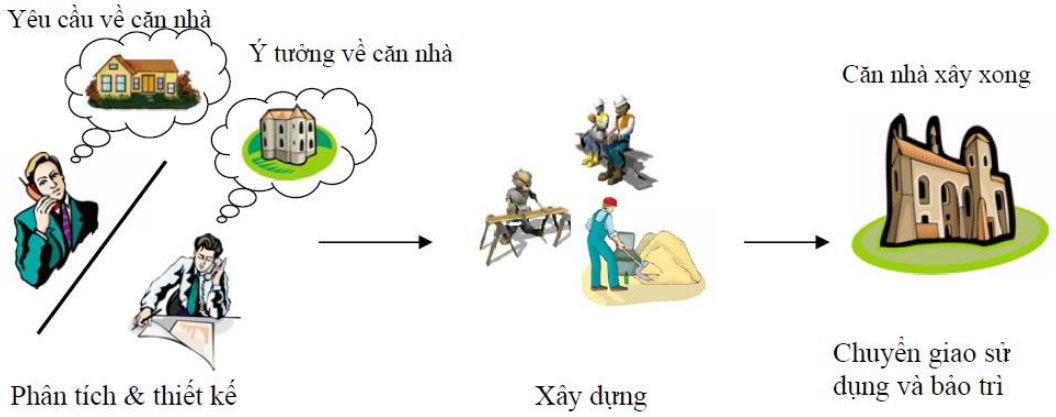
Phần mềm hệ thống (để tạo môi trường cho các chương trình khác chạy được, điều khiển mọi thứ trên máy tính).

- Phần mềm thời gian thực
- Phần mềm nghiệp vụ
- Phần mềm khoa học kỹ thuật
- Phần mềm nhúng tích hợp vào các thiết bị
- Phần mềm máy tính cá nhân
- Phần mềm trí tuệ nhân tạo
- Phần mềm dựa trên nền web

1.2.4 Chu trình phát triển phần mềm (System development life cycle – SDCL)

Chu trình phát triển phần mềm (SDLC) hay còn gọi là vòng đời phát triển hệ thống là một chuỗi các hoạt động, nó bắt đầu từ sự khởi đầu xây dựng cho đến kết thúc việc khai thác hệ thống. Những hoạt động này được thực hiện trong nhiều giai đoạn khác nhau như giai đoạn **khởi tạo, phân tích, thiết kế, xây dựng, thử nghiệm, cài đặt và bảo trì**.

Đối với qui trình phát triển phần mềm, chúng ta có thể hình dung rằng khi chúng ta muốn xây dựng một căn nhà. Công việc đầu tiên là dự tính xem chúng ta sẽ bỏ ra số tiền bao nhiêu để xây dựng căn nhà này. Dựa trên số tiền này chúng ta tìm kiếm và phát họa (có thể chỉ trong ý tưởng) căn nhà này như thế nào, kiểu gì, chiều rộng, chiều dài bao nhiêu, có mấy phòng, màu sắc, nền nhà như thế nào?,… Sau đó chúng ta chọn đơn vị xây dựng, trao đổi với đơn vị xây dựng tất cả các yêu cầu về ngôi nhà tương lai của mình nhằm thống nhất về giá cả cũng như các điều khoản về yêu cầu xây dựng (thời gian, chất lượng,…). Giai đoạn này được xem như giai đoạn phân tích. Tiếp đó, đơn vị xây dựng sẽ thực hiện công việc thiết kế chi tiết của căn nhà, và từng thành phần chi tiết trong căn nhà (phòng, tường, trần, mái, phòng ăn, phòng ngủ, điện, đèn,..). Giai đoạn này được xem là giai đoạn thiết kế. Sau đó, bộ phận thi công dựa vào các bản thiết kế chi tiết của căn nhà sẽ tiến hành việc xây dựng. Đây là giai đoạn xây dựng hay thực hiện. Căn nhà sau khi hoàn tất sẽ được chuyển giao để sử dụng. Trong quá trình sử dụng nếu có các hư hỏng thì đơn vị xây dựng sẽ phải tiến hành bảo trì và sửa chữa.



Hình 4. Hoạt động xây dựng căn nhà

Chu trình phát triển phần mềm:

Khởi tạo: Khảo sát hệ thống, vạch ra các vấn đề tồn tại trong hệ thống và các cơ hội của hệ thống. Đồng thời xác định phạm vi của hệ thống, lập kế hoạch các hoạt động của nhóm, xác định thời gian, nguồn lực cần thiết, chi phí đầu tư và lợi ích mang lại từ hệ thống. Kết quả của giai đoạn này là xác định được dự án hoặc được chấp nhận để phát triển, hoặc bị từ chối, hoặc phải định hướng lại.

Phân tích: Thu thập yêu cầu hệ thống, các phân tích viên làm việc với người sử dụng để xác định tất cả những gì mà người dùng mong muốn từ hệ thống đề xuất.

Nghiên cứu các yêu cầu và cấu trúc hóa (mô hình hóa) để dễ dàng nhận biết và loại bỏ những yếu tố dư thừa.

Phát sinh các phương án thiết kế chọn lựa phù hợp với yêu cầu và so sánh các phương án này để xác định giải pháp nào là đáp ứng tốt nhất các yêu cầu trong một mức độ cho phép về chi phí, nhân lực và kỹ thuật của tổ chức. Kết quả của giai đoạn này là bản mô tả về phương án được chọn.

Thiết kế: Kết quả của giai đoạn phân tích sẽ được chi tiết hóa để trở thành một giải pháp kỹ thuật để thực hiện. Các đối tượng và các lớp mới được xác định để bổ sung vào việc cài đặt yêu cầu và tạo ra một cơ sở kỹ thuật về kiến trúc. Ví dụ: các lớp giao diện, các lớp thuộc phạm vi vấn đề và hạ tầng cơ sở. Giai đoạn thiết kế sẽ đưa ra kết quả là bản đặc tả chi tiết cho gian đoạn xây dựng hệ thống. Về mức độ thiết kế có thể chia làm 2 mức: Thiết kế luận lý và thiết kế vật lý.

Thực hiện (code): Bao gồm việc triển khai các tài liệu thiết kế bằng ngôn ngữ lập trình để đưa ra các mô đun chức năng. Cuối giai đoạn này sẽ cho ra được mã nguồn của chương trình để làm đầu vào cho quá trình kiểm thử tiếp theo.

Kiểm thử: Là vận hành phần mềm để phát hiện lỗi trong phần mềm, lên kế hoạch kiểm tra kết hợp với các bộ tài liệu thiết kế và dữ liệu kiểm thử. Cuối giai đoạn này sẽ đưa ra được báo cáo về các lỗi của phần mềm trong khi kiểm nghiệm.

Cài đặt và bảo trì: Quá trình này tiến hành sau khi phần mềm được chuyển giao cho khách hàng. Mục tiêu của bảo trì là đảm bảo phần mềm vận hành ổn định, khắc phục lỗi trong quá trình vận hành một cách nhanh chóng. Việc nâng cấp phần mềm, thêm tính năng mới sẽ được tiến hành ở giai đoạn này nếu khách hàng yêu cầu.

Bảng 2. Các công cụ hỗ trợ cho các giai đoạn phát triển phần mềm

Giai đoạn	Công cụ	Diễn giải
Khởi tạo	Microsoft Project Management, Cocomo	Quản lý dự án, ước lượng
Phân tích & Thiết kế	Rational Rose, PowerDesigner, StarUML, Oracle, SQL Server 2008	Phân tích, thiết kế phần mềm, quản trị CSDL
code	Dev-C++, Elcipse SDK, NetBeans,	Lập trình C/C++, Java
	Dreamweaver CS6, VertrigoServ	Thiết kế Web, lập trình PHP
Kiểm thử	Quicktest Pro	Kiểm thử chức năng
	JUnit	Kiểm thử đơn vị
Bảo trì	Radare2 và ODA	Các công dùng cho dịch ngược thuộc loại (disassembler)
	Java Decompile và .NET Reflector	Các công cụ dịch ngược thuộc loại (decompiler)

1.2.5 Chất lượng phần mềm

Khi nói về chất lượng phần mềm chuyên nghiệp là xem xét phần mềm được sử dụng và thay đổi như thế nào. Do đó chất lượng phần mềm không chỉ là những gì phần mềm thực hiện được mà còn bao gồm cả hành vi của nó trong suốt quá trình thực hiện, cùng với cấu trúc, tổ chức của chương trình và cả những tài liệu có liên quan. Ví dụ như những thuộc tính của phần mềm về thời gian đáp ứng một yêu cầu truy vấn của người dùng hay sự khó hiểu về code của chương trình. Cụ thể như một game tương tác thì phải nhanh nhạy, hệ thống chuyển mạch điện thoại thì phải đáng tin cậy,…

Chất lượng phần mềm là sự làm đúng theo các yêu cầu **chức năng** và yêu cầu **hiệu năng**, các tiêu chuẩn phát triển, và các đặc điểm tiềm ẩn mà tất cả các phần mềm chuyên nghiệp cần có.

☙ ***Các yêu cầu về chức năng và hiệu năng***

Tính bảo trì được: Trên thực tế phần mềm luôn có yêu cầu sửa đổi. Để thực hiện tốt các yêu cầu này đòi hỏi phần mềm cần phải có kiến trúc tốt. Nghĩa là độ kết dính giữa các thành phần của phần mềm chặt nhưng ghép nối lỏng. Khi đó mã nguồn sẽ trở nên dễ đọc, dễ sửa chữa, dễ phát triển – đây là những hoạt động của bảo trì. Có như thế mức độ ảnh hưởng sau sửa đổi sẽ thấp, chỉ mang tính cục bộ, không làm ảnh hưởng đến toàn hệ thống. Các yếu tố về bảo trì tốt sẽ giúp phần mềm có tuổi thọ cao, chi phí bảo trì thấp và mang lại hiệu quả trong bảo trì.

Tính tin cậy: Tính tin cậy hay được xem là tính đúng đắn của phần mềm đối với người sử dụng. Phần mềm cần có ít khiếm khuyết, đáp ứng được nhu cầu người dùng, đảm bảo chức năng cần dùng, ổn định thời gian làm việc, không có lỗi lớn và cho kết quả xác đáng.

Tính hiệu quả: Có liên quan trực tiếp đến các thiết bị phần cứng máy tính. Phần mềm phải sử dụng tối ưu nguồn tài nguyên phần cứng. Phần mềm cần thực hiện việc tối ưu một cách hợp lý như có dùng ngôn ngữ bậc thấp để phục vụ việc tối ưu, truy cập trực tiếp đến thiết bị, sử dụng thuật toán viết gọn, sử dụng tối ưu CPU, RAM,…

Tính tái sử dụng: Các thành phần đã thực hiện có thể dùng lại trong các phần mềm cùng lớp (hoặc cùng lĩnh vực) với thời gian và công sức ít nhất có thể được.

Tính tiện dụng: Phần mềm dễ đọc, dễ sử dụng, giao diện trực quan, tự nhiên.

Tính tương thích: Phần mềm làm việc, tương tác tốt với các phần mềm khác, có thể nhập, xuất (Import/Export) dữ liệu dễ dàng.

☙ ***Tiêu chuẩn chất lượng phần mềm***

Các tiêu chuẩn chất lượng phần mềm phổ biến nhất được các tổ chức thế giới công nhận là CMM (Capability Maturity Model) và ISO.

CMM là kết quả của một nghiên cứu được không quân Mỹ tài trợ, nghiên cứu này được coi là một phương pháp đánh giá khách quan công việc của các nhà thầu phụ về phần mềm.

CMM (Nguyễn Công Danh & Trần Cao Đệ, 2013) được thiết kế để đánh giá quá trình phát triển phần mềm và chỉ dành riêng cho các tổ chức phát triển phần mềm. Tiêu chuẩn CMM bao gồm những thực tiễn tốt nhất (best practices) được tập hợp và rút kinh nghiệm từ rất nhiều tổ chức phát triển phần mềm khác nhau và chúng được tổ chức thành 5 mức độ trưởng thành khác nhau:

Mức 1 – Khởi tạo: Phụ thuộc cá nhân.

Mức 2 – Lặp lại: Khả năng quản lý.

Mức 3 – Định nghĩa: Xác lập các tiêu chuẩn quản lý.

Mức 4 – Quản lý: Có khả năng dự đoán, chi tiết hóa qui trình quản lý và tiêu chuẩn.

Mức 5 – Tối ưu: Có các hệ thống điều hành quản lý và chất lượng đạt hiệu quả.

ISO hướng đến nhiều loại tổ chức sản xuất và dịch vụ. Đối với phần mềm, ISO chỉ ra mức độ chất lượng yêu cầu tối thiểu mà một qui trình phần mềm phải đạt được (ISO certified) và việc cải tiến qui trình được thực hiện thông qua qui trình kiểm định.

Ngày nay, phần mềm không đứng riêng một mình mà thường là một bộ phận trong hệ thống hoàn chỉnh. Do đó, CMMI (Capability Maturity Model Integration) ra đời hướng đến các qui trình cho việc xây dựng cả hệ thống, bao gồm cả việc tích hợp để xây dựng và bảo trì toàn bộ hệ thống.

1.3 CÁC MÔ HÌNH PHÁT TRIỂN PHẦN MỀM

Mô hình (model) là một dạng thức trừu tượng về một hệ thống, được hình thành để hiểu hệ thống trước khi xây dựng hoặc thay đổi hệ thống đó. Mô hình là một dạng trình bày đơn giản hóa của thế giới thực. Bởi vì, hệ thống thực tế thì rất phức tạp và rộng lớn, khi tiếp cận hệ thống sẽ có những chi tiết, những mức độ phức tạp không cần thiết phải được mô tả và giải quyết. Mô hình cung cấp một phương tiện (các khái niệm) để quan niệm hóa vấn đề và giúp chúng ta có thể trao đổi các ý tưởng trong một hình thức cụ thể trực quan, không mơ hồ.

Hầu hết các kỹ thuật mô hình hóa sử dụng trong phân tích thiết kế là các ngôn ngữ đồ họa (đa số là sơ đồ - diagram), các ngôn ngữ này bao gồm một tập hợp các ký hiệu. Các ký hiệu này được dùng đi kèm các quy tắc của phương pháp luận giúp cho việc trao đổi các quan hệ thông tin phức tạp được rõ ràng hơn việc mô tả bằng văn bản.

Mục đích của mô hình hóa: Việc sử dụng các ký hiệu để trình bày hoặc mô hình hóa bài toán có các mục đích sau:

- Làm sáng tỏ vấn đề: Chúng ta có thể đưa ra được các lỗi hoặc các thiếu sót của hệ thống từ việc tiếp cận trực quan đồ họa hơn là các dạng trình bày khác như văn bản, đoạn mã,… Hơn nữa, việc mô hình hóa giúp chúng ta dễ dàng hiểu được hệ thống.

- Mô phỏng được hình ảnh tương tự của hệ thống: Hình thức trình bày của mô hình có thể đưa ra được một hình ảnh giả lập như hoạt động thực sự của hệ thống thực tế, điều này giúp cho người tiếp cận cảm thấy thuận tiện khi làm việc với mô hình (hình ảnh thu nhỏ của hệ thống thực tế).

- Gia tăng khả năng duy trì hệ thống: Các ký hiệu trực quan có thể cải tiến khả năng duy trì hệ thống. Thay đổi các vị trí được xác định trực quan và việc xác nhận trực quan trên mô hình các thay đổi đó sẽ giảm đi các lỗi. Do đó, chúng ta có thể tạo ra các thay đổi nhanh hơn và các lỗi được kiểm soát hoặc xảy ra ít hơn.

- Làm đơn giản hóa vấn đề: Mô hình hóa có thể biểu diễn hệ thống ở nhiều mức, từ mức tổng quát đến mức chi tiết, mức càng tổng quát thì ký hiệu sử dụng càng ít.

Có thể nói mô hình phát triển hay qui trình xây dựng phần mềm (Software Development/ Engineering Process - SEP) có tính chất quyết định để tạo ra sản phẩm chất lượng tốt với chi phí thấp và năng suất cao. Đồng thời, mô hình phát triển phần mềm giúp cho mọi thành viên trong dự án (từ người cũ đến người mới, trong hay ngoài công ty) đều có thể xử lý đồng bộ công việc tương ứng vị trí của mình thông qua cách thức chung của công ty. Điều này có ý nghĩa quan trọng đối với các công ty sản xuất hay gia công phần mềm trong nền công nghiệp phần mềm đầy cạnh tranh như ngày nay.

Tùy theo mô hình phát triển phần mềm, các nhóm công việc được triển khai theo những cách khác nhau. Để phát triển cùng một sản phẩm phần mềm người ta có thể dùng các mô hình khác nhau. Tuy nhiên không phải tất cả các mô hình đều thích hợp cho mọi ứng dụng. Mỗi mô hình sẽ phù hợp với một trình độ phát triển, quy mô sản phẩm và yêu cầu ràng buộc cụ thể về thời gian và tính chất của hệ thống thực.

Có thể chia thành 3 nhóm mô hình phát triển phần mềm khác nhau:

Nhóm mô hình tuyến tính:

- Mô hình xây dựng và hiệu chỉnh
- Mô hình thác nước- waterfall
- Mô hình phát triển nhanh ứng dụng - RAD - Rapid Application Development

Nhóm mô hình lặp:

- Mô hình bản mẫu - Prototyping - Evolutionary Development
- Mô hình xoắn ốc – Boehm's Spiral Model

Các mô hình khác:

- Mô hình chữ V
- Kỹ thuật thế hệ thứ 4 – 4GT
- Phát triển hướng đối tượng
- Phát triển hướng sử dụng lại
- Phát triển khung làm việc
- Phát triển phần mềm nguồn mở

1.3.1 Nhóm mô hình tuyến tính

☙ ***Mô hình xây dựng và hiệu chỉnh***

Mô hình xây dựng và hiệu chỉnh không có đặc tả hay thiết kế, chỉ đơn giản là làm đi làm lại cho đến khi nào đáp ứng được đầy đủ yêu cầu. Thường sử dụng trong các bài tập lập trình từ 100 đến 200 dòng mã lệnh.

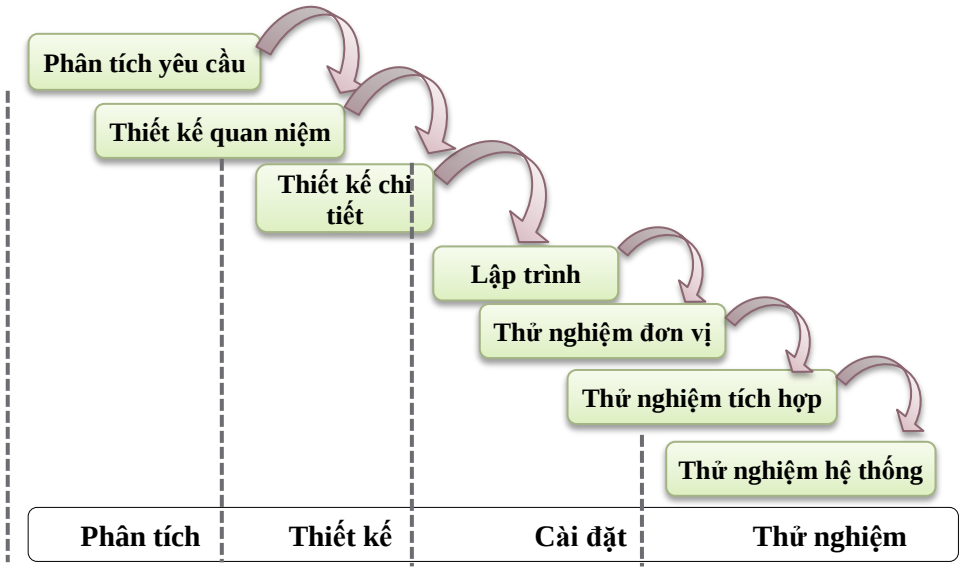
☙ ***Mô hình thác nước- waterfall***

Mô hình thác nước (Iam Sommerville, 2011) là một qui trình được đề xuất đầu tiên và đưa ra được các giai đoạn căn bản nhất tách biệt nhau, đầy đủ cho một quá trình phát triển hệ thống. Từ khi được đề xuất qui trình này nhanh chóng được phổ biến rộng rãi trong giới công nghiệp và cho đến bây giờ đã có nhiều cải tiến hoàn thiện.

Mô hình thác nước do Royce đề xuất năm 1970. Qui trình là các giai đoạn tuần tự nối tiếp nhau, có nghĩa là giai đoạn phân tích phải hoàn thành rồi đến giai đoạn thiết kế và cứ như thế với các giai đoạn tiếp theo, không cho phép sự quay lui. Các lỗi ở một số giai đoạn trước được phản hồi bởi các giai đoạn sau. Mỗi giai đoạn

chỉ hoàn thành khi có đầy đủ tài liệu và được nhóm đảm bảo chất lượng phần mềm (SQA) chấp nhận.

Mô hình thác nước có ưu điểm là đặc tả kỹ, qui định tốt về tài liệu cho mỗi giai đoạn, phân công chuyên trách và có tính kỷ luật cao, do đó rất thuận lợi trong việc bảo trì.



Hình 5. Các giai đoạn của mô hình thác nước

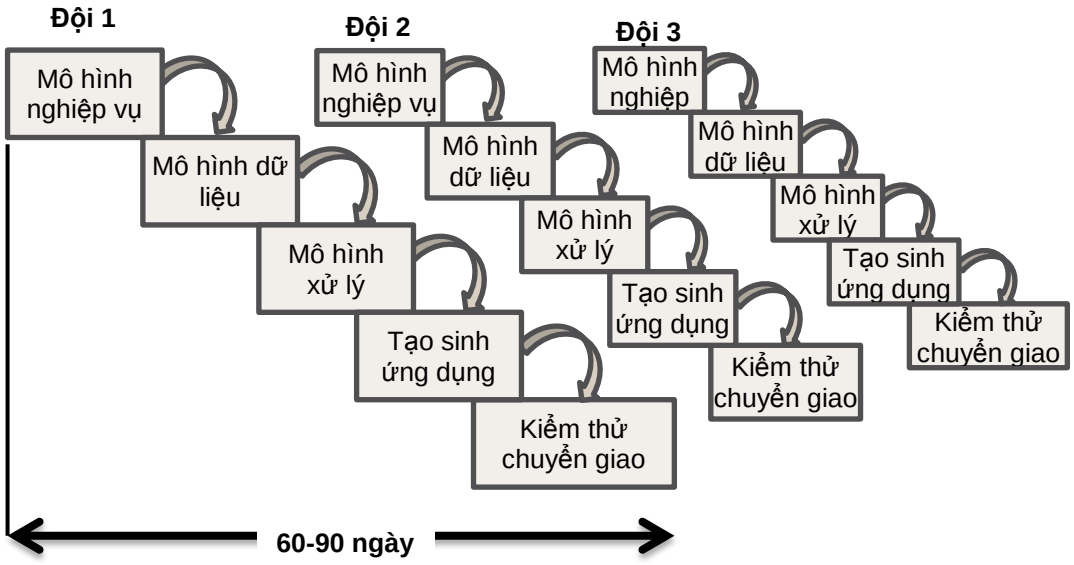
Nhược điểm của mô hình là quá nhiều kiểm thử, thẩm tra tài liệu và khó hiểu đối với khách hàng. Một nhược điểm khác của mô hình thác nước là chậm có phiên bản hoàn chỉnh, khách hàng phải kiên nhẫn chờ đợi sản phẩm phần mềm. Những sai sót được phát hiện muộn sẽ dễ trở thành thảm họa. Đồng thời nhà phát triển sẽ rất khó khăn trong việc thay đổi các pha đã được thực hiện. Giả sử, pha phân tích và xác định yêu cầu đã hoàn tất và chuyển sang pha kế tiếp, nhưng lại có sự thay đổi yêu cầu từ người sử dụng; thì chỉ còn cách là phải bắt đầu thực hiện lại từ giai đoạn xác định yêu cầu về sau.

Vì vậy, mô hình thác nước chỉ thích hợp khi các yêu cầu đã được tìm hiểu rõ ràng và những thay đổi phải có giới hạn trong suốt quá trình thiết kế. Tuy nhiên, trong thực tế có rất ít những hệ thống nghiệp vụ có các yêu cầu ổn định. Do đó dự án phát triển theo mô hình thác nước thường bị kéo dài thời gian (delay). Chính vì vậy mô hình thác nước chỉ có thể áp dụng phù hợp cho những dự án nhỏ, đơn giản.

☙ *Mô hình RAD*

Mô hình phát triển nhanh ứng dụng (RAD - Rapid Application Development) là mô hình phát triển phần mềm dạng tăng trưởng nhấn mạnh đến chu trình phát triển cực ngắn (60-90 ngày), nhanh chóng hoàn thành ứng dụng hệ thống đáp ứng yêu cầu người sử dụng. Để đạt được mục tiêu này, RAD dựa trên các thành phần (component-based) cùng với việc tái sử dụng (reuse) các thành phần thích hợp để xây dựng phần mềm tương đối đầy đủ các chức năng.

Dự án gồm một số nhóm (teams), mỗi nhóm làm một RAD theo các pha: Mô hình nghiệp vụ, mô hình dữ liệu, mô hình xử lý, tạo ứng dụng (dùng kỹ thuật thế hệ thứ 4 để **tự động sinh mã** hoặc tạo các thành phần có thể **tái sử dụng** lại sau này), kiểm thử và đánh giá.



Hình 6. Mô hình RAD

Hạn chế của RAD là cần nguồn nhân lực dồi dào để tạo các nhóm cho các chức năng chính. RAD không phải tốt cho mọi ứng dụng nhất là những ứng dụng không thể mô đun hóa hoặc đòi hỏi tính năng cao. Những dự án có tính mạo hiểm kỹ thuật cao thì không nên dùng RAD.

RAD thích hợp cho những hệ thống quản lý thông tin đơn giản, nhỏ gọn.

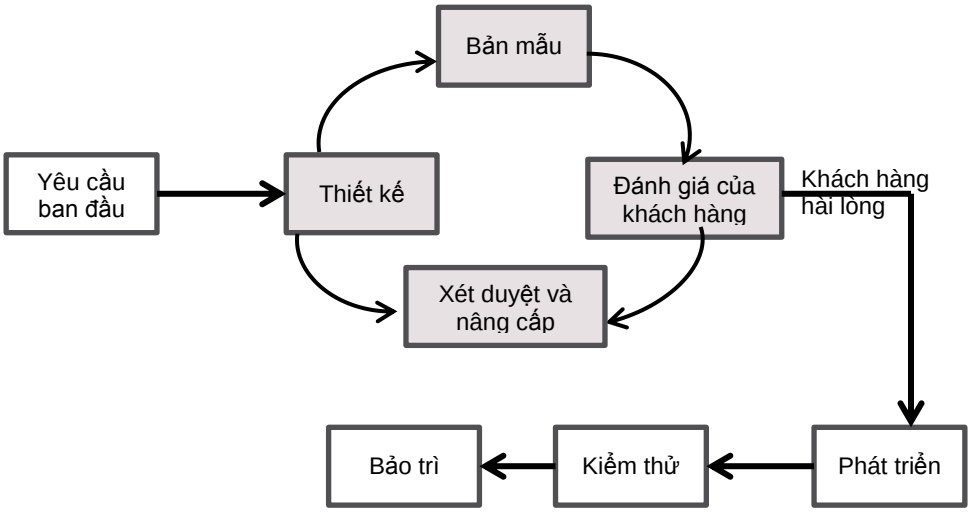
1.3.2 Nhóm mô hình lặp

- ***Mô hình bản mẫu (Prototype)***

Trong mô hình thác nước có rất nhiều lỗi có thể xảy ra nếu có sự thay đổi bất thường của hệ thống do khoảng cách sự hiểu biết của khách hàng và đội ngũ phát triển. Điều này dẫn đến việc phải chỉnh sửa lại một hoặc nhiều phần có khi là toàn bộ hệ thống, tốn kém về thời gian, chi phí. Do đó mô hình bản mẫu ra đời để khắc phục nhược điểm của mô hình thác nước.

Mô hình bản mẫu sẽ tạo một mẫu sản phẩm có chứa sự hạn chế về chức năng, khả năng của hệ thống đang phát triển. Sau đó sẽ giao cho khách hàng mẫu sản phẩm để dùng thử. Khi đó, khách hàng sẽ hình dung được các chức năng sẽ có trong hệ thống của mình và phản hồi những nhận định còn tồn tại ở mẫu thử này. Đội ngũ phát triển sẽ tiếp nhận thêm những yêu cầu mới của khách hàng để tinh chỉnh mẫu thử qua nhiều phiên bản cho đến khi thoả mãn yêu cầu của khách hàng thì dừng lại hoặc sang giai đoạn khác.

Thông thường khách hàng của mô hình bản mẫu chỉ xác định một tập các mục tiêu tổng quát, chung chung cho phần mềm, chưa nhận diện rõ các yêu cầu chi tiết đầu vào, xử lý ra sao, yêu cầu đầu ra như thế nào. Hoặc người phát triển không chắc về tính hiệu quả của thuật toán, thích nghi hệ điều hành hay giao diện người máy cần có. Khi đó mô hình bản mẫu được áp dụng là cách tiếp cận tốt nhất. Ví dụ như trang web thương mại điện tử, bán hàng trực tuyến là một loại có thể thực hiện theo mô hình bản mẫu.



Hình 7. Mô hình bản mẫu

Mô hình bản mẫu là mô hình hoạt động có chức năng tương đương với một tập hợp con của sản phẩm, được sử dụng khi cần thực hiện một cách nhanh chóng một sản phẩm phần mềm trên cơ sở phần mềm tương tự hay phần mềm mà nhóm thực hiện đã có nhiều kinh nghiệm.

Tuy nhiên, nhược điểm của mô hình bản mẫu là thiếu tầm nhìn của cả qui trình. Các hệ thống thường hướng cấu trúc nghèo nàn, đòi hỏi trình độ chuyên môn của đội ngũ phát triển phải cao để phát triển mẫu thử nghiệm. Mô hình bản mẫu không đặt ra mục tiêu tái sử dụng cho giai đoạn phát triển thực sự sau đó.

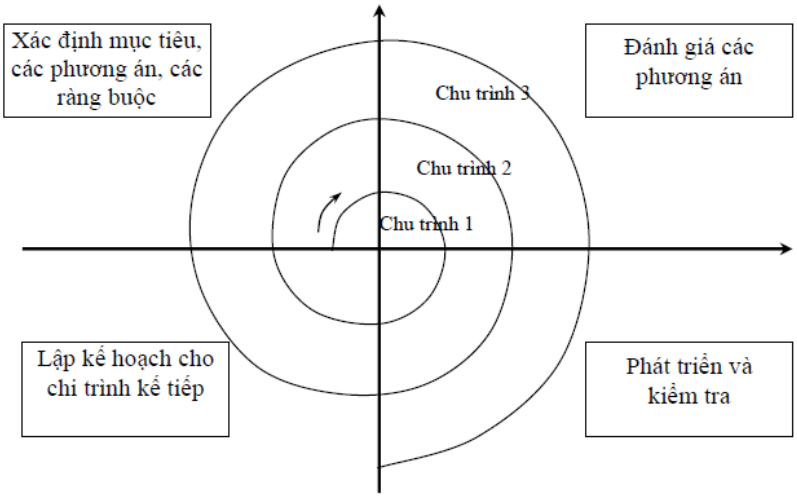
Mô hình bản mẫu chỉ nên áp dụng đối với những hệ thống cần được làm thật nhanh trong thời gian ngắn, hoặc có tương tác ở mức độ nhỏ hoặc vừa; có thể là một phần của những hệ thống lớn; hoặc những hệ thống có thời gian chu kỳ tồn tại ngắn.

☙ ***Mô hình xoắc ốc (Spiral model)***

Mô hình xoắn ốc được Boehm đưa ra. Nó có thể xem là sự kết hợp giữa mô hình thác nước và mô hình bản mẫu và đồng thời thêm một thành phần mới là chức năng **phân tích rủi ro** nhằm giảm thiểu rủi ro trong quá trình phát triển.

Một ví dụ của mô hình xoắn ốc là hệ điều hành Windows từ các phiên bản Windows 3.1 đến Windows 2003. Microsoft đã phát triển hệ điều hành phiên bản đầu tiên là Windows 3.1. Sau đó nhận được sự đánh giá phản hồi Windows 3.1 của thị trường rộng lớn, Microsoft đã lên kế hoạch để phát triển một phiên bản mới của

hệ điều hành đó là Windows 95 được cải tiến và tính linh hoạt về đồ họa. Tương tự các phiên bản Windows tiếp theo ra đời.



Hình 8. Mô hình xoắn ốc

Lập kế hoạch (planning): Xác định mục tiêu, tương tác và ràng buộc. Chu trình 1 - lập kế hoạch và thu thập yêu cầu ban đầu, từ chu trình 2 trở đi kế hoạch phải dựa trên các đánh giá của khách hàng.

Phân tích rủi ro (Risk analysis): Phân tích các lựa chọn và các chỉ định/giải quyết rủi ro.

Xây dựng sản phẩm (Engineering): Phát triển sản phẩm.

Khách hàng đánh giá (Customer evaluation): Đánh giá kết quả xây dựng.

Mô hình xoắn ốc hiện nay là mô hình hướng tiếp cận hiện thực nhất để phát triển các hệ thống lớn. Mô hình này cho phép nhà phát triển áp dụng mô hình bản mẫu tại mỗi chu trình phát triển bởi vì việc tích hợp mô hình bản mẫu vào như cơ chế loại trừ lỗi cho mô hình xoắn ốc. Nó kế thừa cách tiếp cận hệ thống từng bước từ chu kỳ sống cổ điển nhưng kết hợp với quá trình lặp lại phù hợp với thực tế.

Mô hình xoắn ốc xác định mục tiêu quan trọng luôn là chất lượng phần mềm đồng thời giảm nhẹ kiểm thử, nhanh chóng sửa chữa những lỗi xảy ra, bảo trì đơn giản. Mô hình này rất hữu ích đối với các phần mềm được phát hành thành nhiều phiên bản hoặc thường dành riêng cho các phần mềm nội bộ có kích thước lớn.

Mô hình xoắn ốc không phải là công cụ vạn năng, nhược điểm của mô hình là đòi hỏi phải có kỹ năng đánh giá lỗi, chi phí phân tích rủi ro và kích thước phần mềm ảnh hưởng rất lớn lên giá thành của phần mềm.

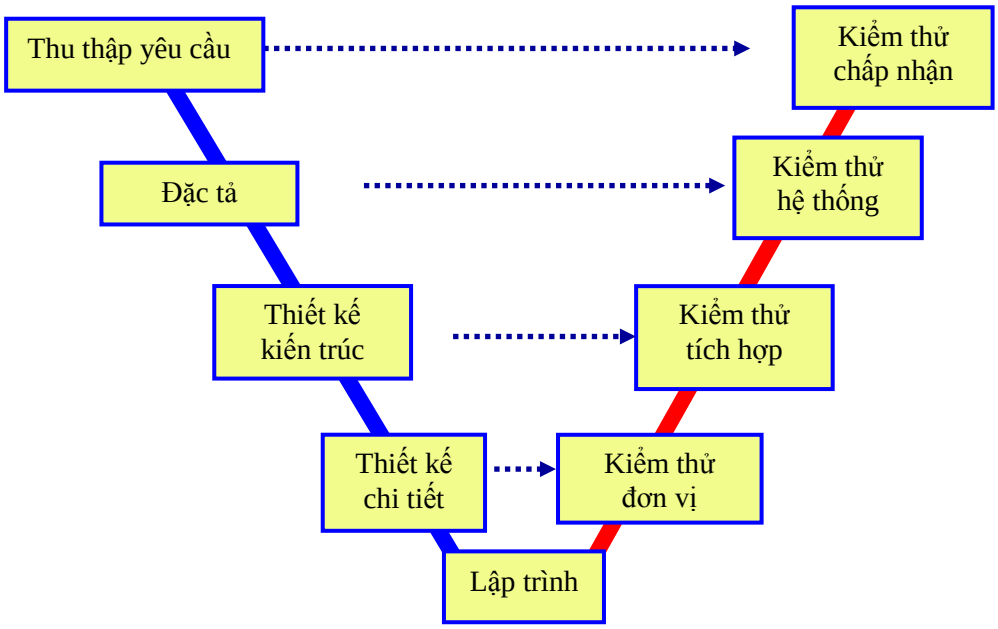
1.3.3 Mô hình khác

☙ *Mô hình chữ V*

Trong mô hình Waterfall, kiểm thử được thực hiện trong một giai đoạn riêng biệt. Còn với mô hình chữ V, toàn bộ qui trình được chia thành hai nhóm giai đoạn

tương ứng nhau: phát triển và kiểm thử. Mỗi giai đoạn phát triển sẽ kết hợp với một giai đoạn kiểm thử tương ứng như được minh họa trong hình 11.

Hướng đến mục tiêu đảm bảo chất lượng phần mềm nên tinh thần chủ đạo của V-model là các hoạt động kiểm thử phải được tiến hành song song (theo khả năng có thể) ngay từ đầu chu trình cùng với các hoạt động phát triển. Ví dụ, các hoạt động cho việc lập kế hoạch kiểm thử toàn hệ thống có thể được thực hiện song song với các hoạt động phân tích và thiết kế hệ thống.



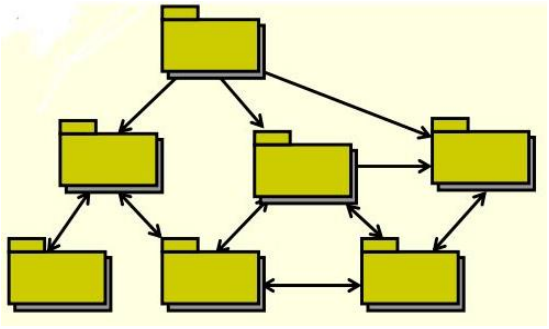
Hình 9. Mô hình chữ V

☙ ***Kỹ thuật thế hệ thứ 4 – 4GT***

Kỹ thuật này nhằm tạo ra những phần mềm nhỏ trong thời gian nhanh nhất và đơn giản nhất để phục vụ nhu cầu cấp bách. 4GT chỉ trợ giúp tự động sinh mã chương trình đối với từng chức năng cụ thể. Các công cụ ứng dụng điển hình như truy vấn CSDL (SQL), tạo báo cáo, bảng tính, sinh giao diện (C#),…Kỹ thuật này tiết kiệm được công sức cho phát triển phần mềm nhỏ. Mặc dù kỹ thuật thế hệ thứ 4 hướng chức năng, tự động sinh mã chương trình theo đặc tả nhưng vẫn xem việc phân tích, thiết kế là bước quan trọng. Kỹ thuật này không hiệu quả với phần mềm lớn, ứng dụng còn hạn chế, không phải mọi 4GT đều dễ dùng.

☙ ***Phát triển hướng đối tượng***

Phát triển hướng đối tượng (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009) tạo ra hệ thống được cấu thành từ các đối tượng. Đối tượng bao gói cả dữ liệu và xử lý nên các thành phần tương đối độc lập, dễ bảo trì, dễ dùng lại, việc mở rộng trở nên đơn giản hơn. Các thành phần được liên kết với nhau bằng truyền thông.



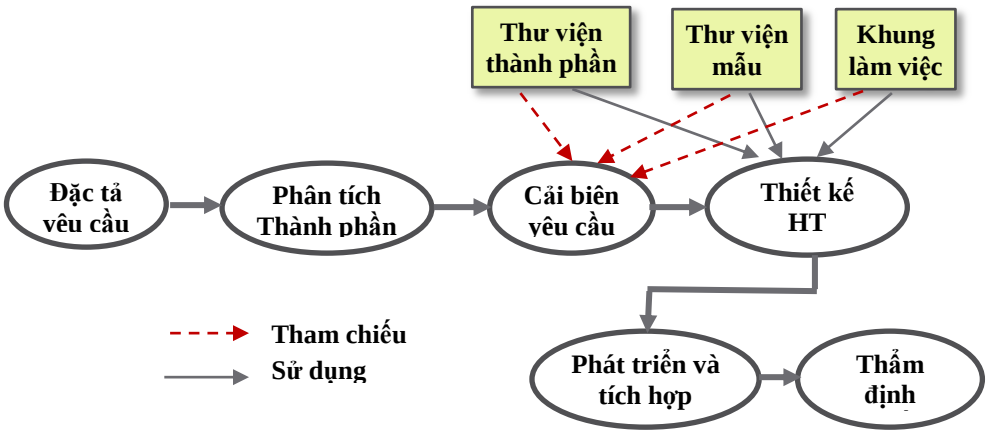
Hình 10. Mô hình cấu trúc hệ thống các đối tượng

☙ *Phát triển hướng sử dụng lại*

Phát triển hướng sử dụng lại dựa trên nền tảng của phát triển hệ thống hướng đối tượng. Do đó, người phát triển cần nhiều kinh nghiệm trong việc thực hiện bởi phần lớn quá trình phát triển đều sử dụng công cụ, tuy nhiên công cụ thì có nhiều hạn chế.

Phát triển hướng sử dụng phải trải qua các giai đoạn:

- Phân tích hệ thống thành các thành phần yêu cầu nhỏ - định hướng thành phần (component – mã nguồn).
- Cải biên các yêu cầu hướng thành phần, hướng mẫu (pattern – phân tích, thiết kế), hướng khung làm việc (framworks – lớp ứng dụng).
- Thiết kế hệ thống hướng tới tài sản sử dụng lại.
- Phát triển và tích hợp.



Hình 11. Tiến trình phát triển hướng sử dụng lại

☙ *Phát triển phần mềm nguồn mở*

Phần mềm mã nguồn mở là phần mềm mà mã nguồn của phần mềm cung cấp sẵn cho người dùng sửa đổi, sử dụng và tái phân phối lại. Mã nguồn của phần mềm là hoàn toàn được cung cấp cho người dùng sử dụng với các mục đích khác nhau như học tập, nghiên cứu, phân tích, sửa đổi, phân phối lại.

Lập trình viên, người đứng đầu một công ty hoặc là nhóm trưởng của một nhóm phát triển phần mềm muốn phát triển một phần mềm OSS vì họ chưa hài lòng

với phần mềm mà họ đang có hoặc chỉ đơn giản là vì các phần mềm mà họ cần chưa tồn tại. Linus Torvalds là một ví dụ, Linus cần một hệ điều hành Unix phù hợp và chạy tốt trên máy của mình, và kết quả là hệ điều hành Linux ra đời. Hay Eric Allman cần một máy chủ thư điện tử (Email Server) hiệu quả hơn dẫn đến phần mềm Sendmail ra đời, Larry Wall cần một công cụ để tự động tạo ra các Web và kết quả là ngôn ngữ lập trình Perl ra đời, Don Knuth cần một công cụ thuận tiện cho việc soạn thảo văn bản khoa học và kết quả là TEX ra đời. Những người phát triển phần mềm cho thấy sự quan tâm của họ trong việc tham gia phát triển phần mềm mã nguồn mở do nhu cầu của riêng họ đối với phần mềm đó và để thỏa mãn nhu cầu của mình họ góp phần vào việc cải tiến phần mềm đó.

Hướng phát triển này công khai thiết kế, công khai mã nguồn nên mọi người đều có thể tham gia phát triển và sử dụng. Phần mềm loại này giải quyết được nhiều vấn đề chung được nhiều người quan tâm như lợi ích, bản quyền (GNU, Linux)

Framework

Theo PCWorldVN, framework là những ngôn ngữ lập trình mới. Chúng là nơi có thể tìm thấy những ý tưởng, triết lý và ứng dụng thực tế của lập trình hiện đại. Ở đây những khó khăn về cú pháp và cấu trúc của ngôn ngữ lập trình hầu như không còn vì thế giới lập trình đã hội tụ vào vài tiêu chuẩn đơn giản hơn.

Hiện nay những hoạt động thú vị diễn ra trong lĩnh vực framework và được thảo luận sôi nổi về:

- **Laravel** - là một framework dùng để xây dựng web application, được thiết kế dựa trên mô hình MVC - Model, Controller, View.

- **Ember** - là 1 framework mã nguồn mở sử dụng cho phía client trong việc phát triển ứng dụng web.

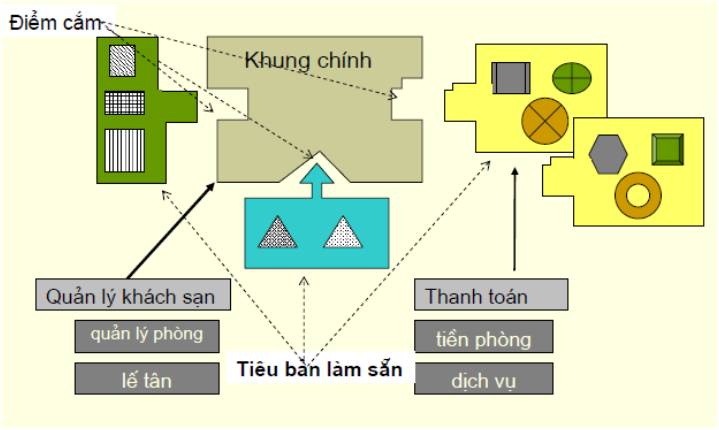
- **Node.js** - là một mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành Window, Linux... có đủ mọi tính năng.

Framework cho phép chúng ta tận dụng tối đa công sức khó nhọc của những người đi trước. Việc thừa hưởng toàn bộ điều tốt và điều xấu của một nền tảng mã lệnh dễ dàng hơn nhiều nếu thông qua framework. Chọn con đường khó khăn là tự mình viết lại toàn bộ bằng ngôn ngữ mới thay vì chọn một trong những framework phổ biến sẽ không cho phép bạn nhanh chóng thưởng thức tinh hoa của ngôn ngữ mới so với cách đơn giản hơn là để frameworkvà các API lo việc đó.

- Cách thức xây dựng framework
  - Xác định lớp bài toán cần giải quyết
  - Xây dựng khung bài toán (dựa trên patterns)
  - Làm sẵn các tiêu bản mẫu (dùng ngay được)

❖ Triển khai

- Phân tích bài toán theo khung
- Xác định tiêu bản tích hợp
- Lắp ghép hay tìm phần thay thế



Hình 12. Mô hình phát triển khung làm việc

Câu hỏi cuối chương

1. Hãy trình bày định nghĩa phần mềm máy tính. Phần mềm có vai trò quan trọng như thế nào trong xã hội? Hãy cho một ví dụ về sự cố phần mềm làm ảnh hưởng nghiêm trọng đến hoạt động của con người.

2. Theo số liệu thống kê của CHAOS, số lượng dự án phần mềm bị thất bại rất cao. Hãy chỉ ra những nguyên nhân có thể dẫn đến sự thất bại đó.

3. Phần mềm nghiệp vụ có đặc điểm gì? Hãy cho ví dụ về phần mềm nghiệp vụ. Vì sao giá thành của sản phẩm phần mềm đặt hàng đắt hơn rất nhiều lần so với sản phẩm phần mềm đại trà?

4. Hãy nêu và giải thích ngắn gọn về những đặc trưng của phần mềm.

5. Nêu định nghĩa sản phẩm phần mềm máy tính và các thành phần của sản phẩm phần mềm. Hãy chỉ ra sự khác biệt giữa phần mềm máy tính và sản phẩm phần mềm.

6. Hãy trình bày định nghĩa công nghệ phần mềm theo Pressman 1995. Hãy giải thích về qui trình, phương pháp và công cụ theo định nghĩa này.

7. Chất lượng phần mềm là gì? Trình bày các tiêu chuẩn chất lượng phần mềm phổ biến như CMM, ISO.

8. Hãy liệt kê các đặc trưng của phần mềm và giải thích vì sao "thay đổi" là một đặc trưng cơ bản của phần mềm? Hãy cho ví dụ về đặc trưng này.

9. Trình bày ngắn gọn 6 giai đoạn chính của chu trình phát triển phần mềm (phân tích khả thi, phân tích yêu cầu, thiết kế, mã hóa, kiểm thử, bảo trì).

10. Theo hình ảnh thể hiện mối quan hệ giữa các lĩnh vực tính toán (CS, IT, IS, SE). Công nghệ phần mềm (SE) có quan hệ như thế nào với các lĩnh vực còn lại? Vì sao SE là lĩnh vực phức tạp và quan trọng nhất trong các lĩnh vực tính toán này?

11. Hãy trình bày ngắn gọn đặc điểm của 3 mô hình tuần tự tuyến tính (mô hình xây dựng và hiệu chỉnh, mô hình thác nước và mô hình RAD). Vì sao 3 mô hình này thuộc nhóm mô hình tuần tự tuyến tính?

12. Mô hình sản xuất phần mềm nào chỉ nên được thực hiện đối với phần mềm nội bộ có kích thước lớn được thực hiện bởi chính nhân sự làm phần mềm của công ty? Vì sao? Trình bày ngắn gọn về mô hình đó.

13. Hãy trình bày đặc điểm, những ưu và khuyết điểm của mô hình bản mẫu. Vì sao mô hình bản mẫu (prototype) được xem là có thể khắc phục được nhược điểm của mô hình thác nước?

14. Trình bày đặc điểm của mô hình RAD. Mô hình RAD có ý nghĩa gì đối với các dự án nhỏ? Vì sao?

15. Nếu một hệ thống được phát triển mà khách hàng không chắc chắn về những gì họ mong muốn, các yêu cầu nghèo nàn, chưa được xác định rõ thì mô hình nào thích hợp để phát triển dự án kiểu này? Vì sao?

Bài tập thảo luận

AGILE, SCRUM VÀ QUÁ TRÌNH PHÁT TRIỂN PHẦN MỀM HƯỚNG ĐẾN KHÁCH HÀNG

Vấn đề của phát triển phần mềm kiểu cổ điển

Bạn đã bao giờ tham gia vào một dự án lớn về phát triển phần mềm kéo dài ròng rã nhiều tháng trời nhưng kết quả là khách hàng không sử dụng được sản phẩm? Chắc chắn hầu hết các lập trình viên đều đã hơn một lần đối mặt với vấn đề đó.

Trong các dự án, đặc biệt là các dự án phần mềm chúng ta sẽ gặp rất nhiều khó khăn trong việc thu thập đầy đủ và chính xác các yêu cầu của sản phẩm để lập kế hoạch tốt ngay từ đầu. Có quá nhiều vấn đề gây ảnh hưởng đến việc phát triển phần mềm. Trong khi đó có quá nhiều vấn đề mà chúng ta không lường trước được. Những vấn đề này có thể đến từ những yếu tố như kinh doanh, kỹ thuật, con người....

Trong giai đoạn những năm 1990 trên thế giới xảy ra cuộc khủng hoảng của quá trình phát triển phần mềm. Những phương pháp phát triển phần mềm theo cách truyền thống ngày càng bộc lộ nhiều nhược điểm và tỷ lệ các dự án thật bại cao. Từ đó các cá nhân và công ty riêng lẻ đã đưa ra các phương pháp phát triển phần mềm khác nhau để thích ứng với tình hình mới khiến cho phương pháp phát triển truyền thống đã không còn phù hợp

Những phương thức phát triển phần mềm này giúp phần nào giải quyết được một số vấn đề nhưng lại nảy sinh vấn đề khác về sự chia sẻ, cộng tác, kỹ thuật, công cụ, hướng phát triển .... Do vậy nhóm 17 người có uy tín trong nghề phát triển phần mềm thời điểm đó đã gặp nhau và đi đến thống nhất cho ra đời một bản tuyên ngôn Agile. Nhóm phát triển đã nói rằng: chúng tôi đã tìm ra các cách thức tối ưu hơn để phát triển phần mềm qua quá trình triển khai thực tế và giúp đỡ những người khác thực hiện điều đó.

Tuyên ngôn của Agile (Christopher Diggins. (k.n.))

- *Cá nhân và các tương tác quan trọng hơn các qui trình và công cụ.*
- *Tập trung làm cho phần mềm chạy được hơn là viết các tài liệu mô tả*
- *Cộng tác với khách hàng hơn là đàm phán hợp đồng*
- *Phản ứng với các thay đổi hơn là chỉ tuân theo một kế hoạch định sẵn*

1) Cá nhân và các tương tác quan trọng hơn các qui trình và công cụ

Ý tưởng là đặt trọng tâm vào con người và sự tương hỗ giữa những thành viên trong nhóm. Cơ bản là nếu dự án có những thành viên có năng lực, chịu làm việc cùng nhau thì sẽ mang đến thành công cho dự án. Nếu dự án của bạn có quy trình làm việc tốt, được hỗ trợ những công cụ tốt nhất nhưng những thành viên không "cùng nhìn về một hướng" thì khả năng dự án thất bại là rất lớn. Nói điều này không có nghĩa là phủ nhận tầm quan trọng của quy trình và công cụ nhưng trong Agile nó

được đặt sau yếu tố con người. Có một câu bằng tiếng Anh khá phổ biến nói về điều này là "a fool with a tool is just a fool"

2) Tập trung làm cho phần mềm chạy được hơn là viết các tài liệu mô tả

Trong một số quy trình phát triển phần mềm, việc tạo ra và cập nhật các tài liệu về sản phẩm là bắt buộc. Nhóm Dev không thể hoặc không đồng ý tiến hành công việc nếu không có tài liệu đặc tả về yêu cầu, thiết kế hệ thống. Nhóm Test thì yêu cầu tài liệu về sản phẩm để có thể viết trường hợp kiểm thử và kiểm thử được. Nhóm QA đòi tất cả các tài liệu phải được viết trước khi sản phẩm được giao cho khách hàng nếu không thì không đủ điều kiện, chuẩn để giao sản phẩm cho khách hàng. Thực ra đứng với góc độ khách hàng thì khách hàng chỉ quan tâm đến sản phẩm có hoạt động được và tốt hay không. Trong khi việc tạo và cập nhật tài liệu mất nhiều thời gian và được cho là buồn tẻ. Vậy tại sao mình phải tập trung quá nhiều cho việc không cần thiết mà không dành thời gian đó để trao đổi để hiểu thêm về công việc phải làm. Mọi người đừng hiểu lầm là làm Agile là không viết tài liệu. Ý tưởng là chỉ viết những gì mà mọi người cần đọc.

3) Cộng tác với khách hàng hơn là đàm phán hợp đồng

"Khách hàng là thượng đế" hay "khách hàng luôn luôn đúng". Tuy nhiên thì khách hàng có đủ loại khách hàng. Có khách hàng am hiểu về công nghệ, có người không. Có người suy nghĩ nhất quán có người thay đổi xoành xoạch, có người lạnh lùng có người cười nói suốt ngày, v.v và cách duy nhất để có thể làm việc tốt là phải cộng tác với khách hàng để hiểu được khách hàng muốn gì và cần gì để có thể tư vấn và điều chỉnh thay vì chỉ dựa vào những điều đã quy định trong hợp đồng.

4) Phản ứng với các thay đổi hơn là chỉ tuân theo một kế hoạch định sẵn

Có một điểm chung mà mình thấy trong hầu hết những dự án mình đã trải qua đó là không có dự án nào không có sự thay đổi điều chỉnh khi thực thi. Sự thay đổi đó có thể là thay đổi về yêu cầu, thay đổi công nghệ, thay đổi nhân sự, thay đổi deadline, thay đổi phương thức làm việc, v.v mặc dù kế hoạch đã được định ra rõ ràng từ đầu. Agile không khuyến khích cho sự thay đổi nhưng khuyến khích chúng ta tập thích nghi với thay đổi.

Có một điều thú vị là đa số trong chúng ta đều cơ bản đồng ý với 4 tuyên ngôn của Agile. Nhiều người hiểu tầm quan trọng của "cá nhân" hay "cá nhân là tài sản quý giá nhất công ty" nhưng sẵn sàng thay đổi nhân lực để tương thích với quy trình/công cụ hiện có. Nhiều người hiểu "khách hàng là thượng đế" và "phải thích nghi với sự thay đổi" nhưng sẵn sàng tuyên bố "Dẹp, không làm nữa" vì khách hàng thay đổi yêu cầu liên tục. Hay như "sản phẩm xài được là quan trọng " nhưng vẫn cố gắng viết thêm tài liệu với ý nghĩ rằng "biết đâu/lỡ sau này có ai cần thì có cái mà cung cấp".

4 tuyên ngôn của Agile nói dễ hơn làm nhưng khi bạn theo Agile thì bạn hãy chuẩn bị tinh thần để "làm" chứ không phải để "nói".

Phát triển sản phẩm hướng đến khách hàng

Agile không chỉ đơn thuần là một sính ngữ, nó còn thể hiện một số mục tiêu mong muốn và những nguyên lý trong việc phát triển các phần mềm thương mại.

Trong khi Scrum và các qui trình phát triển phần mềm khác theo mô hình Agile đã chứng minh được giá trị, tôi nghĩ rằng các tổ chức chấp nhận những phương pháp này vẫn cần phải đảm bảo các qui trình phát triển phần mềm họ đang áp dụng luôn luôn hướng đến khách hàng.

Hình 13. Phát triển phần mềm theo Agile

Những nguyên lý sau có thể giúp bất kỳ một nhóm phát triển phần mềm nào, bất kể phương pháp luận mà họ đang sử dụng, đảm bảo được một qui trình phát triển phần mềm hướng đến khách hàng:

Chuyển giao phần mềm cho khách hàng sớm và liên tục.

Thu thập các phản hồi từ khách hàng sau mỗi lần chuyển giao phần mềm và chia sẻ chúng với cả nhóm phát triển.

Để khách hàng tham gia vào quá trình review và đặt mức ưu tiên cho các chức năng

Đừng thực hiện bất kỳ thứ gì một khi khách hàng chưa đồng ý.

Không gì có thể thay thế cho một khách hàng thực sự./.

Agile là gì?

Phát triển phần mềm linh hoạt (agile software development – gọi tắt là Agile), Agile là một triết lí (philosophy) cho việc phát triển phần mềm. Nói cách khác, đó là một cách "tư duy" về các dự án phần mềm. Các triết lí của Agile được cụ thể hóa bởi một số phương pháp phát triển phần mềm (method), chẳng hạn như Extreme Programming (XP) hay Scrum, gọi tắt là các phương pháp Agile. Mỗi phương pháp Agile bao gồm một tập hợp các quy tắc (pratice), chẳng hạn quy tắc về sử dụng công cụ quản lí mã nguồn, quy tắc về các chuẩn lập trình hay quy tắc trình diễn sản phẩm hàng tuần cho khách hàng.

Agile thường sử dụng cách lập kế hoạch thích ứng (adaptive planning), việc phát triển và chuyển giao theo hướng tiến hóa; sử dụng các khung thời gian ngắn và

linh hoạt để dễ dàng phản hồi lại với các thay đổi trong quá trình phát triển. Ngày nay, triết lí Agile đã vượt xa khỏi khu vực truyền thống của mình là phát triển phần mềm để đóng góp sự thay đổi trong cách thức làm việc, quản lí, sản xuất ở các ngành khác như sản xuất (manufacturing), dịch vụ, sales, marketing, giáo dục v.v.

Thuật ngữ "Agile" chính thức được sử dụng rộng rãi theo cách thống nhất kể từ khi "Tuyên ngôn Agile" được giới thiệu ra công chúng năm 2001. Nhờ tính linh hoạt, đa dạng và hiệu quả cao, các phương pháp Agile ngày càng trở thành sự lựa chọn hàng đầu của các khách hàng, nhà phát triển, các công ty phát triển phần mềm.

Triết lí Agile được đưa ra trong một bản tuyên ngôn (manifesto) gồm **4 điểm và được làm rõ hơn bởi 12 quy tắc (**Christopher Diggins):

Tuyên ngôn 4 điểm của Agile là:

- ❖ Cá nhân và các tương tác quan trọng hơn qui trình và công cụ.
- ❖ Tập trung làm cho phần mềm chạy được thay vì viết tài liệu.
- ❖ Cộng tác trực tiếp với khách hàng thay vì dựa trên hợp đồng.
- ❖ Phản ứng với các thay đổi thay vì tuân theo một kế hoạch định sẵn.

Bản tuyên ngôn được cụ thể hóa bằng 12 nguyên tắc sau:

1. Ưu tiên cao nhất của dự án là thỏa mãn khách hàng bằng việc bàn giao sản phẩm sớm và liên tục.
2. Hoan nghênh các thay đổi từ phía khách hàng, kể cả các thay đổi vào giai đoạn cuối.
3. Bàn giao sản phẩm theo chu kì từ vài tuần đến vài tháng. Chu kì ngắn tốt hơn chu kì dài.
4. Các nhân viên hiểu nghiệp vụ và các lập trình viên phải làm việc cùng nhau hàng ngày.
5. Tổ chức dự án xoay quanh những cá nhân tích cực. Hỗ trợ và tin tưởng họ.
6. Phương pháp giao tiếp tốt nhất trong đội dự án là gặp mặt trực tiếp.
7. Các chức năng đã hoạt động là thước đo chính cho tiến độ dự án.
8. Khuyến khích phát triển bền vững: Lập trình viên, người dùng, nhà quản lí…phải có khả năng tham gia dự án một cách liên tục.
9. Liên tục cải tiến chất lượng thiết kế và mã nguồn.
10. Tính đơn giản giữ vai trò cốt yếu. Làm càng ít càng tốt.
11. Những yêu cầu và thiết kế tốt nhất được nảy nở từ những nhóm làm việc tự chủ.
12. Sau những khoảng thời gian nhất định, đội dự án xem xét cách thức cải tiến hiệu quả công việc.

Vì sao chúng ta cần Agile?

Theo quan niệm truyền thống, một dự án phần mềm được coi là thành công khi sản phẩm được giao đúng hạn, trong ngân sách cho phép và làm đúng yêu cầu

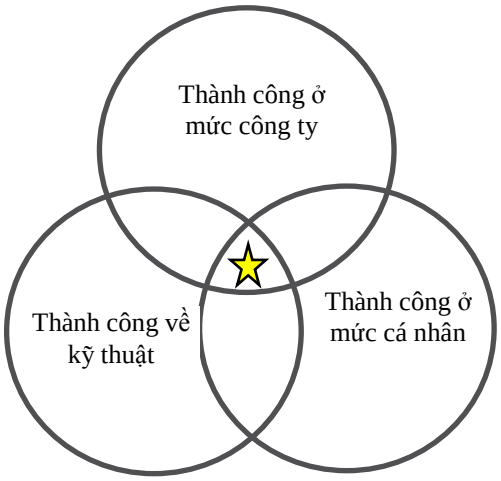
của khách hàng. Trên thực tế, nhiều dự án thỏa mãn tất cả các tiêu chí này nhưng rút cuộc vẫn bị coi là thất bại bởi phần mềm làm ra không được người dùng ưa thích, hoặc không mang lại nhiều lợi ích cho các cá nhân, tổ chức sử dụng chúng. Ngoài các yếu tố truyền thống nói trên, một dự án phần mềm chỉ được coi là thành công khi thỏa mãn ba tiêu chí: Thành công ở mức cá nhân, thành công về mặt kĩ thuật và thành công ở mức công ty. Thành công ở mức cá nhân giúp kích thích các thành viên trong nhóm. Thành công về kĩ thuật đảm bảo khả năng bảo trì và tiến hóa của sản phẩm. Vị trí của nhóm sẽ được bảo đảm nhờ các thành công mà nhóm mang lại cho công ty. Các phương pháp Agile giúp cho dự án phần mềm đạt được ba thành công này

❖ **Thành công ở mức công ty**

Agile nhắm đến việc tạo ra giá trị cho công ty trong khi làm giảm chi phí. Đồng thời, Agile giúp sớm xác định các kì vọng đối với sản phẩm đang được phát triển. Nhờ đó, những dự án không mang lại giá trị như mong đợi sẽ được phát hiện sớm, tiết kiệm chi phí cho công ty.

Theo phương pháp Agile, các chuyên gia về nghiệp vụ (business) sẽ làm việc trực tiếp cùng với đội dự án. Các chức năng quan trọng nhất của sản phẩm được tập trung phát triển trước và được đưa vào vận hành sớm nhất có thể. Các phiên bản mới với các tính năng mới sẽ lần lượt được đưa thêm vào.

Agile giúp tăng cường khả năng giao tiếp giữa các thành viên trong nhóm. Chất lượng mã nguồn được cải tiến liên tục. Tiến độ dự án cũng được xem xét và đánh giá một cách thường xuyên.



Hình 14. Ba tiêu chí đánh giá thành công của một dự án phần mềm

❖ **Thành công về mặt kĩ thuật**

Trong Extreme Programming, một phương pháp tuân theo triết lí Agile, các lập trình viên làm việc cùng nhau. Nhờ vậy, các chi tiết quan trọng sẽ không bị bỏ sót, mỗi đoạn code sẽ được kiểm tra bởi ít nhất hai người. Các lập trình viên liên tục tích hợp những đoạn code vừa viết vào hệ thống, cho phép một phiên bản mới của phần mềm được "ra lò" bất cứ khi nào nó góp thêm một giá trị đáng kể. Hơn nữa,

toàn bộ đội dự án tập trung hoàn thành một chức năng trước khi chuyển sang chức năng tiếp theo. Bởi vậy, tiến độ công việc được kiểm soát tốt hơn và dự án có thể dễ dàng "chuyển hướng" khi có những thay đổi từ phía khách hàng.

Ngoài ra, Extreme Programming cũng đề xuất những quy tắc giúp tạo ra các thiết kế và các đoạn mã tốt. Chẳng hạn, quy tắc "phát triển dựa trên kiểm thử" (test-driven development) trợ giúp lập trình viên viết các chương trình thực hiện đúng chức năng mong muốn.

❖ **Thành công về mặt cá nhân**

Mỗi thành viên trong dự án Agile, dù ở bất kì cương vị nào, cũng đều cảm nhận được một cách rõ ràng sự thành công của bản thân.

Các lập trình viên nhận thấy trình độ kĩ thuật cũng như tầm ảnh hưởng của mình đối với dự án được nâng cao, chẳng hạn trong việc ước lượng và lập kế hoạch. Quyền tự chủ của đội dự án cũng được tăng cường.

Các tester nhận thấy họ có ảnh hưởng lớn đến chất lượng sản phẩm, đồng thời giảm được các công việc lặp lại một cách nhàm chán.

Nhà quản lí dự án hài lòng vì kiểm soát được tiến độ công việc, dự án thực hiện đúng các cam kết và làm thỏa mãn khách hàng.

Khách hàng, người sử dụng, các chuyên gia nghiệp vụ cảm thấy hài lòng vì điều kiển được hướng đi của dự án và các ý kiến được lắng nghe.

Các nhà lãnh đạo cao cấp sẽ cảm thấy hài lòng vì dự án mang lại lợi nhuận lớn cho công ty

Hình 15. Agile và Scrum

Phương pháp Scrum

Scrum là một thành viên của họ Agile. Scrum được xây dựng dựa trên lý thuyết quản lý tiến trình thực nghiệm (empirical process control), hay còn gọi là thực nghiệm luận (empiricism). Lý thuyết này chỉ ra rằng tri thức đến từ kinh nghiệm và việc ra quyết định được dựa trên những gì đã biết. Điều này sẽ giúp giảm thiểu rủi ro và tăng tính chính xác đặc biệt là trong môi trường phát triển phần mềm nhiều biến động.

Ví dụ đơn giản nhất cho khái niệm Scrum đó là những đàn chim di cư. Chúng không hề có kế hoạch chi tiết cho hành trình của mình. Nhưng vẫn vượt qua được hàng chục nghìn km mỗi năm qua những vùng đất xa lạ nhờ việc quan sát và thích nghi liên tục với điều kiện khí hậu thức ăn. Nơi trú ngụ của từng vùng ....

Ba yếu tố nòng cốt tạo thành một mô hình quản lý tiến trình thực nghiệm gồm: sự minh bạch (transparency), thanh tra (inspection) và thích nghi (adaptation).

1) Minh bạch

Các khía cạnh quan trọng của tiến trình phải được hiển thị rõ ràng cho những người có trách nhiệm với thành quả của tiến trình đó. Sự minh bạch yêu cầu các yếu tố này cần được định nghĩa theo một tiêu chuẩn để những người quan sát có thể hiểu những gì họ thấy theo cùng một cách.

2) Thanh tra

Người sử dụng Scrum phải thường xuyên thanh tra các tạo tác và tiến độ đến đích để phát hiện các bất thường không theo ý muốn. Tần suất thanh tra không nên quá dày để khỏi ảnh hưởng đến công việc. Công tác thanh tra có ích nhất khi được thực hiện bởi người có kĩ năng tại các điểm quan trọng của công việc.

3) Thích nghi

Nếu một người thanh tra xác định được rằng có vấn đề nào đó vượt quá giới hạn cho phép, và hậu quả của vấn đề đó đối với sản phẩm là không thể chấp nhận được, thì quy trình hoặc các vật liệu được xử lý (processed material) phải được điều chỉnh. Sự điều chỉnh phải được tiến hành càng sớm càng tốt để giảm thiểu các sai sót khác có thể xảy ra.

Lợi ích của srum mang lại

1) Cải thiện chất lượng phần mềm

Framewrok của Scrum giúp nhóm phát triển Scrum nhận phản hồi liên tục và nhanh chóng điều chỉnh để đảm bảo chất lượng phần mềm cao nhất, đồng thời đáp ứng đúng nhu cầu của thị trường luôn thay đổi. Bằng cách áp dụng các nguyên tắc nghiệm ngặt trong mô hình Scrum, nhóm phát triển Scrum có thể đưa ra thị trường các sản phẩm có chất lượng tốt nhất.

2) Rút ngắn thời gian phát hành phần mềm

Scrum đã được chứng minh là cung cấp sản phẩm đến tay khách hàng cuối cùng nhanh hơn 30%-40% so với phương pháp truyền thống. Vì mô hình Scrum làm việc với nguyên tắc chính là chia nhỏ phần mềm cần sản xuất ra thành các phần nhỏ để phát triển gọi là Sprint. Mỗi Sprint thường mất 2- 4 tuần để hoàn thành.

3) Nâng cao tinh thần đồng đội

Mô hình Scrum áp dụng cách thức tự quản và tự tổ chức (self-managing & self-organizing), với mục đích các thành viên trong nhóm Scrum có thể vui vẻ làm

việc cùng nhau, khơi dậy sự sáng tạo, chủ động trong họ. Cách thức tự quản lí cũng cho phép mọi thành viên trong nhóm Scrum đều có thể ra quyết định. Trong nhóm Scrum sẽ không có nhóm trưởng mà chỉ có Scrum Master, là người giúp nhóm vượt qua các trở ngại và che chắn cho nhóm khỏi những ảnh hưởng từ nội bộ hay bên ngoài.

4) Gia tăng tỷsuất hoàn vốn đầu tư(ROI)

Giảm thời gian sản xuất là lí do chính yếu nhất giúp các dự án Scrum đạt được ROI cao hơn. Bởi vì doanh thu và các mục tiêu khác đến sớm hơn, nên tổng lợi nhuận cao hơn theo thời gian. Đây là một nguyên lý cơ bản của giá trị hiện tại thuần (NPV)

5) Tăng mức độhài lòng của khách hành

Nhóm Scrum cam kết sản xuất ra các sản phẩm hoặc dịch vụ có thể khiến khách hàng hài lòng. Sở dĩ như vậy vì nhóm Scrum xem khách hàng là đối tác và giữ khách hàng tham gia vào dự án; thành phần tham gia dự án Scrum còn có Product Owner là người hiểu rõ các yêu cầu (requirements) của dự án và nhu cầu của khách hàng; thời gian cung cấp sản phẩm nhanh hơn

6) Kiểm soát dựán tốt

Tất cả các thành viên của nhóm dự án Scrum, Product Ower, Scrum Master và các bên liên quan có rất nhiều cơ hội để kiểm tra và điều chỉnh sản phẩm trong suốt dự án và cuối cùng tạo ra sản phẩm tốt nhất. Vì các framework của mô hình Scrum cho phép nhận các phản hồi liên tục và qua đó có thể nhanh chóng điều chỉnh.

7) Giảm thiểu rủ ro

Mô hình Scrum giúp giảm thiểu rủi ro thất bại hoàn toàn khi mất một số tiền đầu tư khổng lồ và thời gian dài để triển khai dự án mà không thu lại được ROI. Vì như đã trình bày, Scrum làm việc theo từng giai đoạn, từng Sprint, nên nhóm dự án có thể thực hiện từng bước, sau đó rút kinh nghiệm hoặc tiếp tục phát huy các ưu điểm của Sprint trước để cải thiện hơn sản phẩm trong Sprint sau tránh gây thất thoát quá lớn trong suốt dự án.

Scrum *(danh từ): Một khung làm việc trong đó người ta có thể phát hiện những vấn đề tồn đọng trong khi vẫn đem lại những giá trị lớn nhất một cách hiệu quả và sáng tạo. Scrum mang các đặc tính:*

• Tinh gọn,

• Đơn giản dễ hiểu,

• Cực kì khó để thành thạo.

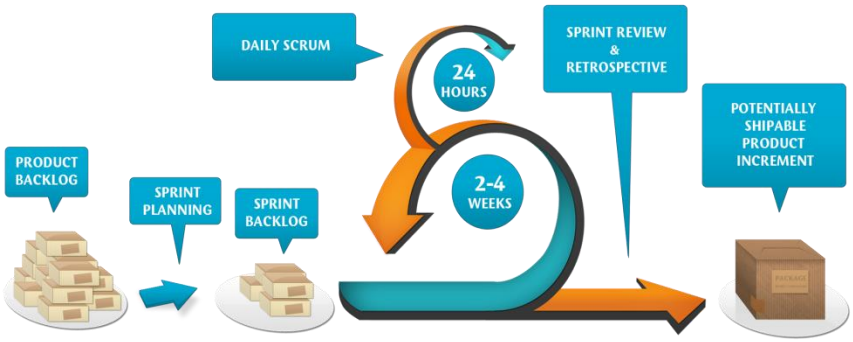
Scrum dựa trên một vòng lặp tăng tiến được gọi là Sprint. Mỗi Sprint bắt đầu với một một cuộc họp Kế Hoạch Sprint và kết thúc với việc cho ra một sản phẩm hoàn thiện. Đặc trưng của Scrum là độ phản hồi thông tin cao và sự minh bạch, cả

bên trong nhóm và bên ngoài nhóm làm việc. Chu kỳ ngắn và cộng tác tự nhiên làm nó lý tưởng cho các dự án có tính thay đổi nhanh chóng và/hoặc có khả năng đổi yêu cầu rất khẩn cấp.

Một trong những điểm cốt lõi của qui trình Scrum là đòi hỏi cả đội dự án phải chuyển giao phần mềm trong chuỗi các thời hạn liên tiếp gọi là sprint hoặc iteration. Đội Scrum bao gồm một product owner, một scrum master và các thành viên của nhóm phát triển. Product owner được xem là tiếng nói của khách hàng, và hoạt động như một đại diện cho khách hàng trong các cuộc họp lập kế hoạch.

Trong phương pháp Scrum, điểm duy nhất mà khách hàng có thể tham gia là quá trình review các sprint, nhưng điều này là không bắt buộc. Lý thuyết là có một product owner (thông thường là những người thiết kế sản phẩm, quản lý sản phẩm, hoặc phát triển phần mềm) hoạt động như một đại diện cho khách hàng là đủ. Vấn đề là họ không phải là khách hàng thực sự. Họ có định kiến và những phương thức riêng nhưng không hẳn đã tương đồng với nhu cầu và mong muốn của khách hàng.

Scrum được xây dựng trên năm giá trị cốt lõi, ba vai trò khác nhau, ba tạo tác, và ba (hoặc bốn) cuộc họp.



Hình 16. Mô hình scrum

❖ **Vai trò**

Trong Scrum, đội ngũ tham gia phát triển phần mềm được phân chia ra ba vai trò với trách nhiệm rõ ràng để đảm bảo tối ưu hóa các công việc đặc thù. Ba vai trò này bao gồm: **Product Owner**(chủ sản phẩm), Scrum Master và Development Team (Đội sản xuất hay Nhóm Phát triển).

Product Owner (chủ sản phẩm)

Là người chịu trách nhiệm về sự thành công của dự án, người định nghĩa các yêu cầu và đánh giá cuối cùng đầu ra của các nhà phát triển phần mềm.

Scrum Master

Là người có hiểu biết sâu sắc về Scrum và đảm bảo nhóm có thể làm việc hiệu quả với Scrum.

Development Team (Đội sản xuất, hay Nhóm phát triển)

Một nhóm *liên chức năng* (cross-functional) *tự quản lý* để tiến hành chuyển đổi các yêu cầu được tổ chức trong Product Backlog thành chức năng của hệ thống.

❖ **Cuộc họp (4 Events)**

Scrum định nghĩa quy tắc cho bốn sự kiện chủ chốt (các cuộc họp) nhằm tạo môi trường và quy cách hoạt động và cộng tác cho các thành viên trong dự án. Các lễ nghi này diễn ra trước khi Sprint bắt đầu (Sprint Planning), trong khi Sprint diễn ra (Daily Scrum) và sau khi Sprint kết thúc (Sprint Review và Sprint Retrospective).

Sprint Planning (Họp Kế hoạch Sprint)

Nhóm phát triển gặp gỡ với Product Owner để lên kế hoạch làm việc cho một Sprint. Công việc lập kế hoạch bao gồm việc chọn lựa các yêu cầu cần phải phát triển, phân tích và nhận biết các công việc phải làm kèm theo các ước lượng thời gian cần thiết để hoàn tất các tác vụ. Scrum sử dụng cách thức lập kế hoạch từng phần và tăng dần theo thời gian, theo đó, việc lập kế hoạch không diễn ra duy nhất một lần trong vòng đời của dự án mà được lặp đi lặp lại, có sự thích nghi với các tình hình thực tiễn trong tiến trình đi đến sản phẩm.

Daily Scrum (Họp Scrum hằng ngày)

Scrum Master tổ chức cho Đội sản xuất họp hằng ngày trong khoảng 15 phút để Nhóm Phát triển chia sẻ tiến độ công việc cũng như chia sẻ các khó khăn gặp phải trong quá trình phát triển phần mềm suốt một Sprint.

Sprint Review (Họp Sơ kết Sprint)

Cuối Sprint, nhóm phát triển cùng với Product Owner sẽ rà soát lại các công việc đã hoàn tất (DONE) trong Sprint vừa qua và đề xuất các chỉnh sửa hoặc thay đổi cần thiết cho sản phẩm.

Sprint Retrospective (Họp Cải tiến Sprint)

Dưới sự trợ giúp của Scrum Master, nhóm phát triển sẽ rà soát lại toàn diện Sprint vừa kết thúc và tìm cách cải tiến qui trình làm việc cũng như bản thân sản phẩm.

Câu hỏi thảo luận

1. Điều đầu tiên trong tuyên ngôn Agile là gì? Vì sao việc thỏa mãn khách hàng được quan tâm hàng đầu trong các nguyên tắc của Agile?

2. Có ý kiến cho rằng Agile là đơn vị kinh doanh nhưng lại tuân thủ nghiêm ngặt qui trình phát triển phần mềm, công nghệ là vấn đề trọng tâm của họ. Vậy ý kiến của bạn về vấn đề này như thế nào? đồng ý hay không? Vì sao?

3. Theo Scrum, product owner là gì? Có vai trò như thế nào trong phát triển phần mềm?

4. Agile là gì? Scrum là gì? Hai thành phần này có mối quan hệ như thế nào với nhau?

5. Theo Agile, một sản phẩm được xem là thành công thì sản phẩm đó phải đạt được các yêu cầu gì?

6. Theo Agile, các lập trình viên phải làm việc theo cách thức như thế nào?

7. Agile thể hiện tinh thần như thế nào đối với "sự thay đổi" từ phía khách hàng (đây là một đặc trưng của phần mềm)?

8. Trong mô hình Scrum, hãy chỉ ra 3 vai trò và 4 sự kiện.

9. Thời gian dành cho sprint daily mỗi này là bao nhiêu? Thời gian này dùng để làm gì?

10. Qua quá trình thực tế, nhóm lập trình viên phát triển Agile đã nghiệm ra những điều gì? Họ xác định giá trị sản phẩm nằm ở khía cạnh con người hay công nghệ? Tính "linh hoạt" được thể hiện như thế nào giữ khía cạnh con người và công nghệ?

CHƯƠNG 2
QUẢN LÝ DỰ ÁN PHẦN MỀM

Quản lý dự án phần mềm là công việc không thuộc qui trình phát triển phần mềm nhưng được thực hiện xuyên suốt từ đầu đến cuối dự án nhằm mục đích quản lý và kiểm soát các hoạt động trong qui trình phát triển phần mềm để phần mềm đạt mức chất lượng đã đặt ra. Trong chương này tôi giới thiệu tổng quan về quản lý dự án phần mềm và một số hoạt động quan trọng trong việc quản lý dự án phần mềm. Đầu tiên là tôi trình bày các lý do cần phải quản lý dự án phần mềm. Nội dung tiếp theo là những khái niệm về dự án, quản lý dự án phần mềm và vai trò của người quản lý dự án. Phần còn lại là trình bày những hoạt động cần thiết của việc quản lý dự án như lập kế hoạch dự án, ước lượng, quản lý rủi ro, quản lý nhân sự.

2.1 ĐẶT VẤN ĐỀ

Trên thế giới, khoảng nửa triệu người quản lý dự án thực hiện khoảng một triệu dự án phần mềm mỗi năm, sản xuất phần mềm trị giá 600 tỷ USD (Nguyễn Công Danh &Trần Cao Đệ, 2013). Nhiều dự án trong số này có chất lượng không như kỳ vọng của khách hàng hoặc không cung cấp các phần mềm trong phạm vi ngân sách và thời gian hoàn thành. Một phân tích cho thấy khoảng một phần ba các dự án có chi phí và thời gian hoàn thành (cost and schedule) vượt hơn 125%. Tại sao quá nhiều dự án phần mềm thất bại? Mặc dù có rất nhiều lý do, một trong những lý do quan trọng nhất là quản lý dự án không phù hợp. Ví dụ, các lý do chính làm cho dự án chệch ra khỏi tầm kiểm soát là mục tiêu không rõ ràng, lập kế hoạch tồi, công nghệ mới, thiếu một phương pháp quản lý dự án, và không đủ nhân sự. Ít nhất ba trong năm lý do này rõ ràng liên quan đến quản lý dự án. Hai lý do còn lại - không đủ nhân sự và công nghệ mới - có thể được coi như những rủi ro. Để quản lý những rủi ro này cũng là một phần công việc của quản lý dự án.

Như vậy, quản lý dự án phần mềm là một phần quan trọng của công nghệ phần mềm. Dự án cần phải được quản lý bởi vì làm công nghệ phần mềm chuyên nghiệp luôn luôn chịu những ràng buộc của tổ chức về ngân sách và lịch biểu. Công việc của người quản lý dự án là đảm bảo cho dự án đáp ứng và vượt qua những hạn chế đó và đồng thời cho ra phần mềm đạt chất lượng cao. Nếu quản lý tốt thì chưa chắc dự án đã thành công, nhưng nếu quản lý tồi thì chắc chắn dự án sẽ thất bại. Dự án thất bại khi phần mềm chuyển giao chậm hơn so với kế hoạch, chi phí lớn hơn dự tính, và không thoả mãn các yêu cầu đề ra. Tiêu chí thành công của quản lý dự án phần mềm phải đạt được các mục tiêu quan trọng như sau (Iam Sommerville, trang 594, 2011):

- Chuyển giao phần mềm đến người dùng đúng thời gian cam kết.
- Giữ chi phí tổng thể trong phạm vi ngân sách.
- Phần mềm đáp ứng được kỳ vọng của khách hàng.
- Duy trì nhóm phát triển vui vẻ và hoạt động tốt.

2.2 CÁC KHÁI NIỆM

2.2.1 Dự án

Dự án là một tập hợp các hoạt động có tổ chức của con người, được thực hiện trong một khoảng thời gian nhất định, với những giới hạn về nguồn lực nhất là nguồn lực về tài chính nhằm đạt được những mục tiêu cụ thể, rõ ràng, và làm thỏa mãn nhu cầu của đối tượng mà dự án hướng đến (Nguyễn Văn Hòa & Hồ Nhã Phong, 2016).

Dự án bao gồm các hoạt động để chuyển giao một sản phẩm cuối cùng. Một số bước phải được thực hiện theo trình tự logic. Dự án phải chịu một loạt các ràng buộc áp đặt lên nó như chi phí, ngân sách và lịch biểu. Sản phẩm của dự án phải đo được theo cách nào đó để kiểm chứng lại phẩm chất của sản phẩm.

Dự án công nghệ thông tin – CNTT là những dự án có liên quan đến phần cứng, phần mềm và mạng. Ví dụ dự án xây dựng hệ thống tính cước và chăm sóc khách hàng tại các Bưu điện Tỉnh/Thành, phục vụ hoạt động sản xuất kinh doanh.

2.2.2 Quản lý dự án

Các tổ chức hay đơn vị thực hiện dự án đều có nguồn lực giới hạn và các hoạt động của dự án phải thỏa mãn các ràng buộc về phạm vi, thời gian và chi phí nên các hoạt động của dự án phải được hoạch định cẩn thận, được giám sát và kiểm soát thường xuyên để phát hiện những bất thường cũng như để đánh giá mức độ hoàn thiện công việc. Các hoạt động này được gọi chung là quản lý.

Quản lý dự án là sự vận dụng các kiến thức, kỹ năng, công cụ và kỹ thuật vào các hoạt động của dự án nhằm đạt được mục tiêu của dự án nhưng vẫn thỏa mãn ba ràng buộc về phạm vi, thời gian và chi phí của dự án[2].

2.3 QUẢN LÝ DỰ ÁN PHẦN MỀM

Quản lý dự án phần mềm tập trung vào các hoạt động trong lập kế hoạch, thực hiện kế hoạch, giám sát và điều phối tài nguyên như kinh phí, con người, thời gian thực hiện, các rủi ro và qui trình thực hiện dự án nhằm đảm bảo thành công cho dự án phần mềm. Quản lý dự án phần mềm cần phải đảm bảo cân bằng ba ràng buộc: thời gian, chi phí và phạm vi.

2.3.1 Vai trò của người quản lý dự án

Sản xuất phần mềm là ngành có tốc độ phát triển nhanh và đem lại nhiều lợi nhuận nhưng sự cạnh tranh của ngành này rất khốc liệt. Khách hàng của những dự án phần mềm luôn mong muốn sẽ nhận được những sản phẩm đúng hạn với đầy đủ yêu cầu đã đề ra với chi phí thấp nhưng có chất lượng cao. Do đó các công ty phần mềm luôn cần những người quản lý dự án phần mềm để đảm bảo các dự án được hoàn thành đúng hạn và đáp ứng các yêu cầu của khách hàng. Vì vậy vai trò và nhiệm vụ của người quản lý dự án rất quan trọng:

- Người quản lý dự án là người chịu trách nhiệm xây dựng kế hoạch của dự án từ việc phân tích yêu cầu của khách hàng cho đến khi sản phẩm được bàn giao.

- Người quản lý là điểm phối hợp duy nhất giữa khách hàng của dự án với đội phát triển dự án để đảm bảo khách hàng sẽ hài lòng ở mức cao nhất khi nhận sản phẩm của dự án.

- Người quản lý dự án cũng là người giải quyết các vấn đề của dự án.

Người quản lý dự án phải có đủ các kiến thức liên quan đến quản lý dự án như phải am hiểu về các kỹ thuật và các công cụ trong quản lý dự án phần mềm, có kinh nghiệm trong lĩnh vực việc quản lý dự án. Đặc biệt, người quản lý dự án phải có kỹ năng lãnh đạo, giao tiếp, đàm phán và giải quyết vấn đề của dự án.

Hầu hết các dự án thất bại vì thiếu quản lý dự án và quản lý con người chứ không phải thất bại vì lý do kỹ thuật.

Hình 17. Các thành viên tham gia dự án

2.3.2 Các lĩnh vực cần quản lý:

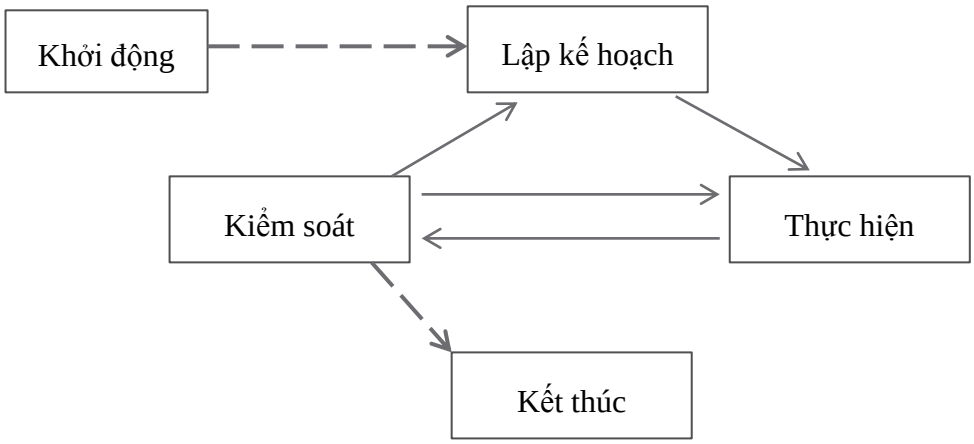
1. Lập kế hoạch tổng thể
2. Quản lý phạm vi
3. Quản lý thời gian
4. Quản lý chi phí
5. Quản lý chất lượng
6. Quản lý nhân lực
7. Quản lý thông tin (truyền thông)
8. Quản lý rủi ro
9. Quản lý hợp đồng và các hoạt động mua sắm

Các vấn đề thường xảy ra đối với dự án:

- Sản phẩm không đạt yêu cầu
- Không hoàn thành đúng hạn
- Chi phí vượt dự toán
- Rủ ro trong quá trình thực hiện.

2.4 CÁC HOẠT ĐỘNG QUẢN LÝ DỰ ÁN

Các hoạt động quản lý dự án được xếp vào 5 nhóm qui trình: khởi động, lập kế hoạch, thực hiện, kiểm soát và kết thúc dự án. Các qui trình này liên kết với nhau, đầu ra của qui trình này sẽ là đầu vào của qui trình tiếp theo.



Hình 18. Các nhóm qui trình quản lý dự án

2.4.1 Lập kế hoạch dự án (project planning)

Phát triển kế hoạch dự án là quá trình văn bản hóa, là công việc quan trọng nhất của một người quản trị dự án. Người quản trị dự án phải chia nhỏ công việc thành nhiều phần và phổ biến các công việc đó đến các thành viên của dự án, lường trước những vấn đề có thể phát sinh và chuẩn bị những giải pháp dự kiến để giải quyết các vấn đề đó.

Bản kế hoạch dự án cho chúng ta biết dự án sẽ được thực hiện, được theo dõi, được giám sát và được kết thúc như thế nào. Nội dung của kế hoạch cũng sẽ thay đổi tùy theo lĩnh vực và sự phức tạp của dự án.

Kế hoạch dự án chính là sơ đồ các nhiệm vụ, thời gian và các mối quan hệ giữa chúng.

Lập kế hoạch dự án thường thực hiện những công việc sau (Lê Văn Tường Lân, 2009):

- Xác định sự phụ thuộc lẫn nhau giữa các công việc.
- Xác định nhân viên tham gia thực hiện.
- Ấn định thời gian hoàn thành cho mỗi công việc bằng các tính toán thời gian hợp lý nhất cho mỗi công việc.
- Định danh hướng đi tới hạn.

- Xem xét lại các tài liệu theo các khía cạnh đầy đủ, nội dung, độ tin cậy và độ chắc chắn.
- Thương lượng, thỏa thuận, cam kết ngày bắt đầu và ngày kết thúc công việc.
- Xác định các giao diện giữa các ứng dụng cần thiết, đặt kế hoạch cho việc thiết kế giao diện chi tiết.

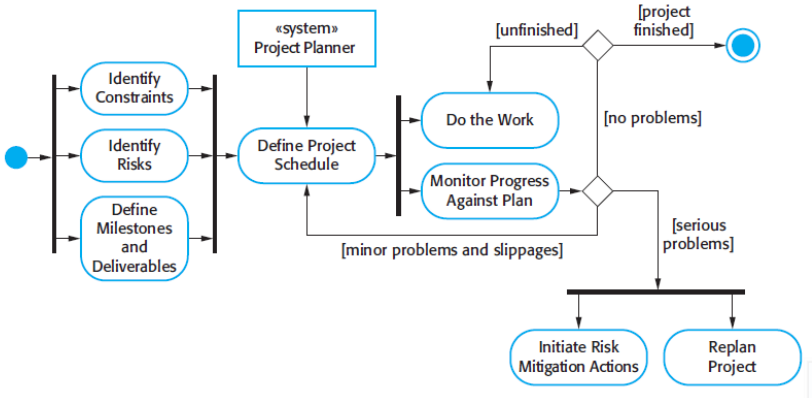
Việc lập kế hoạch cho dự án phần mềm có thể diễn ra trong 3 giai đoạn:

1. Giai đoạn đề xuất: Khi đấu thầu hợp đồng phát triển hoặc cung cấp hệ thống phần mềm thì bản kế hoạch cơ bản được tạo ra để giúp người quản lý quyết định về những tài nguyên (nhân tố con người, máy tính,…) thực hiện công việc và xác định giá thành sản phẩm để báo giá với khách hàng.

2. Giai đoạn khởi động dự án: Bản kế hoạch chi tiết về những ai sẽ tham gia làm việc cho dự án, dự án có thể sẽ gặp những rủi ro gì và cách giải quyết như thế nào, bao nhiêu tài nguyên sẽ phân bổ cho dự án,…Lúc này phải có nhiều thông tin hơn giai đoạn đề xuất. Vì vậy, người quản lý có thể tinh chỉnh, sửa đổi dự toán về nhân lực ban đầu cho dự án.

3. Giai đoạn định kỳ suốt dự án: Người quản lý phải sửa đổi bản kế hoạch của mình theo kinh nghiệm và những thông tin từ sự quan sát thực tế đang diễn ra. Ở giai đoạn này người quản lý học được nhiều hơn về hệ thống sẽ thực hiện và khả năng phát triển của nhóm làm việc. Những thông tin này sẽ cho người quản lý những ước lượng chính xác hơn. Hơn nữa, những yêu cầu của phần mềm thường xuyên thay đổi, có nghĩa là sự cố trong công việc sẽ có thay đổi và lịch trình gia tăng. Do đó, bản kế hoạch tiếp tục được cập nhật và chi tiết hơn về các vấn đề mới phát sinh.

Đối với bản kế hoạch trong giai đoạn đề xuất phần lớn mang tính suy đoán, nên không đầy đủ tất cả các yêu cầu của phần mềm. Người quản lý có trách nhiệm xây dựng bản kế hoạch đề xuất mô tả chi tiết hơn, cấp cao hơn về chức năng của phần mềm ở các giai đoạn tiếp theo.

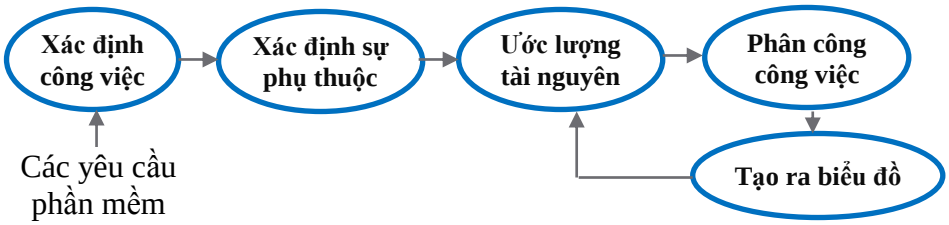


Hình 19. Qui trình lập kế hoạch dự án (Iam Sommerville, trang 625, 2011)

2.4.2 Lập lịch dự án (project scheduling)

Lập lịch dự án là quá trình quyết định thời gian và cách thức thực hiện cho từng nhiệm vụ (task) riêng biệt thuộc dự án. Cần ước tính thời gian cần thiết trong lịch để hoàn thành từng nhiệm vụ, các nỗ lực cần thiết và phải xác định ai sẽ thực hiện nhiệm vụ đó. Người lập lịch phải ước lượng được nguồn tài nguyên cần thiết để hoàn tất nhiệm vụ.

Lịch trình ban đầu thường là kế hoạch về cách thức phân bổ nhân sự cho dự án và kiểm tra sự tiến triển của dự án có bị bất cập với cam kết hợp đồng hay không. Ví dụ, việc lập lịch chia nhỏ công việc của dự án thành nhiều nhiệm vụ riêng biệt và ước lượng thời gian hoàn thành cho từng nhiệm vụ đó. Mỗi nhiệm vụ thường có hạn cuối ít nhất một tuần và không thể dài hơn 2 tháng. Nếu lượng thời gian phân bổ đó tỏ ra không cân đối thì phải sửa kế hoạch lại và cập nhật bản kế hoạch dự án. Lượng thời gian lớn nhất cho bất kỳ một nhiệm vụ nào cũng phải nằm trong khoảng từ 8 đến 10 tuần. Nếu công việc cần thời gian dài hơn thì phải chia nhỏ nhiệm vụ đó ra nữa.



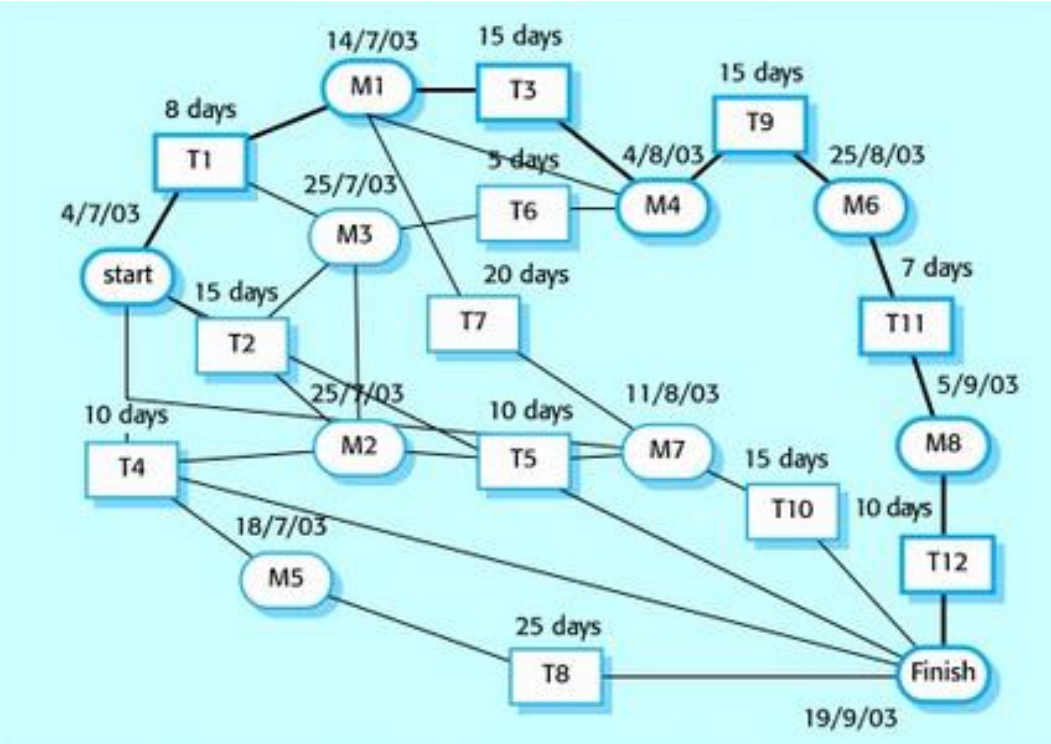
Hình 20. Qui trình lập lịch biểu dự án

Chúng ta sử dụng các ký pháp đồ hoạ để minh hoạ cho lịch biểu của dự án. Sử dụng biểu đồ giúp ta thấy rõ cách chia dự án thành nhiều nhiệm vụ. Chú ý việc bổ sung thêm người vào dự án sẽ làm dự án chậm hơn vì giao tiếp trong dự án trở nên quá tải.

Ví dụ sau đây (Iam Sommerville, trang 629, 2011) liệt kê danh sách các hoạt động Ti (Task i), nhân lực, thời gian thực hiện từng hoạt động và sự phụ thuộc lẫn nhau giữa các hoạt động

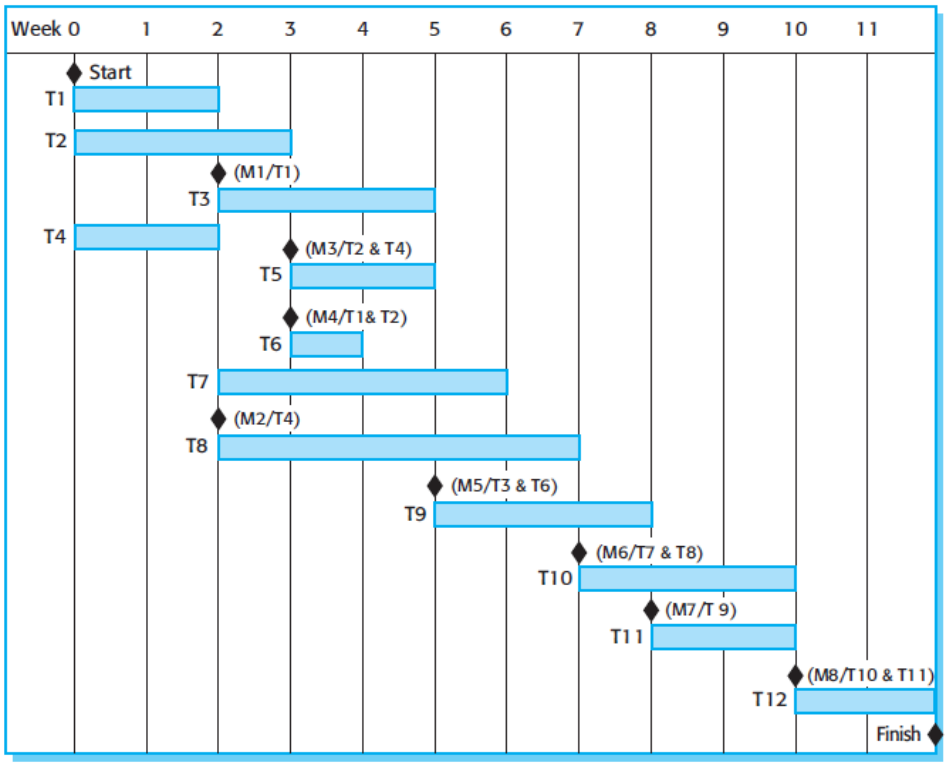
Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Hình 21. Ví dụ danh sách các hoạt động của dự án

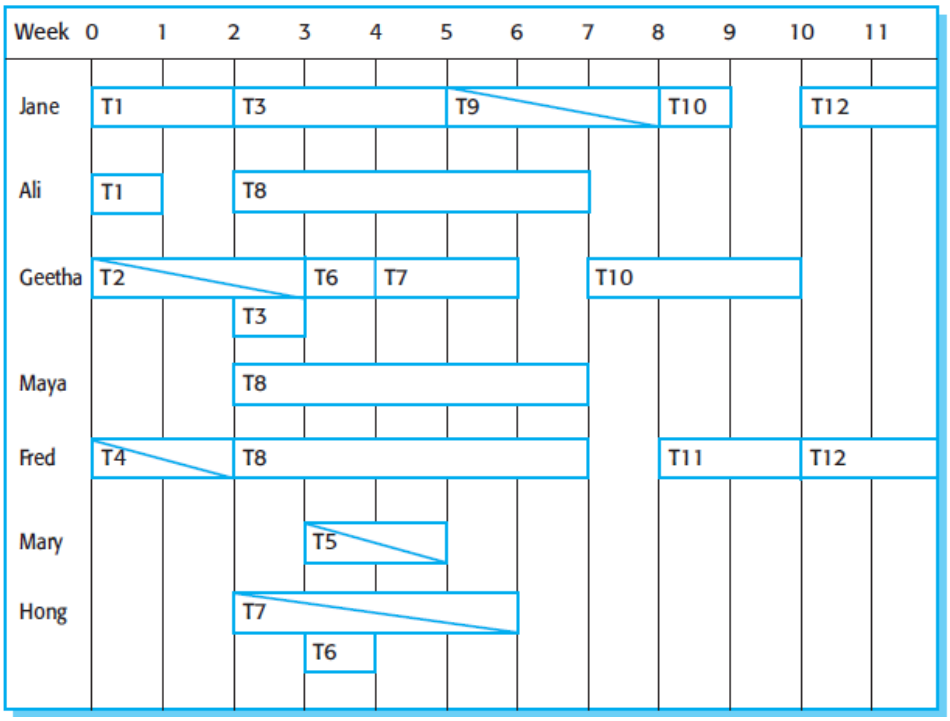


Hình 22. Mạng các hoạt động

Mọi chi tiết trong lịch biểu đều có thời gian dự phòng chứ không nên đóng cứng bởi vì có thể sẽ có nhiều thay đổi trong suốt thời gian làm dự án. Khi một công đoạn nào đó có sự cố xảy ra (ví dụ nhân sự tham gia dự án bị bệnh hoặc bỏ việc, phần cứng bị lỗi, việc cung cấp phần cứng, phần mềm bị trễ hạn,…), hoặc thời hạn lên kế hoạch tỏ ra không phù hợp với thực tế hay có những thay đổi quan trọng trong mục tiêu của dự án thì người quản lý dự án có thể xử lý được mà không làm ảnh hưởng đến toàn bộ lịch trình của dự án.



Hình 23. Biểu đồ thời gian thực hiện các hoạt động



Hình 24. Biểu đồ phân bổ nhân sự

2.4.3 Ước lượng dự án

Việc ước lượng từ lâu đã được xem là vấn đề cốt lõi, có ảnh hưởng trực tiếp đến thành công của dự án phần mềm. Chỉ cần giải quyết được bài toán ước lượng thì năng suất dự án sẽ được nâng cao, đồng thời giảm thiểu rủi ro cho dự án đến mức thấp nhất.

Quá trình quản lý phần mềm được bắt đầu và tiếp diễn bằng một chuỗi các hoạt động ước lượng phần mềm. Trong thực tế, để lấy được dự án phần mềm, các đơn vị tham gia đấu thầu phải nộp hồ sơ dự thầu bao gồm cả chi phí, nhân lực và thời gian phát triển phần mềm.

Để thắng thầu, các đơn vị tham gia dự thầu rất cần phải đưa ra một ước lượng về giá cả, kích cỡ phần mềm, nhân lực và thời gian thực hiện dự án một cách hợp lý. Hợp lý không có nghĩa là ước lượng giá thấp hơn thực tế, vì khi đó công ty sẽ không thu được lợi nhuận hoặc thua lỗ khi hoàn tất dự án. Hợp lý cũng không phải là ước lượng giá cao hơn thực tế, vì khi đó chắc chắn đơn vị sẽ không thể thắng thầu. Do đó, một bảng ước lượng được xem là hợp lý chỉ khi nó phản ánh đúng giá trị thật của đề án. Do đó, ước lượng đã trở thành một công việc khó khăn và phức tạp nhất trong công tác quản lý dự án phần mềm. Các nghiên cứu gần đây chỉ ra rằng ước lượng của dự án phần mềm có độ sai số khoảng 30% (Nguyễn Văn Hòa & Hồ Nhã Phong, 2016). Nhiều lý do sau đây đã tạo ra những khó khăn trong ước lượng:

- Sự phức tạp của dự án phần mềm.

- Phát triển phần mềm được xem như hoạt động thiết kế chứ không liên quan đến hoạt động sản xuất.

- Ước lượng phải được thực hiện ở giai đoạn rất sớm – trước khi các yêu cầu được phân tích.

- Công việc ước lượng chịu rất nhiều áp lực về kinh tế và chính trị

Một ước lượng tốt là một ước tính cung cấp đầy đủ và rõ ràng cho phép lãnh đạo dự án đưa ra quyết định tốt. Người quản lý cần thực hiện ước lượng các chi phí phát triển, thời gian thực hiện, nhân sự tham gia, kích thước dự án.

Một số phương pháp ước lượng như: Phương pháp chuyên gia hay ước lượng dựa trên kinh nghiệm, ước lượng dựa trên thuật toán, phương pháp tương tự (Analogous), phân tích điểm chức năng, phương pháp ước lượng từ dưới lên (Bottom-up), các gói phần mềm hỗ trợ ước lượng…

- **Phương pháp tương tự (Analogous)**

Phương pháp tương tự dùng để ước lượng phần mềm về quy mô, độ phức tạp, nhân lực và thời gian. Theo phương pháp này, số liệu thống kê (metric) và tái sử dụng (reuse) trong qui trình làm công nghệ phần mềm là quan trọng được áp dụng một cách triệt để.

Khi sử dụng phương pháp tương tự đòi hỏi chúng ta phải có dự án phần mềm đã được thực hiện trước đây có bản chất tương tự. Đồng thời, chúng ta cần phải xác định được các thông số tương ứng giữa 2 dự án này. Sau đó áp dụng giải pháp của dự án đã thực hiện trước đó cho dự án mới. Do đó, cần phải thực hiện các bước như xác định các dữ liệu của từng thành phần đã làm nhờ vào số liệu thống kê (metric), tiến hành phân tích dự án mới thành các thành phần, xác định kích thước tương ứng của

từng thành phần với thông tin từ số liệu thống kê và loại trừ kích thước của các thành phần được tái sử dụng (reuse). Cuối cùng tính toán kết quả với hệ số điều chỉnh thì mới xác định được kích thước tổng của dự án.

- ❖ Ưu điểm: Phương pháp này sẽ cho kết quả chính xác khi có đủ dữ liệu cụ thể.
- ❖ Nhược điểm: Không thể thực hiện khi không có dự án tương tự.

➢ **Phương pháp phân tích điểm chức năng**

Phân tích điểm chức năng (Function point analysys) có khả năng định lượng được những giá trị cụ thể trong ước lượng, gần với thực tế dự án. Theo phương pháp này cần tổng hợp các đặc trưng của điểm chức năng như input, output, interface, files,… Sau đó đặt trọng số cho các đặc trưng này. Trọng số ở đây sẽ phụ thuộc vào ngữ cảnh cụ thể của dự án hiện tại như độ phức tạp của bài toán, các yêu cầu về chất lượng, hiệu năng, kích thước của dữ liệu sử dụng.

Phương pháp phân tích điểm chức năng được đề xuất bởi Allan Albertch vào năm 1979. Mục đích của phương pháp này là có thể đưa ra ước lượng số dòng lệnh của hệ thống – LOC (lines of code) dựa trên số lượng điểm chức năng kết hợp với độ phức tạp của điểm chức năng và ngôn ngữ lập trình.

LOC của hệ thống = **FPs** của hệ thống x **LOC/FP** của ngôn ngữ

- ✓ Điểm chức năng - FPs như: Inputs, outputs, queries, files và program interfaces.
- ✓ LOC/FP: Số dòng lệnh cho 1 chức năng theo ngôn ngữ.

Bao gồm các bước:

1. Tính điểm chức năng của 5 loại tiêu biểu:

- Số kiểu người dùng xuất ra: outputs
- Số kiểu người dùng nhập vào: Inputs
- Số kiểu người dùng yêu cầu: queries
- Số giao diện ngoại vi: Interfaces files
- Số File liên quan: Logic files

Số lượng điểm chức năng gắn với độ phức tạp tương ứng, có 3 mức: Thấp, trung bình, cao. Ví dụ các giá trị bên dưới có thể tính được tổng điểm chức năng chưa điều chỉnh (TUFP):

Bảng 3. TUFP cho dự án có mức độ phức tạp: trung bình

Loại chức năng	Độ phức tạp			Tổng
	Thấp	Trung bình	Cao	
Outputs	__x3	5 x 4	__x6	20
Inputs	__x4	6 x 5	__x7	30
Queries	__x3	8 x 4	__x6	32
Logic files	__x7	3 x 10	__x15	30
Interface files	__x5	5 x 7	__x10	35
Total unadjusted function points (TUFP)				**147**

Sau đó tính toán với một số giá trị có liên quan đến các yếu tố ảnh hưởng khác như PCA, TAFP, …ta có được kết quả về tổng điểm chức năng điều chỉnh.

Người quản lý có thể lấy giá trị PCA mặc định như:

- 0.65 : hệ thống rất đơn giản
- 1.00 : hệ thống thông thường
- 1.35: hệ thống phức tạp

Tuy nhiên, số dòng mã lệnh cho mỗi chức năng sẽ phụ thuộc vào ngôn ngữ lập trình được chọn để xây dựng phần mềm. Theo số liệu mới nhất của năm 2013, tỉ lệ LOC/FP như sau (Nguyễn Văn Hòa & Hồ Nhã Phong, 2016):

Bảng 4. Ngôn ngữ lập trình và tỉ lệ LOC/FP

Ngôn ngữ lập trình	Tỉ lệ LOC/FP
Assemply	119
C	97
COBOL	61
C++	50
C#	54
Foxpro	36
HTML	34
Java	53
JavaScript	47
.NET	57
Perl	24
VB.NET	52
Visual Basic	42

Ví dụ: Có 100 điểm chức năng (FP), nếu dùng ngôn ngữ COBOL để thực hiện thì có khoảng 6100 dòng mã lệnh.

- **Mô hình ước lượng COCOMO – Costructive Cost Model**

Mô hình Cocomo là thuật toán phổ biến nhất để ước lượng về nhân lực, thời gian dựa trên kích cở phần mềm (LOC). Mô hình được sử dụng cho các phần mềm lớn.

Sau khi tính được kích thước của dự án bằng phương pháp điểm chức năng, đội dự án có thể sử dụng mô hình Cocomo để ước lượng nhân lực và thời gian.

Phương trình của mô hình Cocomo cơ bản như sau:

- Công sức: $E = a * L^b$
- Thời gian: $T = c * E^d$
- Số người: $N = E/T$

Trong đó: L là số dòng lệnh (KLOC); a, b, c, d là tham số

Các bước tiến hành:

1. Xác định kiểu dự án (cơ sở để chọn tham số):
   - Cấu trúc rõ ràng, môi trường quen: **dễ**
   - Nhiều ràng buộc chức năng, môi trường lạ: **Khó hơn**
   - Hệ thống có cả cứng – mềm, phức tạp, ràng buộc chặt chẽ, cần nhiều kinh nghiệm: **Khó**
2. Phân rã môđun chức năng và ước lượng số dòng lệnh từng môđun
3. Tính lại số dòng lệnh trên cơ sở tái sử dụng
4. Tính nỗ lực phát triển E cho từng môđun
5. Tính lại E dựa trên độ phức tạp của dự án, độ tin cậy, độ lớn của CSDL và yêu cầu tốc độ, bộ nhớ
6. Tính thời gian và số người tham gia

Bảng 5. Cho bảng tham số cơ sở

Kiểu dự án / hệ số	a	b	c	d
Đơn giản	3.2	1.05	2.5	0.38
Trung bình	3.0	1.12	2.5	0.35
Phức tạp	2.8	1.2	2.5	0.32

Ví dụ: Phần mềm có kích cỡ 33.3 KLOC, mức khó là tương đối nên các tham số được chọn: a=3.0 b=1.12 c=2.5 d=0.35

Tính toán ta được:

$E = 3.0 * 33.3^{1.12} = 152$ người-tháng

$T = 2.5*E^{0.35} = 14.5$ tháng

N = E/T = ~ 11 người

- **Cách đơn giản nhất** để ước lượng về thời gian và nhân sự cho dự án là lấy thời gian đã dùng của giai đoạn lập kế hoạch để dự đoán cho các giai đoạn còn lại. Mặc dù cách này có khuyết điểm là ít linh hoạt và thiếu thực tế, cụ thể như sau:

Giai đoạn lập kế hoạch chiếm 15%, giai đoạn phân tích là 20%, giai đoạn thiết kế là 35%, thực thi là 30%.

Ví dụ: planning là 4 person-months → (4 ÷0.15 =26,66)

Vậy: Analysis là 5.33 person-months

Design là 9.33 person-months

Implementation là 8 person-months

- Để giảm thiểu những thiệt hại trong ước lượng, giúp cho việc ước lượng được dễ dàng và đạt được độ chính xác cao hơn người ta có thể áp dụng kết hợp nhiều cách ước lượng trong quá trình thực hiện công việc tính toán ước lượng.

2.4.4 Quản lý rủi ro

Bất kỳ dự án nào cũng phải đối diện với những sự kiện có khả năng tác động đến mục tiêu dự án. Những sự kiện này có thể được dự báo trước hoặc đôi khi không thể dự báo trước. Một khi được dự báo trước, nhóm dự án sẽ có những biện pháp dự phòng chủ động, và vì thế sẽ hạn chế được tác động có thể xảy ra của rủi ro.

Có những rủi ro có thể dễ dàng nhìn thấy trước khi bắt đầu tiến hành dự án, nhưng cũng có những rủi ro chỉ nhìn thấy được khi đã xảy ra. Dự án càng có nhiều sự cố xảy ra càng minh chứng cho việc nhóm dự án không thấy được đa số các rủi ro có thể xảy ra trong dự án, và cũng có thể minh chứng cho việc nhóm dự án có thể thấy được rủi ro nhưng không có biện pháp phòng ngừa hợp lý.

Việc một sự cố xảy ra mà không có kế hoạch dự phòng sẽ dẫn đến mục tiêu dự án bị tác động. Mức độ ảnh hưởng nhỏ sẽ làm trễ tiến độ, vượt ngân sách, giảm chất lượng, rối loạn nội bộ tổ chức, xa hơn nữa có thể làm thất bại toàn bộ dự án, ảnh hưởng đến uy tín tổ chức và quan hệ với khách hàng.

Vì thế, nói đến quản lý rủi ro tức là nói đến việc quản lý chủ động, tích cực hơn là xử lý sự cố, thụ động. Ý nghĩa lớn nhất của quản lý rủi ro là khai phá những rủi ro tiềm ẩn chưa được nhận diện thành những rủi ro có thể nhận diện, song hành với việc phân tích và có giải pháp hợp lý để đối phó với những rủi ro ấy.

Quản lý rủi ro dự án là nỗ lực để giảm thiểu tác động do các rủi ro gây ra. Mục đích của quản lý rủi ro không phải là để tránh đi vào các dự án có rủi ro mà để giảm thiểu các tác động của rủi ro trong các dự án được thực hiện.

Rủi ro là một sự kiện có tính xác suất, nó có thể xảy ra hoặc không xảy ra. Vì lý do này mà chúng ta thường có khuynh hướng lạc quan, không nhìn thấy được các

rủi ro hoặc mong muốn rằng chúng sẽ không xảy ra. Đây là nhân tố làm cho dự án gặp rắc rối nếu các sự kiện rủi ro thực sự xảy ra, nhất là trong những dự án lớn.

Qui trình quản lý rủi ro có hai thành phần chính:

- Đánh giá rủi ro là hoạt động nhận diện các rủi ro có thể xảy ra và đánh giá hậu quả hay mức độ tác động của rủi ro đó đến dự án.

- Kiểm soát rủi ro thông qua việc phát triển kế hoạch quản lý rủi ro và kế hoạch hành động ứng phó với các rủi ro.

Các dạng rủi ro có thể gặp phải trong dự án như rủi ro thị trường (có mang lại lợi nhuận không, bán được không, người dùng chấp nhận không,…), rủi ro tài chính (có đủ tài chính để thực hiện không), rủi ro kỹ thuật (lựa chọn công nghệ nào, phần cứng, phần mềm, mạng,…) rủi ro con người (kỹ năng của nhân viên, kỹ thuật quản lý, kinh nghiệm,…)

2.4.5 Quản lý nhân lực dự án

Trong một tổ chức phần mềm, số lượng nhân viên làm việc rất lớn, song song đó là chi phí tuyển dụng, đào tạo và giữ chân người giỏi của công ty – đây là một bài toán khó và lâu dài trong việc quản lý nhân sự. Điều quan trọng nhất để giữ chân nhân viên chính là ở cách xây dựng văn hóa và môi trường làm việc. Nhà lãnh đạo tài tình là người biết cách trao quyền thực sự cho người dưới quyền. Tức là nhà lãnh đạo phải ý thức được tầm quan trọng của nhân viên đó, biết ủy thác các quyền hạn nhất định để họ có thể làm việc tốt và tin tưởng họ để cho họ làm tốt công việc của mình.

Quản lý nguồn nhân lực dự án bao gồm các quá trình được tiến hành để sử dụng hiệu quả nhất nguồn nhân lực dự án. Quản lý nguồn nhân lực bao gồm 4 quá trình: hoạch định tài nguyên nhân sự, thu nhận nhân sự (bao gồm tìm kiếm và phân công), phát triển đội (xây dựng các kỹ năng cá nhân, kỹ năng nhóm để nâng cao hiệu quả công việc) và quản lý đội.

Để có được sự thành công về kinh tế thì yếu tố con người phải được quan tâm và tôn trọng khi đó họ mới thực hiện tốt trách nhiệm được giao bằng tất cả kỹ năng và kinh nghiệm của mình.

Điều quan trọng của người quản lý dự án phần mềm là am hiểu tốt về kỹ thuật trong việc phát triển phần mềm. Có những kinh nghiệm nhất định trong công việc và một số yếu tố quan trọng như: Cần nhất quán, tôn trọng người khác, trung thực,…

Hầu hết các phần mềm chuyên nghiệp được phát triển bởi các nhóm dự án (project teams). Mỗi nhóm từ hai đến hàng trăm người. Việc quản lý đông người sẽ gặp khó khăn, do đó người ta chia nhóm lớn ra thành những nhóm nhỏ hơn. Mỗi nhóm không quá 10 người và sẽ chịu trách nhiệm thực hiện một phần của hệ thống. Một nhóm được xem là tốt phải có sự đoàn kết và có tinh thần đồng đội.

Thuận lợi của nhóm đoàn kết là nhóm có thể thiết lập chuẩn chất lượng của riêng mình, các cá nhân học hỏi và hỗ trợ lẫn nhau, chia sẻ kiến thức và khuyến khích cải tiến liên tục.

Người quản lý tốt luôn luôn khuyến khích sự gắn kết nhóm thông qua các hoạt động thể thao, trò chơi, tổ chức các sự kiện xã hội. Cách hiệu quả nhất là đối xử với các thành viên của nhóm bằng trách nhiệm và sự tin cậy, tăng cường sự trao đổi thông tin lẫn nhau. Ví dụ trong khi mọi người giải lao cùng nhau cũng là lúc họ có thể giúp nhau học được những công nghệ mới.

Có 3 yếu tố tác động đến nhóm làm việc:

- Nhân sự trong nhóm: Có sự kết hợp nhân sự đa dạng về hoạt động như có khả năng đàm phán với khách hàng, có khả năng lập trình, kiểm thử, lập tài liệu.
- Tổ chức nhóm: Nhóm phải được tổ chức sao cho cá nhân có thể đóng góp tốt nhất khả năng của họ để công việc hoàn thành như dự kiến.
- Giao tiếp kỹ thuật và quản lý: Điều cần thiết là giao tiếp tốt giữa các thành viên, giữa nhóm kỹ sư phần mềm và các bên liên quan.

Một đội hình tốt không thể đảm bảo cho sự thành công của dự án bởi vì còn nhiều vấn đề khác phải đối mặt như sự biến động trong thương mại và môi trường kinh doanh. Tuy nhiên, nếu người quản lý không quan tâm đến thành phần, tổ chức và giao tiếp nhóm thì nguy cơ dự án rơi vào khó khăn là điều không tránh khỏi.

Câu hỏi cuối chương

1. Ở góc nhìn của người quản lý dự án phần mềm, hãy trình bày những lý do làm ảnh hưởng nghiêm trọng đến dự án dẫn đến việc cần phải quản lý dự án phần mềm. Mục tiêu của quản lý dự án phần mềm là gì?

2. Dự án là gì? Dự án công nghệ thông tin là gì? Hãy trình bày khái niệm quản lý dự án.

3. Những khó khăn gì thường xảy ra đối với dự án phần mềm? Hãy liệt kê các lĩnh vực cần quản lý của một dự án phần mềm.

4. Kế hoạch dự án là gì? Lập kế hoạch cho dự án là cần thực hiện những công việc gì? Vì sao mọi chi tiết trong kế hoạch không nên đóng cứng?

5. Vì sao nói nhiều thông tin trong kế hoạch dự án chỉ mang tính suy đoán? Hãy liệt kê những chi phí chính của một dự án phần mềm.

6. Qui trình lập lịch cho dự án là thực hiện những công việc gì? Việc phân bổ thời gian cho các nhiệm vụ trong lịch trình phải được thực hiện như thế nào?

7. Ước lượng có vai trò như thế nào trong dự án phần mềm? Cần phải ước lượng những gì cho dự án phần mềm? Vì sao các giá trị ước lượng phải mang tính **hợp lý**?

8. Mô hình Cocomo dùng để ước lượng các yếu tố nào trong dự án phần mềm? Trình bày cơ sở của mô hình và các bước tiến hành của nó.

9. Trình bày đặc điểm của phương pháp ước lượng tương tự. Phương pháp này có ưu, khuyết điểm gì?

10. Kể tên các cách (phương pháp) ước lượng cho dự án phần mềm. Vì sao nói nên áp dụng kết hợp nhiều phương pháp cho vấn đề ước lượng?

11. Người quản lý cần ước lượng thời gian và số nhân công cho dự án. Theo cách ước lượng đơn giản nhất (lấy thời gian lập kế hoạch để tính thời gian cho các giai đoạn khác), giả sử giai đoạn lập kế hoạch là 3 person-months. Hãy trình bày chi tiết cách tính đó và hãy tính thời gian cho các giai đoạn còn lại.

12. Trong ước lượng, việc phân tích điểm chức năng có thể định lượng và cung cấp các số liệu cụ thể. Hãy trình bày ngắn gọn cách thức thực hiện phân tích điểm chức năng? Ngôn ngữ lập trình có ý nghĩa gì trong vấn đề ước lượng?

13. Hãy kể tên một số rủi ro có thể có của một dự án phần mềm. Vì sao phải quản lý rủi ro? Quản lý rủi ro cần phải thực hiện như thế nào?

14. Việc quản lý nhân sự có ý nghĩa như thế nào trong phát triển phần mềm? Người quản lý nhân sự tốt cần phải có những phẩm chất, hoạt động, kỹ năng gì?

Bài tập thảo luận

QUẢN LÝ DỰ ÁN Ở INFOSYS

Infosys thực hiện hàng trăm dự án mỗi năm. Toàn bộ trách nhiệm để thực hiện một dự án thuộc về người quản lý dự án, những người phải chắc chắn rằng nhóm sẽ giao phần mềm có chất lượng cao cho khách hàng, đúng thời gian và trong giới hạn chi phí.

Infosys là một công ty phần mềm có trụ sở tại Bangalore, Ấn Độ. Nhiệm vụ của nó như đã nêu là "là một công ty toàn cầu cung cấp các giải pháp phần mềm tốt nhất bởi những người giỏi nhất." Nó sử dụng các mô hình phân phối toàn cầu, trong đó khách hàng có thể ở bất cứ nơi nào trên thế giới. Trong mô hình này, khách hàng được tìm kiếm từ bất cứ nơi nào trên thế giới nơi để mà công ty đạt được lợi nhuận cao nhất. Để đáp ứng yêu cầu khách hàng, một sự kết hợp của các qui trình, công nghệ, và quản lý được sử dụng để tách riêng các công việc để mà giá trị đó có thể được thêm vào trong các vị trí tối ưu nhất và sau đó tích hợp lại để giao cho khách hàng.

Infosys hiện đang sử dụng khoảng 10.000 người, với khoảng 15 trung tâm phát triển tại bốn quốc gia và có văn phòng tại hơn một chục quốc gia. Công ty được thành lập năm 1981 bởi bảy chuyên gia phần mềm với nguồn vốn ban đầu chỉ có $ 300. Hôm nay, Infosys có nguồn vốn hơn 8 tỷ đô la (Nguyễn Công Danh & Trần Cao Đệ, 2013). Khách hàng của nó ở khắp toàn cầu và bao gồm tập đoàn lớn - hơn 60 trong số họ là các công ty Fortune 1000 – tức là bao gồm các doanh nghiệp đa dạng như: ngân hàng, bán lẻ, sản xuất, viễn thông, dịch vụ tài chính, bảo hiểm, và giao thông vận tải.

Infosys là một công ty có uy tín cao đã được đánh giá là công ty quản lý tốt nhất và uy tín nhất ở Ấn Độ và là một trong các công ty công nghệ thông tin hàng đầu châu Á (IT). Infosys cung cấp một cơ sở hạ tầng hàng đầu để những người quản lý dự án có thể phục vụ tốt hơn nhu cầu của khách hàng trên toàn thế giới. Công ty đã cung cấp thiết bị cho hội nghị truyền âm thanh cho hầu hết các nhóm để những người quản lý dự án có thể tương tác dễ dàng với khách hàng và với các thành viên trong nhóm nằm ở các vị trí địa lý khác nhau. Tương tự, các phòng hội nghị qua video được sử dụng cho sự tương tác giữa các địa điểm khác nhau của công ty cũng như cho các cuộc họp ảo.

Để cung cấp các dịch vụ khách hàng, Infosys có nhiều đơn vị kinh doanh. Trong mỗi đơn vị kinh doanh, một nhóm (team), đứng đầu là một người quản lý dự án (project manager), phát triển một dự án. Người quản lý dự án chịu trách nhiệm cho tất cả các khía cạnh trong thực hiện dự án, từ việc xác định các yêu cầu để đến cài đặt sau cùng của phần mềm. Người quản lý dự án báo cáo cho người quản lý kinh doanh (business manager) để người này tiếp tục báo cáo cho người đứng đầu đơn vị kinh doanh (business unit head). Để xử lý các tình huống không thể được giải quyết bởi người quản lý dự án, người quản lý cấp cao (senior management) liên quan vào

các dự án là cần thiết. Ở Infosys, người quản lý kinh doanh thường xuyên tương tác với người quản lý dự án và giám sát dự án thông qua các báo cáo tình trạng và báo cáo tại các cột mốc (milestones). Ngoài việc thường xuyên theo dõi, người quản lý kinh doanh cũng giúp giải quyết các vấn đề không thể được xử lý bởi nhóm dự án và phải cần đến cấp độ của mình. Người quản lý kinh doanh cũng tương tác với khách hàng để đảm bảo rằng họ hài lòng và bất kỳ vấn đề nào được đưa ra để giải quyết kịp thời. Ngoài ra, những người cấp cao khác cũng xem xét lại (review) các dự án một cách định kỳ bằng cách thường xuyên tham gia vào cuộc kiểm toán nội bộ. Thông qua hai hệ thống – được gọi là PRISM (xem xét lại (review) dự án bởi người quản lý cấp cao) và IPM (quản lý dự án tích hợp/integrated project management) – các báo cáo ở các cột mốc (milestones) và kế hoạch dự án đã được chuẩn bị sẵn sàng cho người quản lý cấp cao xem xét lại. Tất cả người quản lý cấp cao dự kiến sẽ xem xét một số dự án định kỳ thông qua hệ thống này và để cung cấp thông tin phản hồi đến những người lãnh đạo dự án. Nhìn chung, người quản lý cấp cao duy trì sự liên quan đến dự án chủ yếu bằng cách giám sát để đảm bảo rằng các mục tiêu của dự án được đáp ứng và khách hàng hoàn toàn hài lòng

Qui trình quản lý dự án

Để một nhóm thực hiện một dự án thành công, hàng trăm công việc (tasks) phải được thực hiện, nhiều công việc trong số chúng phụ thuộc lẫn nhau. Quản lý một cách hiệu quả qui trình này là cực kỳ quan trọng để dẫn đến thành công. Ở Infosys, tập hợp các hoạt động được thực hiện bởi người quản lý dự án được xác định cụ thể trong qui trình quản lý dự án. Điều này khá chuẩn, có ba giai đoạn chính:

- Hoạch định dự án
- Thực hiện dự án
- Kết thúc dự án

Trong giai đoạn lập kế hoạch cho dự án (project planning), người quản lý dự án xem xét lại các cam kết trong hợp đồng và tạo ra một kế hoạch để đáp ứng chúng. Việc tạo một kế hoạch cho dự án liên quan đến việc định nghĩa một qui trình vòng đời (chu kỳ sống/lifecycle process) để thực hiện theo, ước lượng nỗ lực (effort) và thời gian hoàn thành (schedule), chuẩn bị thời gian biểu chi tiết cho các công việc, v.v… Nó cũng bao gồm các kế hoạch về chất lượng và quản lý cấu hình cũng như quản lý rủi ro. Trong giai đoạn này, các hoạt động chính của người quản lý dự án như sau:

- Thực hiện các công việc khởi động và quản lý.
- Tạo một kế hoạch cho dự án và ước lượng thời gian hoàn thành (schedule).
  - o Xác định các mục tiêu của dự án.
  - o Xác định một qui trình chuẩn phù hợp để thực hiện dự án.

o Thay đổi qui trình chuẩn để đáp ứng được các yêu cầu (requirements) của dự án.

o Định nghĩa một qui trình để quản lý các thay đổi yêu cầu.

o Ước lượng các nỗ lực (effort).

o Lập kế hoạch cho nguồn nhân lực và tổ chức nhóm.

o Xác định các cột mốc (milestones) của dự án và ước lượng thời gian hoàn thành.

o Xác định các mục tiêu chất lượng và kế hoạch chất lượng để đạt được chúng.

o Lập kế hoạch để phòng ngừa lỗi.

o Xác định rủi ro và lập kế hoạch để giảm thiểu chúng.

o Xác định một kế hoạch đo lường đối với dự án.

o Xác định một kế hoạch đào tạo (training plan) cho dự án.

o Xác định các thủ tục theo dõi dự án.

- Thực hiện xem xét lại các kế hoạch và thời gian hoàn thành dự án.

- Có được ủy quyền từ người quản lý cấp cao.

- Xác định và xem xét lại kế hoạch quản lý cấu hình.

- Nhắc nhở nhóm dự án thực hiện theo kế hoạch quản lý dự án.

Ngoài người quản lý dự án, giai đoạn này liên quan đến các khách hàng, một đại diện của SEPG, và người quản lý kinh doanh của dự án. Tiêu chuẩn nhập là hợp đồng hoặc uỷ quyền dự án đã có. Tiêu chuẩn xuất là kế hoạch dự án đã được lập tài liệu và nhóm đã xem xét lại.

Giai đoạn thứ hai, thực hiện dự án (project execution), liên quan đến việc thực hiện kế hoạch dự án, theo dõi tình trạng của dự án, và làm các hiệu chỉnh bất cứ khi nào hiệu suất của dự án đi chệch với kế hoạch. Nói cách khác, nó liên quan đến việc theo dõi và kiểm soát việc thực hiện theo qui trình của dự án. Giai đoạn này là dài nhất trong qui trình quản lý dự án, kết hợp với các công việc định kỳ như giám sát tình trạng dự án và chất lượng và thực hiện bất cứ hiệu chỉnh nào nếu cần thiết. Trong giai đoạn này, người quản lý dự án thực hiện các hoạt động chính sau:

- Thực hiện dự án theo kế hoạch dự án.

- Theo dõi tình trạng dự án.

- Xem xét lại tình trạng dự án với người quản lý cấp cao.

- Giám sát xem có tuân theo qui trình dự án đã được xác định.

- Phân tích các lỗi và thực hiện các hoạt động ngăn chặn lỗi.

- Giám sát hiệu suất ở mức chương trình (program level).

- Tiến hành xem xét lại tại các cột mốc và hoạch định lại nếu cần thiết.

Các thành viên khác trong nhóm cũng tham gia trong giai đoạn này. Tiêu chuẩn nhập là kế hoạch dự án được hoàn thành và phê duyệt. Tiêu chuẩn xuất là tất cả các sản phẩm công việc (work products) được chấp nhận bởi khách hàng.

Giai đoạn cuối cùng của qui trình quản lý dự án, kết thúc dự án (project closure), liên quan đến việc kết thúc dự án một cách có hệ thống sau khi khách hàng chấp nhận. Mục tiêu chính ở đây là học hỏi từ kinh nghiệm để qui trình có thể được cải tiến. Phân tích dữ liệu của các dự án đã hoàn thành, hình thành các hoạt động chính; số đo được phân tích, các sản phẩm của qui trình (process assets) (nhiều tài liệu, như các bản mẫu (templates) và hướng dẫn (guidelines) được sử dụng để hỗ trợ thu thập và sử dụng trong tương lai, và các bài học được ghi nhận. Bởi vì học từ dự án là mục tiêu chính, đây là một nhóm các hoạt động có liên quan đến người quản lý dự án, SEPG, và các thành viên khác của nhóm. Tiêu chuẩn nhập là các khách hàng đã chấp nhận sản phẩm.

Câu hỏi thảo luận

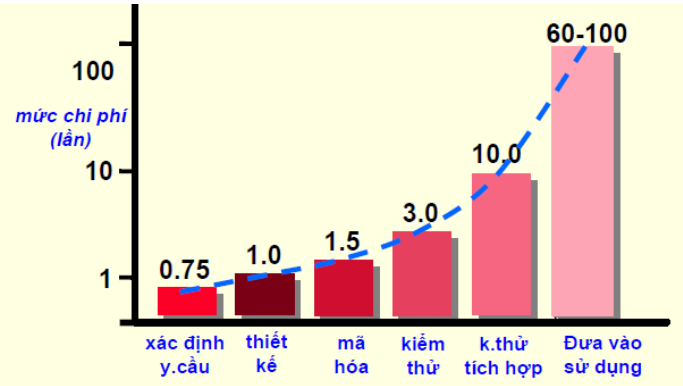
1. Infosys trong bài viết trên là gì? Hoạt động trong lĩnh vực nào? Infosys ở đâu? Qui mô và tầm cỡ của nó như thế nào?
2. Infosys được đánh giá cao ở khả năng nào trong công việc?
3. Người quản lý kinh doanh có vai trò gì trong công việc? Anh ta phải thực hiện việc quản lý của mình như thế nào?
4. Người quản lý dự án và qui trình dự án có quan hệ như thế nào với nhau? 3 giai đoạn quản lý dự án là gì?
5. Những công việc nào có liên quan đến việc lập kế hoạch dự án?
6. Trong quá trình tạo kế hoạch dự án và ước lượng thời gian hoàn thành, Infosys thực hiện những công việc cụ thể gì?
7. Trong 3 giai đoạn khởi động, theo dõi và kết thúc giai đoạn nào diễn ra dài nhất? Trong suốt quá trình thực hiện dự án, kế hoạch dự án không được có sự thay đổi nào. Ý kiến này đúng hay sai? Giải thích.
8. Người quản lý dự án thực hiện những công việc gì trong giai đoạn thực hiện dự án?
9. Giai đoạn kết thúc là thực hiện những công việc gì?
10. Các thông tin trên có liên quan gì đến nội dung chương quản lý dự án phần mềm đã học?

CHƯƠNG 3
PHÂN TÍCH YÊU CẦU PHẦN MỀM

Công việc phân tích yêu cầu phần mềm có thể được xem là công việc chuyển thông tin và yêu cầu của hệ thống thực tế sang giai đoạn thiết kế phần mềm. Do đó nội dung chương này sẽ trình bày các khái niệm về các loại yêu cầu như yêu cầu chức năng, yêu cầu phi chức năng, các ràng buộc… Phần tiếp theo giới thiệu một số hoạt động và kỹ thuật trong phân tích yêu cầu như khảo sát, phân tích (thu thập yêu cầu, xác định yêu cầu, đặc tả yêu cầu, phân tích yêu cầu) và kiểm chứng tính hợp lệ của các yêu cầu hệ thống. Phần cuối chương giới thiệu một số mô hình phân tích để thấy được sự mô tả thông tin yêu cầu của hệ thống thực tế bằng các mô hình phân tích.

3.1 GIỚI THIỆU

Phân tích yêu cầu phần mềm là giai đoạn khá quan trọng trong quá trình phát triển phần mềm. Hoạt động phân tích góp phần quyết định mức độ chất lượng phần mềm đạt được với chi phí dự kiến và thời gian cho trước. Giai đoạn phân tích cần sự phối hợp giữa nhà phát triển và khách hàng trong việc cung cấp và trao đổi thông tin về hệ thống. Kết quả của giai đoạn này là các đặc tả về tính năng của hệ thống dưới dạng văn bản, sơ đồ, mô hình,…và là dữ liệu đầu vào cho giai đoạn thiết kế và viết code. Biểu đồ sau (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009) cho thấy mức chi phí phải trả cho việc tìm và sửa lỗi qua các giai đoạn.



Hình 25. Mức chi phí phải trả do sót lỗi qua các giai đoạn

Dù việc thiết kế và lập trình có tốt đến đâu thì một chương trình được phân tích và đặc tả nghèo nàn sẽ làm cho người dùng thất vọng, đem lại nuối tiếc cho người phát triển và chi phí phải trả cho việc sửa lỗi ngày càng tăng nếu không kịp thời phát hiện các lỗi trong phân tích. Vì vậy, việc hiểu biết đầy đủ về các yêu cầu phần mềm là điều quan trọng cho sự thành công trong nỗ lực phát triển phần mềm.

Một trong những khó khăn trong phân tích và đặc tả là chất lượng thông tin thu thập được từ hệ thống thực tế. Số lượng thông tin trao đổi giữa hai bên khách hàng và nhà phát triển là rất lớn, cơ hội hiểu lầm, hiểu sai là khó tránh khỏi, hay sự mơ hồ trong việc nắm bắt thông tin là chắc chắn có thể xảy ra. Ngoài ra, trong phân

tích ta có thể gặp những khó khăn khác như sự đặc tả nhập nhằng của khách hàng, các thông tin thu thập được mâu thuẫn nhau,...

Nhiệm vụ phân tích yêu cầu là một tiến trình khám phá, làm mịn, mô hình hóa và đặc tả. Quá trình khám phá này đòi hỏi người phát triển phải thực hiện những hoạt động tích hợp, phân tích, đánh giá, giải quyết vấn đề,... Việc làm mịn được thực hiện từ việc lập kế hoạch dự án, đến việc phân tích, thiết kế,… càng về sau càng chi tiết hơn và gần với thực tế hơn. Các mô hình và luồng điều khiển của thông tin, hành vi vận hành và nội dung dữ liệu đều được tạo ra và đặc tả chi tiết trong giai đoạn phân tích này. Kết quả của phân tích làm nền tảng, cơ sở cho giai đoạn thiết kế về sau.

3.2 PHÂN LOẠI YÊU CẦU

3.2.1 Yêu cầu người dùng và yêu cầu hệ thống

Những yêu cầu của một hệ thống là sự mô tả về những chức năng, dịch vụ mà hệ thống sẽ cung cấp và những ràng buộc phải tuân thủ trong hoạt động. Những yêu cầu này phản ánh mục tiêu, nhu cầu của khách hàng trong việc sử dụng dịch vụ của hệ thống như điều khiển thiết bị, thay đổi trật tự hay tìm kiếm thông tin.

Để phân biệt sự khác nhau về cấp độ trong việc đặc tả yêu cầu, người ta chia ra thành hai loại yêu cầu: yêu cầu người dùng và yêu cầu hệ thống.

Yêu cầu người dùng là những khẳng định bằng ngôn ngữ tự nhiên kết hợp với mô hình thể hiện những dịch vụ sẽ cung cấp cho người dùng và những ràng buộc mà hệ thống phải tuân thủ. Các yêu cầu này thường ở mức trừu tượng cao (mang tính tổng quát, chung chung) vì yêu cầu này được viết dành cho khách hàng hiểu về hệ thống.

Yêu cầu hệ thống là những đặc tả **chi tiết hơn** về chức năng, dịch vụ và những ràng buộc trong hoạt động. Tài liệu yêu cầu hệ thống phải được xác định chính xác những gì hệ thống cần thực hiện. Tài liệu này là cơ sở cho việc mời thầu và một phần của bản hợp đồng ký kết giữa khách hàng và người phát triển phần mềm.

Ví dụ sau cho thấy sự khác nhau của hai cấp độ yêu cầu người dùng và yêu cầu hệ thống của một dự án phần mềm "Quản lý chăm sóc sức khỏe bệnh nhân" (MHC-PMS) (Iam Sommerville, 2011):

Đặc tả yêu cầu người dùng: Hàng tháng, hệ thống sẽ tạo ra những báo cáo quản lý về chi phí thuốc được kê đơn bởi phòng khám.

Đặc tả yêu cầu hệ thống (chi tiết hơn):

- Vào ngày làm việc cuối mỗi tháng, hệ thống sẽ tạo ra tổng số thuốc được kê đơn, giá thành của chúng và các đơn thuốc được kê bởi phòng khám.

- Hệ thống tự động tạo ra báo cáo để in vào 17:30 ở ngày làm cuối của tháng.

- Một báo cáo sẽ được tạo ra cho mỗi phòng khám và liệt kê tên của từng loại thuốc cụ thể, tổng số đơn thuốc, tổng số liều qui định và tổng số chi phí thuốc được kê.

- Nếu thuốc có sẵn các đơn vị liều khác nhau (10mg, 20mg) thì tách báo cáo theo mỗi đơn vị liều thuốc.

- Việc truy xuất vào các báo cáo về chi phí sẽ bị nghiêm cấm đối với người dùng không được phép.

3.2.2 Yêu cầu chức năng và yêu cầu phi chức năng

Những yêu cầu hệ thống thường được phân thành hai loại: yêu cầu chức năng và yêu cầu phi chức năng.

Yêu cầu chức năng

Yêu cầu chức năng là những **chức năng**, **dịch vụ** mà hệ thống sẽ cung cấp, hay nói cách khác, yêu cầu chức năng là những hành vi hệ thống đáp ứng lại những dữ liệu đầu vào, cách hệ thống xử lý và trả kết quả ở đầu ra.

Đôi khi một số yêu cầu chức năng cũng không phải là cách tốt nhất để đặc tả đầy đủ về chức năng. Do đó cần kết hợp với các yêu cầu phi chức năng.

Các yêu cầu chức năng được mô tả tùy thuộc vào phần mềm được phát triển, tùy vào sự mong đợi của người dùng và hướng tiếp cận chung của đơn vị viết ra các yêu cầu đó. Khi thể hiện yêu cầu người dùng thì yêu cầu chức năng thường được mô tả một cách tóm lược để người dùng dễ hiểu. Tuy nhiên, các yêu cầu chức năng của hệ thống thì được mô tả một cách chi tiết về chức năng xử lý, hoạt động nhập, xuất, xử lý ngoại lệ,…Yêu cầu chức năng của hệ thống rất đa dạng từ những yêu cầu chung, tổng quát đến các yêu cầu cụ thể rất chi tiết bên trong.

Ví dụ về các yêu cầu chức năng của hệ thống "Quản lý chăm sóc sức khỏe bệnh nhân" MHC-PMS:

1. Người dùng có thể tìm thấy danh sách cuộc hẹn cho tất cả các phòng khám.

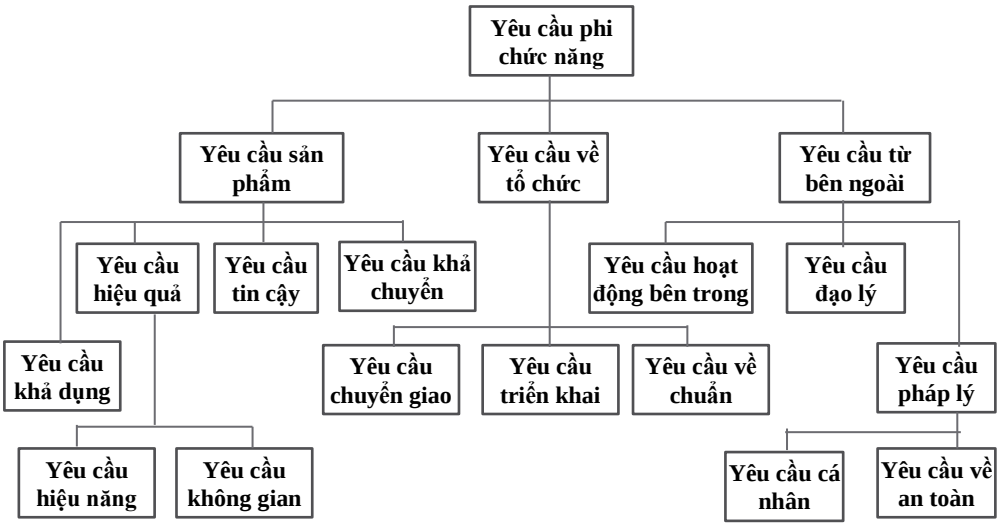
2. Mỗi ngày, ở mỗi phòng khám hệ thống sẽ tạo ra danh sách các bệnh nhân dự kiến tham gia cuộc hẹn ngày hôm đó (các bệnh nhân có thể sẽ đến tái khám).

3. Mỗi nhân viên của hệ thống sẽ được xác định duy nhất bởi 8 ký tự số được qui định trong thẻ nhân viên của mình.

Yêu cầu phi chức năng

Yêu cầu phi chức năng là các đặc điểm chất lượng của chức năng mà hệ thống cần đáp ứng nhằm thỏa mãn nhu cầu người dùng. Đó là những **thuộc tính** và **ràng buộc** của **dịch vụ, chức năng** được hệ thống cung cấp. Những yêu cầu phi chức năng là những yêu cầu không có liên quan trực tiếp đến những dịch vụ cụ thể của hệ thống, nó chỉ liên quan đến những thuộc tính cấp thiết của hệ thống như độ tin

cậy, bảo mật, tốc độ, giao diện, khả năng lưu trữ hoặc xác định những ràng buộc trong thực thi. Những yêu cầu phi chức năng thường đặc tả những thuộc tính ràng buộc.



Hình 26. Các yêu cầu phi chức năng

Trên thực tế sự phân biệt giữa hai loại yêu cầu này chỉ mang tính chất tương đối. Ví dụ, một yêu cầu của người dùng quan tâm đến vấn đề bảo mật như giới hạn quyền truy xuất của người dùng, yêu cầu này có thể là yêu cầu phi chức năng. Tuy nhiên, khi phát triển chi tiết hơn thì yêu cầu này sinh ra một yêu cầu khác thuộc về chức năng như cần các cơ sở xác thực của người dùng.

3.3 PHÂN TÍCH YÊU CẦU

Đây là giai đoạn xác định cái khách hàng cần chứ không phải xác định cái khách hàng muốn (hướng tới), đồng thời phải phân tích càng chính xác càng tốt thực trạng hiện tại của hệ thống.

Một số khó khăn chính thường gặp phải trong phân tích như khách hàng không biết họ cần gì và ngay cả khi khách hàng biết rõ mình cần gì thì cũng sẽ khó khăn khi chuyển tải những thông tin này cho nhà phát triển theo hướng tin học hóa. Hay họ thể hiện yêu cầu theo thuật ngữ riêng làm cho người tiếp nhận không nắm bắt đúng với yêu cầu thực của họ. Tính đa dạng của người cung cấp thông tin có thể có nhiều yêu cầu mâu thuẫn nhau. Nhiều yếu tố khác ảnh hưởng như yêu cầu thường mang tính đặc thù, khó hiểu, khó có chuẩn chung; các yêu cầu thay đổi trong quá trình phân tích (môi trường nghiệp vụ thay đổi, có người mới tham gia); những yếu tố tổ chức và chính sách,… cũng là những khó khăn trong phân tích.

Nhiệm vụ của người phân tích yêu cầu là gợi mở, xác định đúng, đầy đủ, chính xác các yêu cầu. Bắt đầu của giai đoạn phân tích yêu cầu là khi các thành viên của nhóm phân tích yêu cầu tiếp xúc với khách hàng để thu thập yêu cầu. Thông thường thì khách hàng sẽ sắp xếp những buổi phỏng vấn đầu tiên và các buổi phỏng vấn bổ sung thêm cũng được lên lịch trong tiến trình phỏng vấn. Tiến trình phỏng

vấn kết thúc khi nhóm phân tích yêu cầu đã nắm được các thông tin liên quan từ khách hàng và những người sẽ sử dụng sản phẩm trong tương lai.

Người phân tích phải nghiên cứu kỹ các yêu cầu thu thập được từ phía khách hàng để xác định hành vi của hệ thống, xác định chi tiết các yêu cầu làm cơ sở để tiến hành đặc tả chi tiết về hệ thống. Kết quả đặc tả được thảo luận, trao đổi với khách hàng và các bên liên quan để đưa những đánh giá và tinh chỉnh đúng với yêu cầu thực tế của hệ thống.

3.3.1 Khảo sát hiện trạng

Khảo sát hiện trạng được xem là bước phát biểu bài toán, đặt vấn đề hay nghiên cứu sơ bộ. Thông thường hệ thống mới được xây dựng nhằm thay thế một hệ thống cũ đã bộc lộ nhiều bất cập. Người phát triển hệ thống cần làm quen và thâm nhập vào chuyên môn nghiệp vụ mà hệ thống đó phải đáp ứng, tìm hiểu các nhu cầu đặt ra đối với hệ thống, tập hợp các thông tin cần thiết như:

– Hiểu rõ các nghiệp vụ chuyên môn.

–Tìm hiểu vai trò, chức năng, nhiệm vụ, cách thức hoạt động của hệ thống.

– Nêu ra được các chỗ hợp lý của hệ thống cần được kế thừa và những vấn đề bất hợp lý của hệ thống cần phải được nghiên cứu thay đổi.

Ngoài ra, khảo sát hiện trạng cần ghi nhận lại các thông tin về hiện trạng tổ chức, hiện trạng tin học, nghiên cứu các chức trách, nhiệm vụ, các trung tâm ra quyết định và điều hành, sự phân cấp các quyền hạn của hệ thống sẽ phát triển. Xác định cơ cấu tổ chức nội bộ để có cái nhìn tổng thể về một tổ chức hoạt động của hệ thống, cần khảo sát về môi trường của tổ chức, khả năng ảnh hưởng của môi trường làm việc lên hệ thống cần phát triển. Ví dụ: Để xây dựng hệ thống đăng ký học phần trực tuyến của Đại học An Giang, người phát triển phải khảo sát hiện trạng tổ chức của đơn vị này, bộ phận nào thuộc trường đại học sẽ trực tiếp quản lý hệ thống này – Phòng Đào tạo, những bộ phận nào có thể gây ảnh hưởng hoặc bị ảnh hưởng bởi hệ thống sẽ phát triển,…

3.3.2 Thu thập và xác định yêu cầu

Thu thập yêu cầu là xác định các yêu cầu về chức năng, dịch vụ của hệ thống, điều kiện vận hành hệ thống như thiết bị, con người,.. thông qua một số phương pháp thu thập thông tin như phỏng vấn, gửi bản câu hỏi, quan sát, phân tích tài liệu,...

Phỏng vấn có thể khảo sát thu thập thông tin trực tiếp từ người dùng. Mỗi người dùng sẽ có những hiểu biết nhất định về một phần công việc của mình ở hệ thống hiện hành và mong muốn ở hệ thống mới.

❖ Phỏng vấn theo cấu trúc

Phỏng vấn theo cấu trúc cần chuẩn bị sẵn các dạng **câu hỏi đóng** để nêu ra các vấn đề cần quan tâm. Người trả lời sẽ chọn lựa một hoặc nhiều tình huống đã được xác định trước.

Câu hỏi đóng là câu hỏi mà sự trả lời là việc chọn lựa một hoặc nhiều trong những tình huống xác định trước. Do đó, câu hỏi đóng được dùng xác định một tình huống cụ thể.

Ví dụ: "Điều nào dưới đây là tốt nhất đối với hệ thống thông tin Anh (Chị) đang sử dụng?"

□ Dễ dàng truy cập đến tất cả dữ liệu cần

□ Thời gian trả lời tốt nhất của hệ thống

□ Khả năng chạy đồng thời với các ứng dụng khác

Câu hỏi đóng thường được thiết kế theo một trong những dạng sau:

- Đúng – sai
- Nhiều chọn lựa
- Tỉ lệ trả lời: từ xấu đến tốt, từ rất đồng ý đến hoàn toàn không đồng ý. Mỗi điểm trên tỉ lệ nên có một nghĩa rõ ràng, nhất quán và thường có một điểm trung lập ở giữa, xếp hạng các chọn lựa theo thứ tự mức độ quan trọng

Phương pháp này có ích khi người hỏi đã có chủ định điều tra và cần biết rõ các chi tiết. Ưu điểm của phương pháp này là thời gian trả lời ngắn, nội dung tập trung vào chi tiết; khuyết điểm là cần nhiều thời gian chuẩn bị câu hỏi, khó mở rộng được kết quả trả lời.

❖ Phỏng vấn không theo cấu trúc

Phỏng vấn không theo cấu trúc thường là dạng các **câu hỏi mở**, nhằm khuyến khích người trả lời đưa ra được tất cả ý kiến có thể một cách tự do trong khuôn khổ câu hỏi. Do đó, câu hỏi mở dùng để thăm dò, gợi mở vấn đề và người trả lời phải có một kiến thức tương đối. Chẳng hạn như bằng cách hỏi khách hàng "Anh/ chị đang xử lý thông tin gì?" hoặc "Anh/chị có khó khăn gì khi quản lý dữ liệu của mình?".

Với người phỏng vấn nhiều kinh nghiệm, họ có thể đặt các câu hỏi mở rộng sau khi đã lắng nghe cẩn thận và dẫn dắt cuộc nói chuyện đi xa hơn để có nhiều thông tin tốt. Ưu điểm của cách này là không ràng buộc kết quả trả lời, có thể thu thập được ý tưởng mới, thể hiện được ý kiến cá nhân; khuyết điểm là có thể kéo dài thời gian, nội dung trả lời có thể vượt phạm vi câu hỏi.

❖ Khảo sát dùng bảng câu hỏi

Phỏng vấn là một phương pháp hiệu quả để trao đổi và thu thập được những thông tin quan trọng từ phía người dùng. Tuy nhiên, thực hiện phỏng vấn cũng rất tốn kém về thời gian và nguồn lực. Phương pháp khảo sát dùng bảng câu hỏi ít tốn kém hơn, trả lời nhanh hơn, kết quả xác định hơn và thu thập được thông tin từ nhiều đối tượng hơn trong cùng một thời gian ngắn. Tuy nhiên nhược điểm là thụ động, ít thông tin mở rộng, ít mang lại chiều sâu hơn phương pháp phỏng vấn và cần tổng hợp kết quả của tất cả các bảng câu hỏi thu thập được.

Thông thường, người ta kết hợp các phương pháp lại. Trong bất kỳ trường hợp nào khi nhận được trả lời chúng ta nên kiểm tra lại các trường hợp không trả lời để tìm ra nguyên nhân và xem xét kết quả trả lời là hợp lệ và đủ để được chấp nhận không.

❖ Phỏng vấn nhóm

Do kết quả phỏng vấn các đối tượng khác nhau có thể dẫn đến sự không nhất quán thông tin về hệ thống hiện hành và yêu cầu về hệ thống mới. Do đó, chúng ta lại phải thực hiện việc kiểm tra, chọn lọc và quyết định chính xác đâu là thông tin đúng và được chấp nhận cuối cùng. Thông thường chúng ta tiếp tục thực hiện các trao đổi và gặp gỡ các nhân vật quan trọng có thể quyết định và giới hạn được kết quả thông tin. Các cuộc phỏng vấn mới này thường tốn thời gian và có khi lại trả lời lại các câu hỏi mà chúng ta đã được trả lời trước đó bởi những người khác. Do đó, phương pháp phỏng vấn từng cá nhân riêng lẻ vẫn còn những hạn chế nhất định.

Phỏng vấn nhóm là một phương pháp tốt có thể giúp giải quyết được những yêu cầu trái ngược nhau. Các đặc điểm của phỏng vấn nhóm bao gồm:

- Nhiều phân tích viên phụ trách nhiều lĩnh vực khác nhau.
- Nhiều đối tượng phỏng vấn khác nhau, mỗi đối tượng phụ trách một lĩnh vực, có thể phân cấp từ quản lý đến nhân viên trực tiếp liên quan.
- Tổ chức buổi phỏng vấn chung.
- Mỗi phân tích viên có thể đặt câu hỏi và các đối tượng đều có thể trả lời

Ưu điểm: Giảm thời gian phỏng vấn, cho phép các đối tượng phỏng vấn nghe được ý kiến chủ đạo của lãnh đạo trên những ý kiến bất đồng liên quan đến một vấn đề đặt ra. Đây là cơ hội làm cho các đối tượng thông suốt được ý kiến chủ đạo liên quan đến hệ thống mới.

Nhược điểm: Khó tổ chức được buổi phỏng vấn nhóm vì khó tìm được thời gian và vị trí thích hợp cho tất cả mọi người.

❖ Các phương pháp mới xác định yêu cầu

ᔓ Thiết kế kết hợp người dùng (JAD- Join Application Design)

Mục tiêu của JAD là một tiến trình xác định yêu cầu trong đó người dùng, nhà quản lý và các nhà phân tích làm việc với nhau trong một vài ngày diễn ra trong các buổi họp tập trung để xác định hoặc kiểm tra lại các yêu cầu hệ thống các thiết kế chi tiết.

Mục đích chính của JAD trong giai đoạn phân tích là thu thập yêu cầu hệ thống một cách đồng thời từ nhiều đối tượng khác nhau, kết quả là một tiến trình tập trung, có cấu trúc nhưng có hiệu quả cao. JAD cho phép các phân tích viên quan sát và xác định được ở đâu đồng ý và ở đâu có bất đồng trong những người dùng. Các cuộc gặp gỡ diễn ra trong vòng nhiều ngày tạo ra cơ hội để giải quyết bất đồng hoặc ít nhất cũng hiểu được tại sao có bất đồng.

ᘓ Sử dụng bản mẫu (prototype) để xác định yêu cầu

Sử dụng bản mẫu như một kỹ thuật xác định yêu cầu, phân tích viên làm việc với người dùng để xác định các yêu cầu cơ bản và ban đầu của hệ thống. Sau đó, phân tích viên dựa trên yêu cầu này để xây dựng một bản mẫu ban đầu. Bản mẫu khi hoàn thành sẽ gửi đến người dùng để sử dụng thử và kiểm tra. Đặc biệt, việc trực quan hóa các mô tả yêu cầu bằng lời được chuyển đổi thành hệ thống vật lý sẽ nhắc nhở người dùng thay đổi những yêu cầu tồn tại không phù hợp và phát sinh những yêu cầu mới. Kết quả thử nghiệm của người dùng sẽ được phản hồi tới phân tích viên và phân tích viên sẽ dùng thông tin phản hồi này để cải tiến bản mẫu rồi tiếp tục gửi đến người dùng và vòng lặp này cứ tiếp tục như vậy.

Ví dụ: Trong buổi phỏng vấn ban đầu, người dùng muốn xây dựng một form nhập hóa đơn với tất cả thông tin về khách hàng, hóa đơn, dịch vụ, hàng hóa, quá trình thanh toán,… theo cách nghĩ của người dùng là tiện lợi. Tuy nhiên, sau khi sử dụng bản mẫu, người dùng sẽ cảm thấy phức tạp, lẫn lộn và sẽ thay đổi yêu cầu với nhiều form khác nhau và cần sự di chuyển hợp lý giữa các form.

Sau khi thu thập thông tin thì việc xác định yêu cầu là phải xây dựng được danh sách các yêu cầu và được thực hiện bằng ngôn ngữ tự nhiên nhằm để khẳng định các yêu cầu, mô tả trừu tượng các dịch vụ mà hệ thống cần cung cấp và các ràng buộc trong vận hành của hệ thống. Các yêu cầu phải phản ánh chính xác điều mà người dùng muốn.

Các yêu cầu là những đặc tả sơ bộ bên ngoài của hệ thống và không liên quan đến các đặc tính thiết kế bên trong. Phần tài liệu này được viết sao cho người đọc có thể hiểu được mà không cần một kiến thức chuyên môn đặc biệt nào.

3.3.3 Đặc tả yêu cầu

Đặc tả yêu cầu là quá trình viết ra các **yêu cầu người dùng** và **yêu cầu hệ thống** trong tài liệu yêu cầu một cách rõ ràng, dễ hiểu, đầy đủ và thống nhất. Trên thực tế đây là một điều khó khăn để đạt được bởi vì một số lý do như: những yêu cầu thường hay có những xung đột và mâu thuẫn nhau, các bên liên quan thì hiểu các yêu cầu theo cách riêng của họ. Vì vậy những người dùng hệ thống khi đọc thì họ không hiểu được bởi họ không có kiến thức về kỹ thuật. Cách tốt nhất là tài liệu yêu cầu của người dùng chỉ mô tả những hành vi bên ngoài của hệ thống, không mô tả chi tiết về kiến trúc, thiết kế sâu bên trong. Do đó, nếu xây dựng yêu cầu người dùng thì không nên sử dụng biệt ngữ phần mềm, những ký hiệu về cấu trúc hay ký hiệu hình thức mà nên viết bằng ngôn ngữ tự nhiên, các bảng, biểu mẫu đơn giản, sơ đồ trực quan.

Do đó, yêu cầu hệ thống là phiên bản mở rộng của yêu cầu người dùng, để dành cho các kỹ sư và là phần mở đầu của giai đoạn thiết kế hệ thống. Yêu cầu này được bổ sung chi tiết và diễn giải cách thức người dùng yêu cầu đối với hệ thống. Nó phải được đặc tả một cách chi tiết, hoàn chỉnh cho toàn bộ hệ thống.

Dựa trên việc xác định các yêu cầu, người phát triển đưa ra các đặc tả cho hệ thống để trả lời các yêu cầu sau:

- Đầu ra của hệ thống là gì?
- Hệ thống sẽ phải làm gì để có kết quả mong muốn, nghĩa là phải xử lý những cái gì?
- Những tài nguyên mà hệ thống yêu cầu là gì?

Người phân tích phải hiểu rõ nguồn gốc, các dạng thông tin cần cung cấp cho hệ thống hoạt động, các vấn đề mà hệ thống phải giải quyết, những kết quả mà hệ thống phải đưa ra. Xác định được mối quan hệ giữa cái vào và cái ra cho quá trình hoạt động của hệ thống. Các đặc tả chi tiết này nhằm phục vụ cho việc kiểm tra xem những nhiệm vụ đặt ra có hoàn tất được không, có mâu thuẫn nhau không. Nhiều vấn đề khác cũng cần được quan tâm xử lý như:

- Các hệ thống phần mềm lớn luôn đòi hỏi **cải tiến** từ hệ thống hiện hành. Mặc dù các khó khăn của hệ thống hiện tại có thể xác định được nhưng các ảnh hưởng và hiệu ứng của hệ thống mới khó có thể dự đoán trước được.
- Hệ thống lớn thường có nhiều cộng đồng sử dụng khác nhau, họ có các yêu cầu và ưu tiên khác nhau. Các yêu cầu hệ thống cuối cùng không tránh khỏi các thỏa hiệp.

Ngôn ngữ tự nhiên không hoàn toàn thuận tiện trong việc đặc tả yêu cầu bởi nhiều lý do như dễ gây nhầm lẫn vì cách hiểu về khái niệm khác nhau giữa hai bên, hay một vấn đề có thể được mô tả bằng quá nhiều cách, hoặc các yêu cầu không được phân hoạch tốt, khó tìm mối quan hệ,… Do đó người ta có thể đặc tả yêu cầu bằng một số cách khác như dùng ngôn ngữ tự nhiên có cấu trúc, ngôn ngữ mô tả thiết kế - giống với ngôn ngữ lập trình nhưng có mức trừu tượng cao hơn, ngôn ngữ đặc tả yêu cầu, đặc tả toán học,…

3.3.4 Đánh giá yêu cầu

Đánh giá các yêu cầu phần mềm liên quan đến việc cho biết các yêu cầu đã được xác định, đặc tả có thực sự đáp ứng đầy đủ đòi hỏi của khách hàng chưa. Nếu việc đánh giá này không chính xác, các lỗi trong phần yêu cầu sẽ truyền tới giai đoạn thiết kế và giai đoạn triển khai hệ thống. Khi đó chi phí sửa chữa lỗi sẽ rất lớn. Sự thay đổi trong phân tích yêu cầu sẽ kéo theo sự thay đổi trong thiết kế và triển khai.

Một số khía cạnh về đánh giá yêu cầu cần được triển khai như thỏa hiệp giữa các nhu cầu có sự xung đột, mọi yêu cầu không được mâu thuẫn với các yêu cầu khác, mọi chức năng và ràng buộc phải được đặc tả và định nghĩa một cách hoàn chỉnh,…

3.3.5 Tài liệu hóa yêu cầu

Các yêu cầu hệ thống được trình bày trong tài liệu các yêu cầu phần mềm là những thứ mà người phát triển hệ thống cần biết. Tài liệu này bao gồm các các định nghĩa và các đặc tả về yêu cầu.

Tài liệu yêu cầu không phải là tài liệu đặc tả. Nó mô tả cái hệ thống cần phải làm chứ không phải mô tả cách thức làm. Tài liệu này cần dễ dàng đặc tả và ánh xạ sang các phần tương ứng của thiết kế hệ thống. Nếu các dịch vụ, ràng buộc và các đặc tả thuộc tính trong tài liệu yêu cầu phần mềm được thỏa mãn bởi thiết kế thì thiết kế này được coi là giải pháp thích hợp với vấn đề.

Về nguyên tắc các yêu cầu cần được hoàn chỉnh và chắc chắn. Mọi chức năng hệ thống cần được đặc tả và các yêu cầu không được mâu thuẫn.

3.4 CÁC MÔ HÌNH PHÂN TÍCH

Mô hình là một dạng trình bày đơn giản hóa của thế giới thực. Bởi vì hệ thống thực tế thì rất phức tạp và rộng lớn nhưng khi tiếp cận hệ thống sẽ có những chi tiết, những mức độ phức tạp không cần thiết phải được mô tả và giải quyết. Mô hình cung cấp một phương tiện để quan niệm hóa vấn đề và giúp chúng ta có thể trao đổi các ý tưởng trong một hình thức cụ thể trực quan, không mơ hồ.

Các mô hình hóa sử dụng trong phân tích thiết kế là các dạng ngôn ngữ đồ họa (các sơ đồ - diagram), các ngôn ngữ này bao gồm một tập hợp các ký hiệu đi kèm các qui tắc giúp cho việc trao đổi thông tin phức tạp được rõ ràng hơn việc mô tả bằng văn bản.

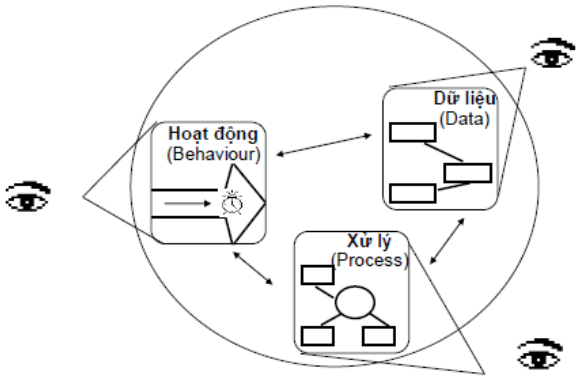
3.4.1 Vai trò của mô hình phân tích

- Mô hình trợ giúp cho người phân tích trong việc hiểu về thông tin, chức năng và hành vi của hệ thống.

- Mô hình là thành phần tiêu điểm, mấu chốt để xác định tính đầy đủ, nhất quán và tính chính xác trong đặc tả.

- Mô hình là nền tảng cho giai đoạn thiết kế, cung cấp cho người thiết kế một cách biểu diễn chủ yếu về phần mềm có thể ánh xạ vào sản phẩm cài đặt.

Có nhiều phương pháp có thể cấu trúc hóa yêu cầu như: **phương pháp phân tích hướng cấu trúc**, **phương pháp hướng đối tượng**.



Hình 27. Nhìn hệ thống từ nhiều góc độ

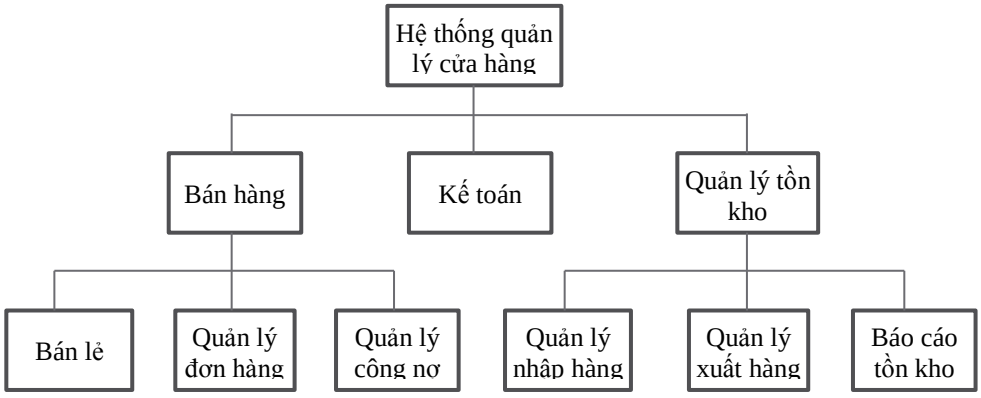
Các yêu cầu của người sử dụng thường được viết bằng ngôn ngữ tự nhiên để những người không có kiến thức về mặt kỹ thuật có thể hiểu được nó. Tuy nhiên, những yêu cầu hệ thống chi tiết phải được mô hình hóa. Mô hình hóa hệ thống giúp cho người phân tích hiểu rõ các chức năng của hệ thống.

Ta có thể sử dụng các mô hình khác nhau để biểu diễn hệ thống từ nhiều khía cạnh khác nhau.

3.4.2 Các loại mô hình phân tích

a. Mô hình tổ chức

Mô hình tổ chức sau mô tả chức năng tổng quát các bộ phận quản lý của hệ thống cửa hàng nước giải khát. Cửa hàng chia thành nhiều bộ phận để việc quản lý được dễ dàng hơn như quản lý việc bán hàng, quản lý tiền bạc, quản lý hàng hóa (Nguyễn Văn Vy & Nguyễn Việt Hà, 2009).

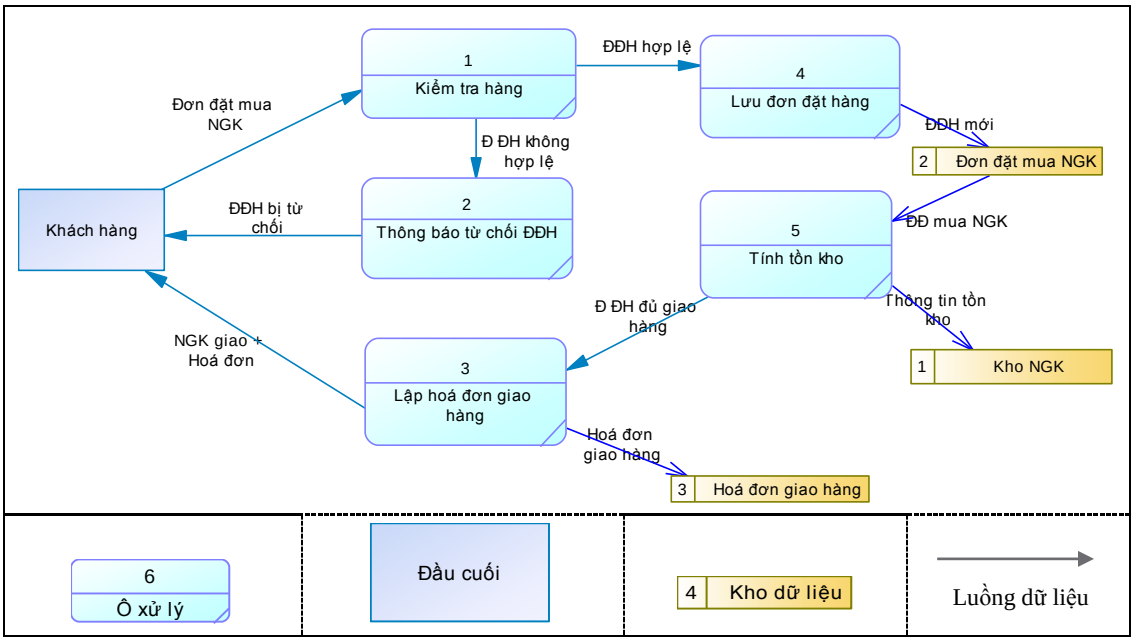


Hình 28. Ví dụ sơ đồ cây chức năng

b. Mô hình dòng dữ liệu (DFD)

Mô hình DFD (Data Flow Diagram) được sử dụng để mô hình hóa qui trình xử lý dữ liệu của hệ thống. Mô hình này được sử dụng để mô tả quá trình hoạt động xử lý thông tin nghiệp vụ, biểu diễn cách thức **dữ liệu di chuyển**, **được xử lý**, **lưu trữ** trong hệ thống và **trao đổi** với môi trường. Mô hình này có thể được gọi là mô hình chức năng và dòng thông tin.

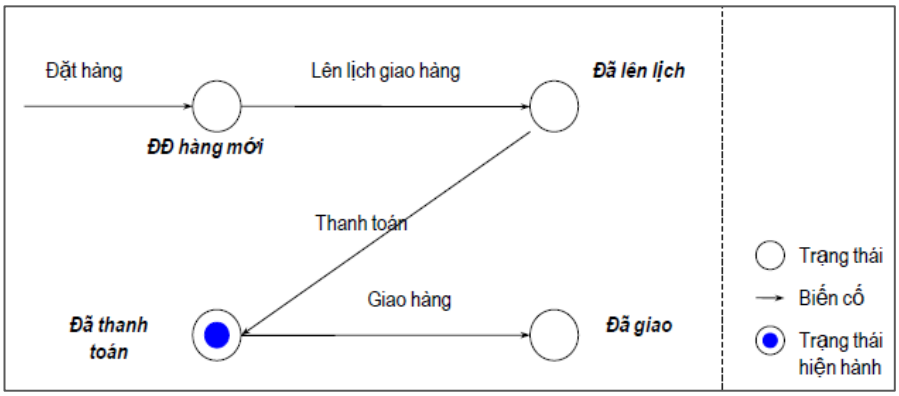
- Mô tả dòng thông tin di chuyển (flow) xuyên qua các hệ thống.
- Diễn tả các tương tác xuất nhập dữ liệu với con người và các hệ thống khác
- Mô hình dòng dữ liệu DFD cung cấp 4 ký hiệu cơ bản để mô tả sự di chuyển , xử lý, lưu trữ và trao đổi thông tin của hệ thống.
- Mô hình DFD có nhiều mức trừu tượng khác nhau, mức sau chi tiết hơn mức trước.



Hình 29. Mô hình DFD xử lý đặt hàng - cửa hàng nước giải khát

c. Mô hình hành vi- mô hình trạng thái (STD)

Mô hình trạng thái mô tả đáp ứng của hệ thống với các sự kiện bên trong và bên ngoài của nó. Mô hình trạng thái biểu diễn các trạng thái khác nhau của hệ thống và các sự kiện gây ra sự dịch chuyển trạng thái. Mô tả chi tiết hơn điều kiện xảy ra của các hành vi và cung cấp một hình ảnh động về hệ thống



Hình 30. Ví dụ các trạng thái của một đơn đặt hàng nước giải khát

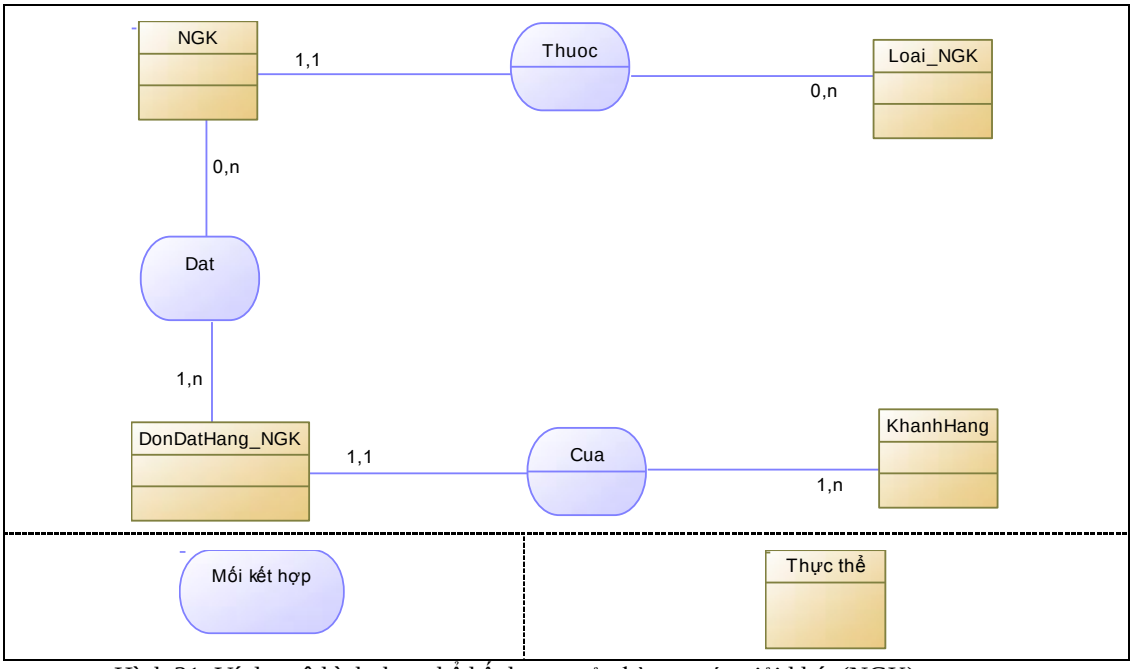
Hai mô hình DFD và mô hình STD biểu diễn những góc nhìn khác nhau, nhưng cả hai đều cần thiết để mô tả các ứng xử của hệ thống.

d. Mô hình thực thể kết hợp (ERD)

Mô hình thực thể kết hợp được sử dụng để đặc tả các thông tin về dữ liệu của hệ thống, cấu trúc dữ liệu và các quan hệ và ràng buộc dữ liệu.

Chúng ta sử dụng mô hình ERD để thiết lập các thực thể của hệ thống, quan hệ giữa các thực thể, thuộc tính của các thực thể và ràng buộc dữ liệu. Mô hình này được sử dụng trong thiết kế cơ sở dữ liệu (CSDL) và thường được cài đặt trong các CSDL quan hệ.

Tuy nhiên, mô hình dữ liệu thường không chi tiết. Cho nên, chúng ta có thể sử dụng từ điển dữ liệu làm công cụ bổ trợ. Từ điển dữ liệu là danh sach tất cả các tên gọi được sử dụng trong các mô hình hệ thống như các thực thể, quan hệ và các thuộc tinh. Ưu điểm của từ điển dữ liệu là hỗ trợ quản lý tên và tránh trùng lặp tên, lưu trữ kiến thức một cách có tổ chức kết nối pha phân tích, thiết kế và cài đặt.



Hình 31. Ví dụ mô hình thực thể kết hợp - cửa hàng nước giải khát (NGK)

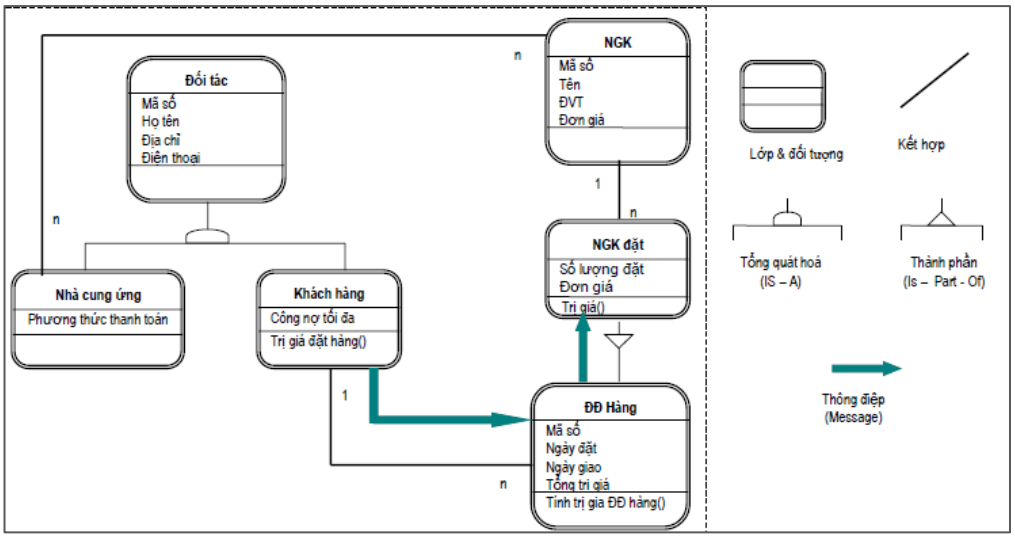
e. Mô hình đối tượng

Sử dụng mô hình DFD hay mô hình ERD thường rất khó mô tả các vấn đề có liên quan đến thế giới thực. Mô hình đối tượng đã giải quyết được vấn đề này bằng cách kết hợp ứng xử và dữ liệu thành đối tượng.

Mô hình đối tượng được sử dụng để biểu diễn cả dữ liệu và quy trình xử lý của hệ thống. Nó mô tả hệ thống dựa theo thuật ngữ các lớp đối tượng và các quan hệ của nó. Một lớp đối tượng là sự trừu tượng hóa trên một tập các đối tượng có thuộc tính và phương thức chung.

Mô hình đối tượng phản ánh các thực thể trong thế giới thực được vận dụng trong hệ thống. Nếu ta càng có nhiều thực thể trừu tượng thì việc mô hình hóa càng khó khăn. Phát hiện các lớp đối tượng là một quy trình rất khó khăn khi tìm hiểu sâu về lĩnh vực của ứng dụng. Các lớp đối tượng thường phản ảnh các thực thể liên quan tới miền ứng dụng của hệ thống.

Các mô hình đối tượng bao gồm: mô hình thừa kế, mô hình kết hợp và mô hình ứng xử.



Hình 32. Mô hình hướng đối tượng – lớp (class)

Từ điển dữ liệu

Trong quá trình phân tích sẽ có rất nhiều phần tử được tạo ra trong mô hình phân tích: dữ liệu, chức năng, điều khiển… cần phải có một cách thức quản lý các phần tử đó sao cho hiệu quả, đồng thời tạo được sự thống nhất, chính xác về thông tin giữa người tạo ra mô hình và người sử dụng mô hình. Do đó từ điển dữ liệu được sử dụng trong giai đoạn này.

Từ điển dữ liệu là một danh sách có tổ chức của tất cả các phần tử dữ liệu cần thiết cho hệ thống. Các phần tử được định nghĩa chính xác và chặt chẽ sao cho cả phân tích viên và khách hàng cùng chia sẻ một suy nghĩ về chúng.

Câu hỏi cuối chương

1. Trình bày vai trò của giai đoạn phân tích yêu cầu. Kết quả của giai đoạn này là gì? Vì sao chất lượng thông tin thu được từ thực tế là vấn đề khó khăn trong giai đoạn này?

2. Yêu cầu người dùng và yêu cầu hệ thống (gọi chung là yêu cầu phần mềm) dùng để làm gì? Phân biệt sự khác nhau giữa hai loại yêu cầu này.

3. Nhiệm vụ của giai đoạn phân tích yêu cầu là gì? Hãy cho một ví dụ để thấy được sự khác nhau giữa yêu cầu người dùng và yêu cầu hệ thống.

4. Trình bày ngắn gọn yêu cầu chức năng và yêu cầu phi chức năng. Trong hai yêu cầu này, yêu cầu nào được áp dụng trên toàn bộ hệ thống? Yêu cầu nào mang tính quyết định hơn, vì sao?

5. Hãy nêu và trình bày ngắn gọn các cách thu thập yêu cầu về hệ thống?

6. Đặc tả yêu cầu phần mềm là gì? Nêu những khó khăn khi đặc tả yêu cầu. Tại sao cần phân biệt yêu cầu hệ thống và yêu cầu người dùng?

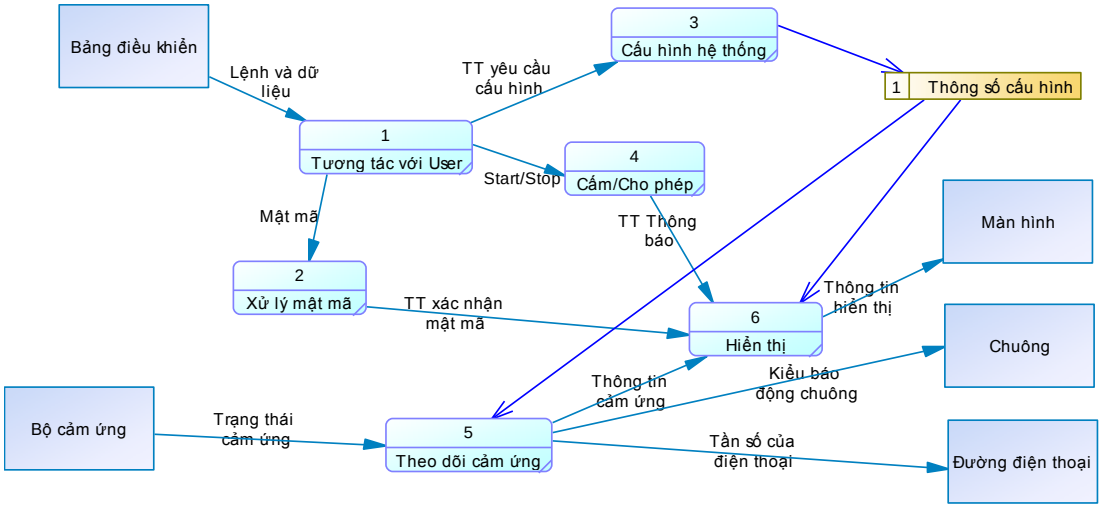
7. Mô hình phân tích hệ thống là gì? Trình bày vai trò của mô hình phân tích hệ thống. Liệt kê tên các mô hình phân tích phổ biến.

8. Mô hình dòng dữ liệu (DFD) biểu diễn những vấn đề nào của hệ thống? Mô hình dữ liệu (ERD) đặc tả những thông tin nào của hệ thống? Từ điển dữ liệu có ý nghĩa gì trong các giai đoạn phát triển phần mềm?

Bài tập thảo luận

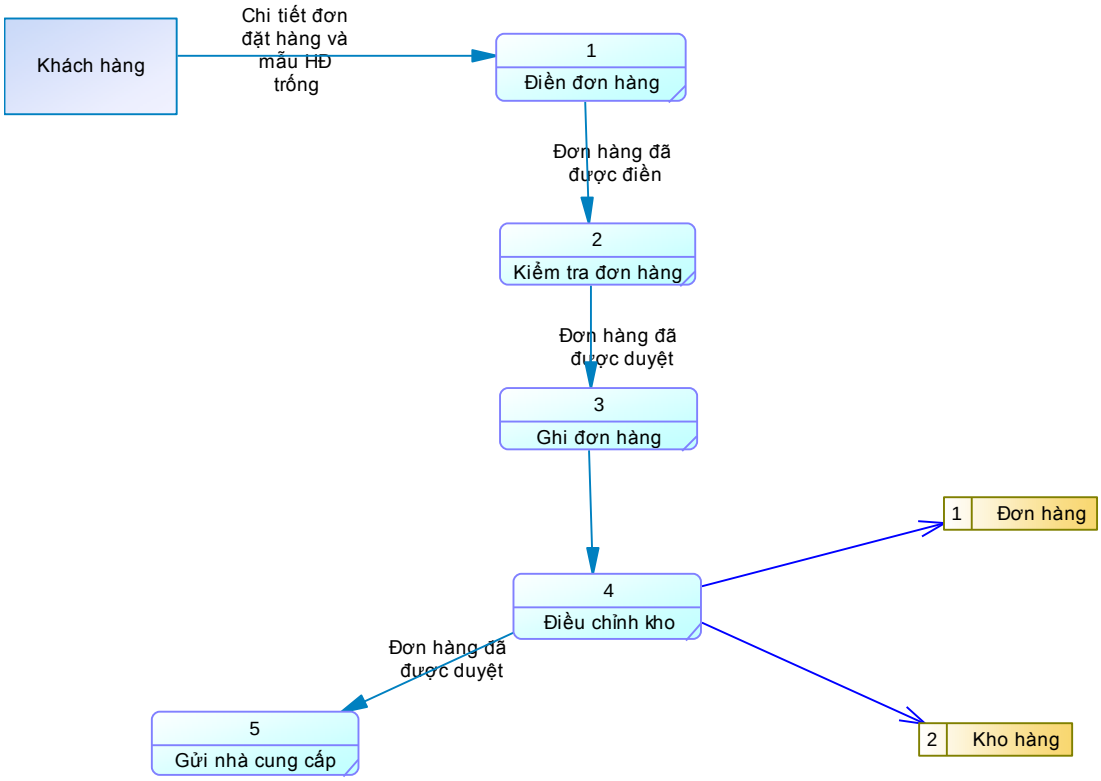
Hãy gọi tên các mô hình sau đây. Mỗi mô hình hướng về vấn đề gì trong phân tích? Từng mô hình đã cho chúng ta biết những thông tin gì?

a. Hệ thống SafeHome



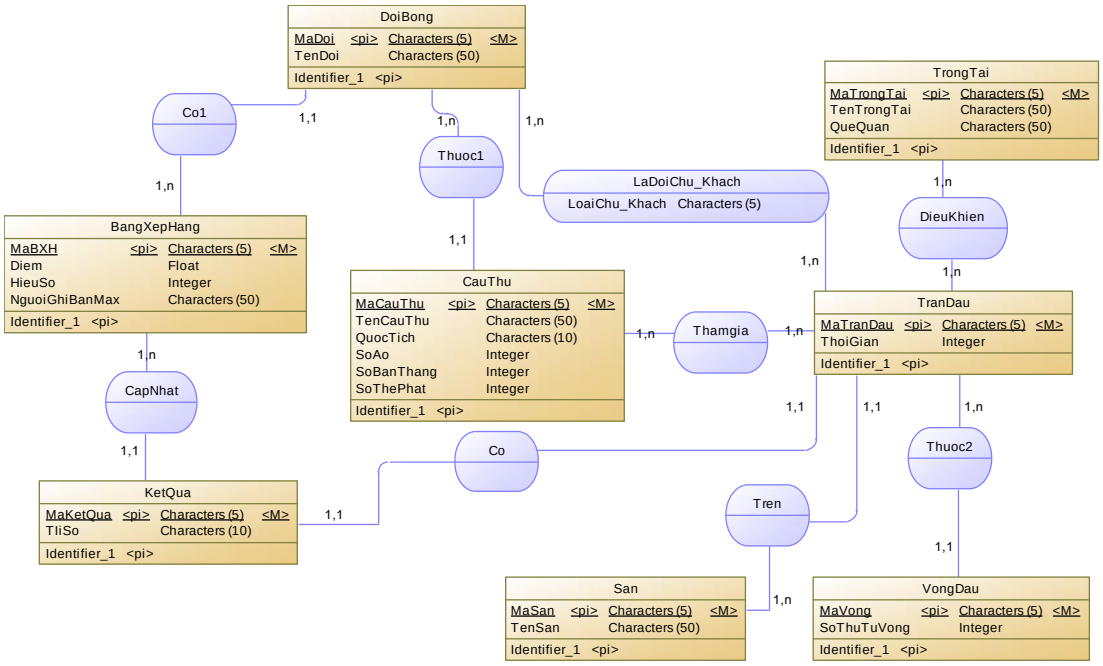
Hình 33. Mô hình - SafeHome

b. Hệ thống bán hàng



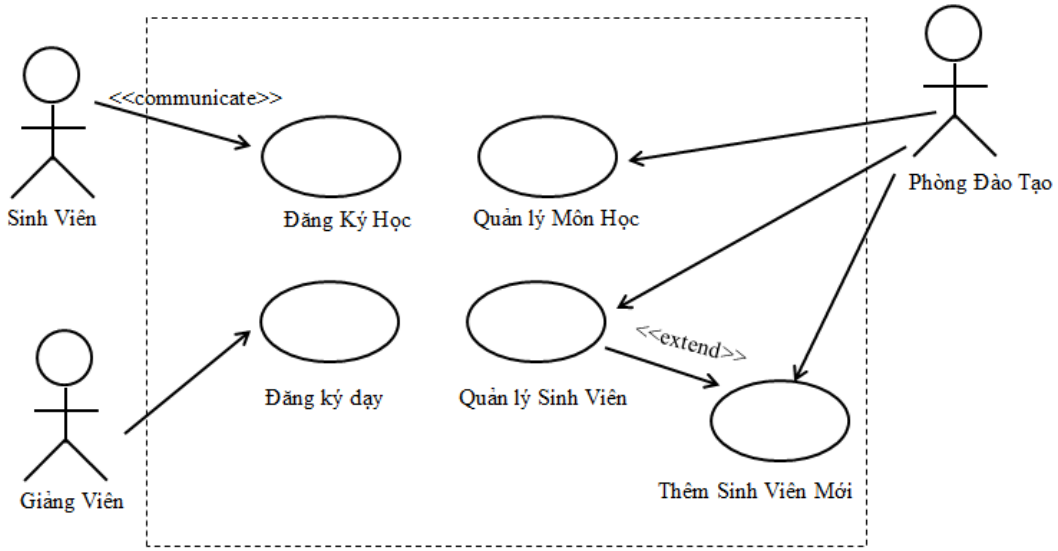
Hình 34. Mô hình xử lý đơn hàng

c. Hệ thống quản lý giải bóng đá

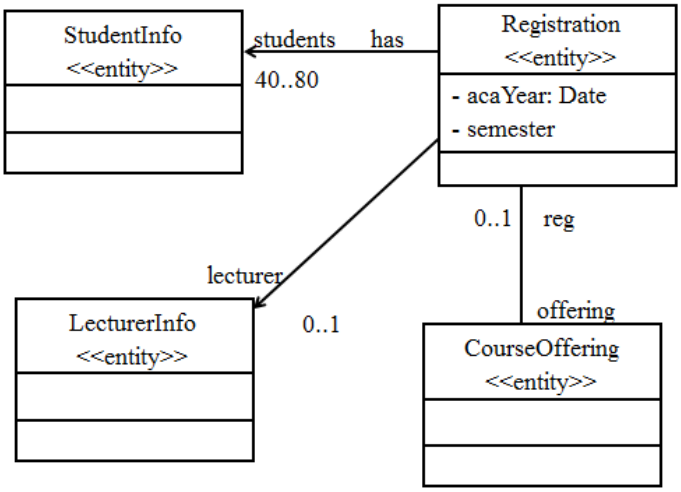


Hình 35. Mô hình quản lý giải bóng đá

d. Hệ thống đăng ký học phần trực tuyến



Hình 36. Mô hình đăng ký học phần trực tuyến



Hình 37. Mô hình hướng đối tượng

CHƯƠNG 4
THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

Chương 3 đã giới thiệu về giai đoạn phân tích – giai đoạn đầu trong qui trình phát triển phần mềm. Kết quả giai đoạn này dùng làm dữ liệu đầu vào cho giai đoạn thiết kế. Giai đoạn tiếp theo trong qui trình phát triển phần mềm là thiết kế và mã hóa. Do đó trong chương này, tôi sẽ giới thiệu một số khái niệm trong thiết kế và mã hóa như môđun, phân chia môđun, ghép nối, kết dính giữa các môđun. Phần còn lại sẽ giới thiệu các hoạt động của việc lập trình và tích hợp các môđun.

4.1 THIẾT KẾ

4.1.1 Tổng quan trong thiết kế

Phát triển phần mềm là một quá trình gồm nhiều giai đoạn, phân tích là giai đoạn xác định phần mềm sẽ thực hiện cái gì, thiết kế sẽ xác định cách thức thực hiện những gì đã được đặt ra ở phần phân tích. Thiết kế là giai đoạn rất quan trọng góp phần tạo nên sự thành công của phần mềm. Giai đoạn thực hiện là triển khai những gì đã được thiết kế. Nếu trong quá trình thực hiện có khó khăn thì phải quay lại sửa bản thiết kế. Bản thiết kế tốt sẽ là cơ sở để giúp cho việc quản lý và giảm chi phí cho công việc bảo trì phần mềm sau này.

Sau khi phân tích yêu cầu hệ thống, nhóm phát triển chuyển sang pha thiết kế và cài đặt hệ thống. Thiết kế là quyết định chọn lựa, xây dựng đặc tả về hành vi, bổ sung các chi tiết cần thiết cho việc cài đặt trên hệ thống máy tính bao gồm việc tổ chức quản lý dữ liệu, kiến trúc, thủ tục, tương tác với người dùng,… Bản thiết kế phải đủ chi tiết về mặt quản lý và kỹ thuật như thiết kế sơ bộ, thiết kế chi tiết, thiết kế dữ liệu, thiết kế kiến trúc, thiết kế thủ tục, thiết kế đối tượng, thiết kế giao diện. Từ bản thiết kế, người lập trình có thể dùng ngôn ngữ để phát triển thành chương trình sao cho phần mềm vận hành được, đáp ứng được yêu cầu đặt ra.

Trên thực tế phân tích và thiết kế không có sự tách biệt nhau mà hai hoạt động này được tiến hành song song và bổ sung cho nhau bởi vì phân tích là tìm ra những yêu cầu của khách hàng, thiết kế sẽ quyết định cách thức để thực hiện những yêu cầu đó, đặc tả một cách đầy đủ, chi tiết những yêu cầu đã được xác định trong phân tích.

Có 3 cấp độ thiết kế phần mềm:

Thiết kế kiến trúc (thiết kế sơ bộ) là bản thiết kế có tính trừu tượng cao nhất, hệ thống phần mềm được xác định gồm nhiều thành phần tương tác nhau. Người thiết kế phải đề xuất ý tưởng cho miền giải pháp.

Thiết kế cấp cao sẽ chia nhỏ các thực thể đơn ở phần thiết kế kiến trúc thành nhiều hệ thống con và các mô đun có mức độ trừu tượng thấp hơn, đồng thời mô tả sự tương tác giữa các thành phần này với nhau. Thiết kế cấp cao sẽ tập trung vào cách thức hệ thống và tất cả các thành phần của nó có thể thực hiện được trong

những dạng thức của mô đun. Cấp độ này thể hiện cấu trúc mô đun, mối quan hệ và sự tương tác giữa các mô đun với nhau.

Thiết kế chi tiết có liên quan đến sự thực thi những thành phần được nhìn thấy trong hai bản thiết kế trước. Chi tiết hơn cho những mô đun và sự thực thi của các mô đun đó. Nó xác định cấu trúc logic của mỗi mô đun và các giao diện để giao tiếp giữa các mô đun đó.

☙ ***Phân chia mô đun***

Mô đun là tập hợp của một hay nhiều câu lệnh kế tiếp nhau nhằm thực hiện một tác vụ nào đó và được đặt tên. Các phần khác nhau trong chương trình có thể kích hoạt mô đun bằng tên của nó. Mỗi mô đun có tập hợp các tên biến riêng biệt. Mô đun là một khối đơn các mã lệnh có thể kích hoạt giống như thủ tục, hàm hay phương thức.

Khái niệm mô đun đã xuất hiện khoảng 4 thập niên trở lại đây. Mô đun hóa là một kỹ thuật chia hệ thống phần mềm thành nhiều mô đun rời rạc và độc lập nhau. Các mô đun sẽ có khả năng thực hiện công việc một cách độc lập. Những mô đun mới này có thể làm việc như những cấu trúc cơ bản cho toàn bộ phần mềm. Người thiết kế muốn thiết kế các mô đun như thế để chúng có thể thực thi, biên dịch một cách riêng biệt và độc lập được.

Việc phân chia mô đun trong thiết kế vô tình đi theo nguyên tắc "chia để trị" - một chiến lược giải quyết các bài toán phức tạp (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009).

Giả sử C(x) là độ phức tạp của đoạn mã x, E(x) là công sức để thực hiện x.

Nếu C(p1) > C(p2) thì E(p1) > E(p2).

Nếu phân chia p = p1 + p2 ta thấy (một cách trực quan):

C(p1 + p2) > C(p1) + C(p2) -> E(p1 + p2) > E(p1) + E(p2)

Như vậy: Việc phân chia mô đun thành những mô đun nhỏ hơn thì công sức để thực hiện các mô đun giảm xuống và như thế độ phức tạp cũng giảm theo nhưng công sức tích hợp các mô đun có chiều hướng tăng lên.

▶ Cách thức phân chia mô đun đã mang đến nhiều thuận lợi trong việc thiết kế phần mềm như:

- Hệ thống được chia thành các thành phần nhỏ hơn, điều này sẽ gia tăng tính cục bộ giữa các mô đun, giúp cho việc sửa đổi, bảo trì được dễ dàng.

- Chương trình có thể được phân chia thành các mô đun theo chức năng của hệ thống.

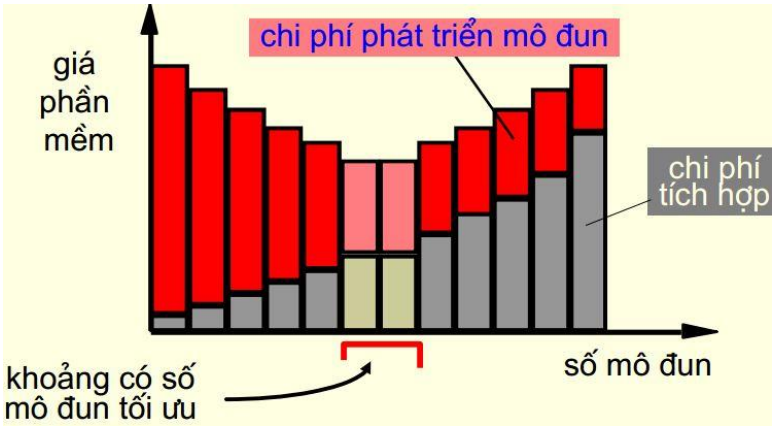
- Mức độ trừu trượng mong muốn cũng được đưa vào chương trình.

- Các thành phần có độ gắn kết cao có thể tái sử dụng lại.

- Có khả năng phát triển song song.

- Giảm độ phức tạp.

Phần mềm được xây dựng bằng cách phân chia thành nhiều mô đun để cho việc quản lý phần mềm khoa học hơn. Về sau các mô đun này phải được tích hợp lại với nhau. Khi đó, nhà phát triển cần chi phí tích hợp và giải quyết những khó khăn trong quá trình tích hợp mô đun. Vì vậy cần xác định số mô đun tối ưu trong quá trình phân chia (số lượng mô đun không quá nhiều, không quá ít). Hình dưới đây cho thấy những thông tin cần xem xét và cân nhắc trong việc phân chia mô đun (Nguyễn Văn Vỵ & Nguyễn Việt Hà, 2009).



Hình 38. Các yếu tố ảnh hưởng đến phân chia mô đun

☙ ***Chất lượng thiết kế***

Chất lượng thiết kế có thể được xem xét qua các đặc trưng sau:

- Thiết kế phải triển khai được tất cả các yêu cầu trong mô hình phân tích và yêu cầu tiềm ẩn mà khách hàng đòi hỏi.
- Thiết kế phải là bản hướng dẫn dễ đọc, dễ hiểu cho người viết chương trình, người kiểm thử và người bảo trì.
- Thiết kế cần cung cấp một bức tranh đầy đủ về tổng thể phần mềm trên quan điểm triển khai hướng đến các mặt dữ liệu, chức năng và hành vi của hệ thống.

Ngoài ra, chất lượng của việc phân chia mô đun cũng cần phải được xem xét, đo lường. Như chúng ta đã biết, các tác vụ của hệ thống được phân chia thành nhiều mô đun dựa trên các thuộc tính của chúng. Các mô đun được xem như những thực thể độc lập nhưng có thể tương tác với nhau trong hoạt động xử lý. Do đó, việc ghép nối (coupling) và độ kết dính (cohesion) giữa các mô đun được dùng để đánh giá tính độc lập và sự tương tác của chúng. Kết dính chặt và ghép nối lỏng giữa các mô đun là tiêu chuẩn của một thiết kế tốt.

❖ Ghép nối (Coupling):

- Độ đo sự liên kết (trao đổi dữ liệu) giữa các mô đun

- Ghép nối chặt chẽ thì khó hiểu, khó sửa đổi do phải tính đến các liên kết có thể, dễ gây lỗi lan truyền.

❖ Kết dính (Cohesion)

- Độ đo sự phụ thuộc lẫn nhau của các thành phần trong một mô đun
- Kết dính cao thì tính cục bộ cao (độc lập chức năng); dễ hiểu, dễ sửa đổi

☙ ***Che giấu thông tin***

Che dấu thông tin là một trong những nguyên lý quan trọng của việc phân chia mô đun. Các mô đun giao tiếp với nhau bằng những thông tin thật sự cần thiết. Những thông tin về thủ tục và dữ liệu cục bộ của mỗi mô đun phải được che dấu khỏi các mô đun khác. Một trong những lợi ích của việc che giấu thông tin là kiểm soát được thay đổi và sửa lỗi dễ dàng.

Khi xây dựng một hệ thống phần mềm, người sử dụng không quan tâm đến cấu trúc bên trong của hệ thống là đơn giản hay phức tạp, không cần biết cách thức cài đặt của nó thông qua thuật toán gì, cấu trúc dữ liệu như thế nào, có tài nguyên hệ thống ra sao. Họ chỉ quan tâm đến giao diện có những tham số và giá trị nhập vào, trả lại như thế nào. Sử dụng mô đun thông qua các giao diện là cách thức che giấu thông tin. Việc che giấu thông tin có thể mang lại nhiều lợi ích như:

- Giảm hiệu ứng phụ khi sửa đổi thông tin
- Giảm tác động của thiết kế tổng thể lên thiết kế cục bộ
- Nhấn mạnh trao đổi thông tin thông qua giao diện
- Loại bỏ việc sử dụng dữ liệu dùng chung
- Hướng tới sự đóng gói chức năng – một thuộc tính của thiết kế tốt.

☙ ***Trừu tượng và tinh chế***

❖ **Trừu tượng**

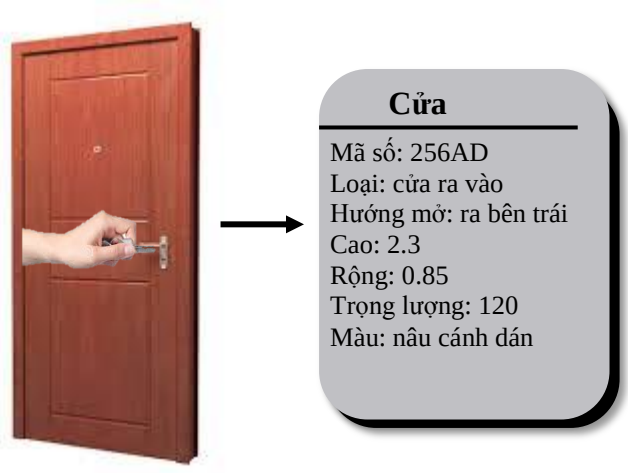
Trừ tượng hóa là quá trình ánh xạ một sự vật/hiện tượng của thế giới thực thành một khái niệm lôgic. Quá trình thiết kế trải qua nhiều mức trừu tượng hoá khác nhau. Mức trừu tượng cao nhất là các vấn đề được mô tả một cách tổng quát và có sử dụng thuật ngữ hướng vấn đề. Ở các mức thấp hơn thì hướng đến thủ tục xử lý chi tiết; kết hợp các thuật ngữ hướng đến hiện thực. Mức thấp nhất thì mô tả vấn đề theo cách có thể thực hiện trực tiếp.

Có 2 loại trừu tượng hoá: Trừu tượng hóa thủ tục và trừu tượng hóa dữ liệu.

Trừu tượng hoá thủ tục là chuỗi các lệnh liên tiếp thực hiện chức năng nào đó. Ví dụ: Thủ tục mở cửa (bao gồm đi đến cửa, cầm lấy tay nắm, xoay tay nắm, kéo cánh cửa, đi vào…); hay thêm một phần tử vào danh sách có thứ tự (xác định vị trí cần thêm, chèn phần tử mới vào vị trí vừa xác định).

Hình 39. Trừu tượng thủ tục

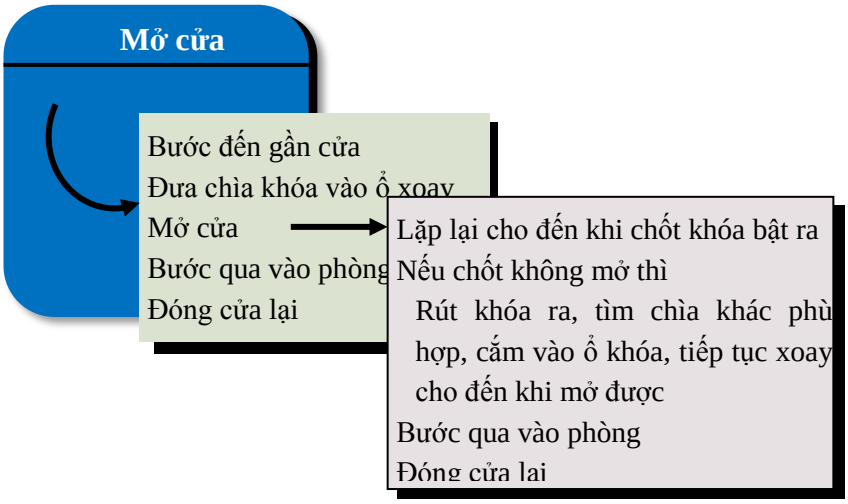
Trừu tượng hoá dữ liệu là tổ hợp dữ liệu mô tả một đối tượng dữ liệu (có thể liên hệ tới đối tượng thực thể trong UML). Ví dụ: hàng, chồng, cánh cửa...



Hình 40. Trừu tượng dữ liệu

❖ **Tinh chế**

Tinh chế là quá trình làm rõ vấn đề. Tinh chế và trừu tượng hoá là hai khái niệm bù trừ nhau: càng tinh chế thì càng hạ thấp mức trừu tượng hoá. Trong thiết kế phần mềm người ta thực hiện trừu tượng hóa trước, sau đó tiến hành tinh chế hoá nhằm tạo ra bản thiết kế hoàn chỉnh và tiến hóa trong thiết kế.

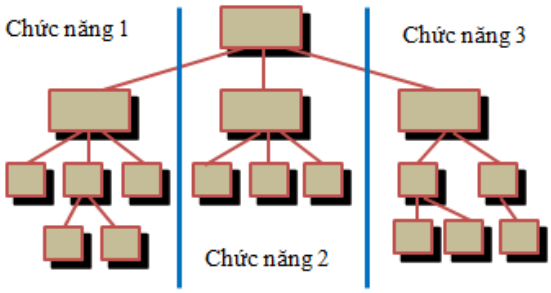


Hình 41. Làm mịn từng bước

4.1.2 Thiết kế kiến trúc

Thiết kế kiến trúc thường sử dụng biểu đồ cấu trúc để mô tả về hệ thống một cách tổng thể, thể hiện được mối quan hệ giữa các mô đun, giao diện giao tiếp giữa các mô đun. Thiết kế kiến trúc không cần chỉ ra cụ thể các đặc điểm như thứ tự thực hiện, số lần thực hiện hay chi tiết thiết kế.

Kiến trúc phần mềm mô tả các thành phần (component) kiến tạo nên hệ thống phần mềm và sự giao tiếp giữa các thành phần đó như các mô đun mã nguồn, các file thực thi (*.dll, *.exe, *.class...), các thành phần của kiến trúc hệ thống: ActiveX control, bean..., các trang HTML, *.asp, *.jsp.



Hình 42. Kiến trúc môđun theo chức năng

4.1.3 Thiết kế dữ liệu

Thiết kế dữ liệu là tìm kiếm biểu diễn luận lý cho các phần tử dữ liệu đã được nhận diện trong giai đoạn phân tích yêu cầu. Thiết kế các cấu trúc dữ liệu của chương trình và cơ sở dữ liệu. Sau đó thực hiện tinh chế từng bước sao cho không dư thừa dữ liệu, tối ưu hóa không gian lưu trữ, chuẩn hóa cơ sở dữ liệu bằng lược đồ ERD và dạng chuẩn 3.

Thiết kế cấu trúc dữ liệu là thiết kế cơ sở dữ liệu động trong hệ thống bởi vì cấu trúc dữ liệu mô tả sự tổ chức, phương thức truy xuất, mức độ liên kết và các xử lý khác của thông tin. Cơ sở dữ liệu có thể là dữ liệu đơn, có thể là dữ liệu dạng phức

tạp. Dữ liệu đơn là dạng cấu trúc dữ liệu đơn giản nhất chỉ bao gồm một phần tử thông tin mà có thể được truy xuất bằng một định danh. Một số dạng phức tạp hơn như vector, ma trận, mảng nhiều chiều, danh sách liên kết, hàng, chồng, cây nhị phân… Các dạng dữ liệu này được biểu diễn ở các mức trừu tượng hoá khác nhau. Việc chọn lựa cách biểu diễn các đối tượng thiết kế cũng có ảnh hưởng mạnh mẽ đến chất lượng phần mềm.

Có nhiều mức thiết kế dữ liệu, trong đó thiết kế cấu trúc logic sẽ liên quan đến các quan hệ chuẩn, các khóa, các tham chiếu, các cấu trúc thao tác dữ liệu. Thiết kế cấu trúc vật lý quan tâm đến các các file, các kiểu, kích cỡ dữ liệu.

4.1.4 Thiết kế thủ tục

Thiết kế thủ tục là thiết lập thuật giải cho các mô đun đã kiến tạo sao cho có thể dễ dàng mã hoá bằng ngôn ngữ lập trình có cấu trúc. Người ta có thể biểu diễn thuật giải bằng lưu đồ thuật giải như Flowchart, ngôn ngữ PDL.

Ngôn ngữ PDL vay mượn từ vựng của ngôn ngữ tự nhiên và cú pháp của ngôn ngữ lập trình có cấu trúc. Nó có các tính chất như cú pháp chặt chẽ của các từ khoá, hỗ trợ đặc tả cấu trúc, khai báo dữ liệu, phân chia module; cú pháp tự do của ngôn ngữ tự nhiên giúp miêu tả xử lý; phương tiện mô tả dữ liệu đơn cũng như dữ liệu tổ hợp; cơ chế định nghĩa chương trình con và phương cách gọi.

Ví dụ dùng ngôn ngữ PDL mô tả thủ tục xác định tam giác:

```
procedure AnalyzeTriangle( a, b, c: in real; type: out string)
begin
sort a, b, c so that a >= b >= c;
  if (c>0 and a < b + c )
                if ( a = c )
                type := “Equilateral”
        else
                if ( a = b or b = c )
                        type := “Isosceles”
                else
                        if ( a*a = b*b + c*c )
                                type := “Right”
                        else
                                type := “Scalene”
  else
                type := “Error”
end
```

4.1.5 Thiết kế giao diện

Chúng ta phải luôn nhớ một nguyên tắc quan trọng khi xây dựng một hệ thống phần mềm, đó là người sử dụng không quan tâm đến cấu trúc bên trong của hệ thống, đơn giản hay phức tạp; cái mà họ có thể đánh giá được và cảm nhận được chính là giao diện tương tác giữa hệ thống và người sử dụng. Nếu người sử dụng cảm thấy giao diện không thích hợp, khó sử dụng thì rất có thể họ sẽ không sử dụng cả hệ

thống; cho dù hệ thống đó có đáp ứng tất cả các chức năng nghiệp vụ mà họ muốn. Và như vậy, dự án của chúng ta sẽ thất bại.

Giao diện người dùng cần phải được thiết kế sao cho phù hợp với kỹ năng, kinh nghiệm và sự mong đợi của người sử dụng nó.

Người sử dụng thường đánh giá hệ thống thông qua giao diện hơn là chức năng của nó. Giao diện của hệ thống nghèo nàn có thể khiến người sử dụng tạo ra các lỗi hết sức nghiêm trọng. Đó là lý do tại sao nhiều hệ thống phần mềm không bao giờ được sử dụng.

Tác nhân con người trong thiết kế giao diện

Một nhân tố quan trọng ảnh hưởng tới quá trình thiết kế giao diện đó chính là người sử dụng hệ thống. Do đó, chúng ta phải tìm hiểu một số đặc điểm của người sử dụng có liên quan đến giao diện hệ thống:

- *Khả năng nhớ tức thời của con người bị hạn chế*: Con người chỉ có thể nhớ ngay khoảng 7 loại thông tin. Nếu ta biểu diễn nhiều hơn 7 loại, thì có thể khiến người sử dụng không nhớ hết và gây ra các lỗi.

- *Người sử dụng có thể gây ra lỗi*: Khi người sử dụng gây ra lỗi khiến hệ thống sẽ hoạt động sai, những thông báo không thích hợp có thể làm tăng áp lực lên người sử dụng và do đó, càng xảy ra nhiều lỗi hơn.

- *Người sử dụng là khác nhau*: Con người có những khả năng khác nhau. Những người thiết kế không nên chỉ thiết kế giao diện phù hợp với những khả năng của chính họ.

- *Người sử dụng thích các loại tương tác khác nhau*: Một số người thích hình ảnh, văn bản, âm thanh …

Các nguyên tắc thiết kế giao diện

Không phải tất cả các nguyên tắc thiết kế giao diện đều có thể được áp dụng cho tất cả các giao diện. Sau đây là các nguyên tắc thiết kế giao diện:

- *Sự quen thuộc của người sử dụng*: giao diện phải được xây dựng dựa trên các thuật ngữ và các khái niệm mà người sử dụng có thể hiểu được hơn là những khái niệm liên quan đến máy tính. Ví dụ: hệ thống văn phòng nên sử dụng các khái niệm như thư, tài liệu, cặp giấy … mà không nên sử dụng những khái niệm như thư mục, danh mục …

- *Thống nhất*: hệ thống nên hiển thị ở mức thống nhất thích hợp. Ví dụ: các câu lệnh và menu nên có cùng định dạng …

- *Tối thiểu hóa sự bất ngờ*: Nếu một yêu cầu được xử lý theo cách đã biết trước thì người sử dụng có thể dự đoán các thao tác của những yêu cầu tương tư.

- *Khả năng phục hồi*: Hệ thống nên cung cấp một số khả năng phục hồi từ lỗi của người sử dụng và cho phép người sử dụng khôi phục lại từ chỗ bị lỗi. Khả năng này bao gồm cho phép làm lại, hỏi lại những hành động như xóa, huỷ …

- *Hướng dẫn người sử dụng* như hệ thống trợ giúp, hướng dẫn trực tuyến …

- *Tính đa dạng*: Hỗ trợ nhiều loại tương tác cho nhiều loại người sử dụng khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị.

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hóa dữ liệu. Trực quan hóa có thể phát hiện ra mối quan hệ giữa các thực thể và các xu hướng trong dữ liệu. Ví dụ: thông tin về thời tiết được hiển thị dưới dạng biểu đồ, trạng thái của mạng điện thoại nên được hiển thị bởi các nút có liên kết với nhau.

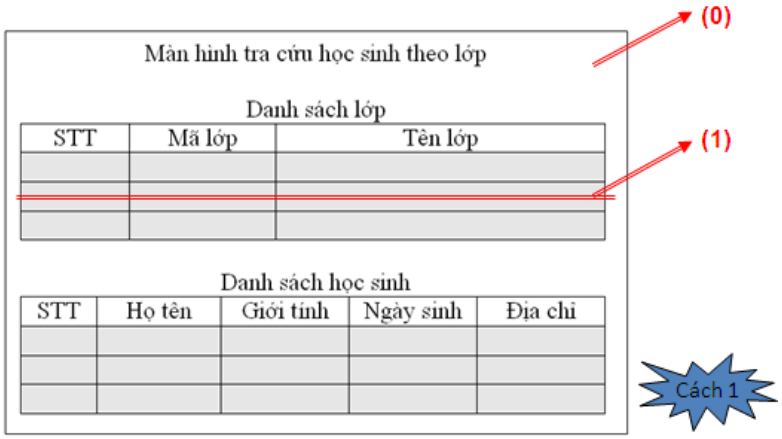
Chúng ta thường sử dụng màu trong khi thiết kế giao diện. Màu bổ sung thêm một chiều nữa cho giao diện và giúp cho người sử dụng hiểu được những cấu trúc thông tin phức tạp. Màu sắc có thể được sử dụng để đánh dấu những sự kiện ngoại lệ.

Tuy nhiên, khi sử dụng màu để thiết kế giao diện có thể gây phản tác dụng. Do đó, chúng ta nên quan tâm tới một số hướng dẫn sau:

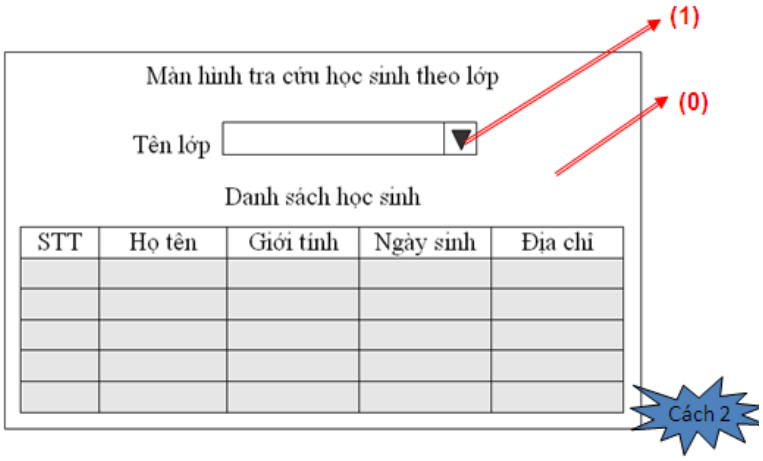
- Giới hạn số lượng màu được sử dụng và không nên lạm dụng việc sử dụng màu.
- Thay đổi màu khi thay đổi trạng thái của hệ thống
- Sử dụng màu để hỗ trợ cho những nhiệm vụ mà người sử dụng đang cố gắng thực hiện.
- Sử dụng màu một cách thống nhất và cẩn thận với các cặp màu đối kháng nhau..

Khi người sử dụng tương tác với hệ thống, rất có thể xảy ra lỗi và hệ thống phải thông báo cho người sử dụng biết lỗi gì đã xảy ra hoặc đã có chuyện gì xảy ra với hệ thống. Do đó, thiết kế thông báo lỗi là vô cùng quan trọng. Nếu thông báo lỗi nghèo nàn có thể làm cho người sử dụng từ chối hơn là chấp nhận hệ thống. Vì vậy, thông báo lỗi nên ngắn gọn, súc tích, thống nhất và có cấu trúc. Việc thiết kế thông báo lỗi nên dựa vào kỹ năng và kinh nghiệm của người sử dụng.

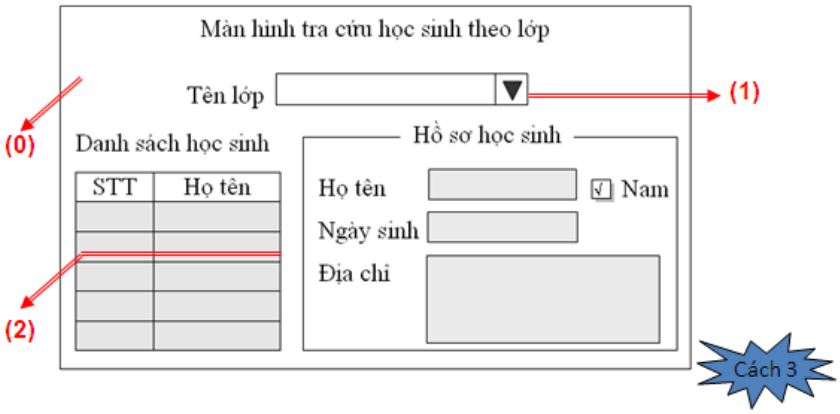
Ví dụ một số giao diện:



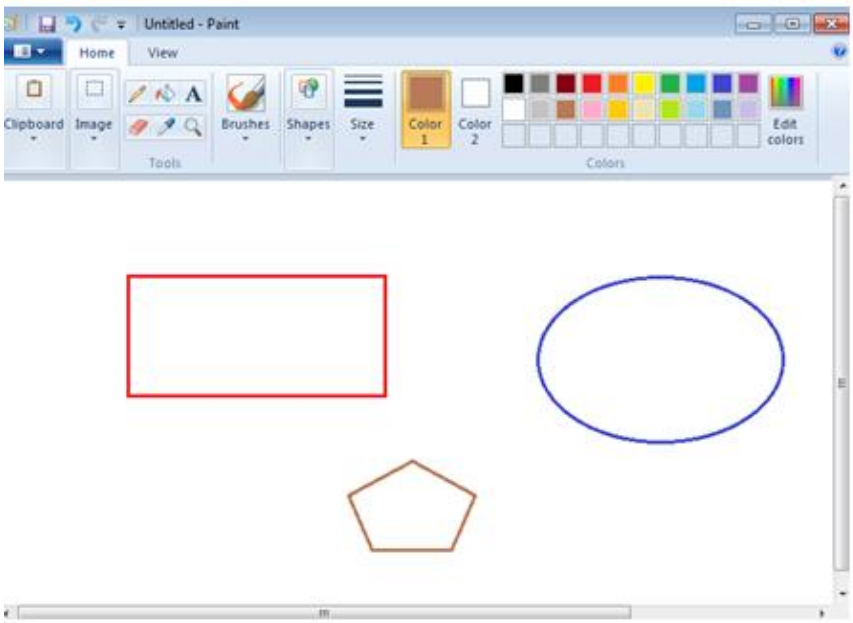
Hình 43. Giao diện 1



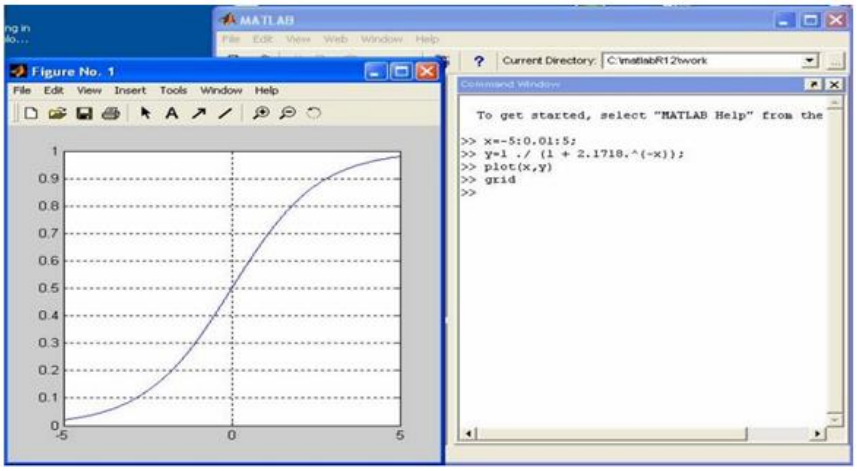
Hình 44. Giao diện 2



Hình 45. Giao diện 3

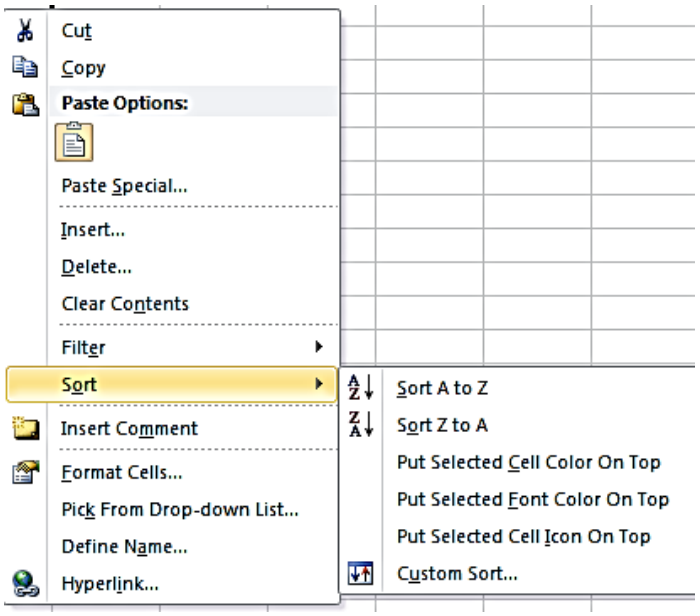


Hình 46. Giao diện tương tác trực tiếp



Hình 47. Giao diện dạng thực đơn

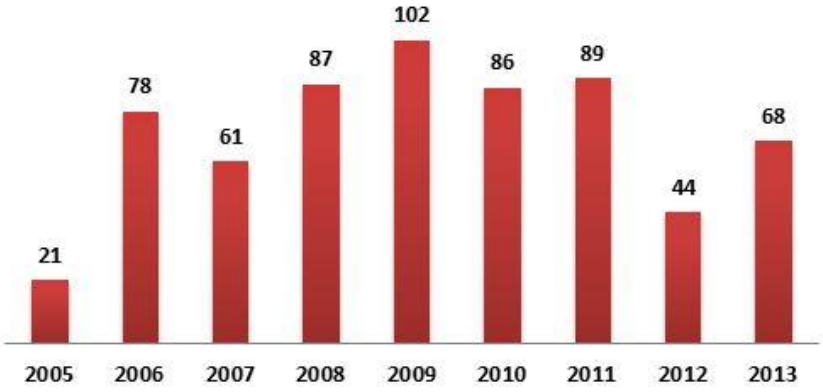
- Không cần nhớ lệnh
- Tối thiểu hóa dùng bàn phím
- Tránh các lỗi như sai lệnh, sai tham số
- Dễ dàng tạo các trợ giúp ngữ cảnh



Hình 48. Giao diện dạng menu

Cách hiển thị thông tin

- Hiển thị văn bản (text): Chính xác, dễ cài đặt
- Hiển thị đồ họa (graphic): Trực quan, dễ nhận dạng, thấy được tổng thể và mối quan hệ



Hình 49. Dạng biểu đồ

Thời gian phản hồi

- Thời gian trung bình
  - Thời gian trung bình phản hồi với thao tác
  - Người dùng không thể đợi quá lâu (<3s)
  - Cần chứng tỏ hệ thống đang hoạt động
- Độ biến thiên thời gian
  - Chênh lệch không được lớn
  - Đều đặn là tốt

Cách đưa thông báo

- Phải có phản hồi của hệ thống đối với các thao tác
- Thông báo phải có nghĩa, dễ hiểu, các thông tin là hữu ích
  - Tránh đưa các số hiệu
  - Định dạng thông báo phải nhất quán (vị trí, nội dung)
- Thông báo lỗi
  - Chính xác
  - Có tính xây dựng (chỉ ra nguyên nhân lỗi, gợi ý cách khắc phục lỗi,…)

Gợi ý thiết kế giao diện

- Nên đồng nhất (menu, lệnh, hiển thị...)
- Nên cung cấp feedback cho người dùng
- Yêu cầu xác nhận những tác vụ mang tính phá hoại (xoá file, account)
- Nên hỗ trợ UNDO, REDO
- Hạn chế lượng thông tin phải ghi nhớ giữa 2 tác vụ liên tiếp
- Tối ưu trong trình bày hộp thoại và di chuyển chuột
- Chấp nhận lỗi từ phía người sử dụng
- Cung cấp trợ giúp trực tuyến
- Dùng động từ đơn giản và ngắn gọn để đặt tên các lệnh

4.1.6 Design pattern (tạm dịch mẫu thiết kế)

Trong phát triển phần mềm hiện đại, kiến trúc tổng thể của dự án đóng một vai trò quan trọng, đặc biệt với framework và design pattern. Design pattern cung cấp giải pháp ở dạng tổng quát, giúp tăng tốc độ phát triển phần mềm bằng cách đưa ra các mô hình test, mô hình phát triển đã qua kiểm nghiệm. Sử dụng design pattern mang lại rất nhiều lợi ích như tránh các vấn đề tiềm ẩn có thể gây ra những lỗi lớn, dễ dàng nâng cấp và bảo trì hệ thống về sau. Một lợi thế lớn nữa là các lập trình viên khác có thể dễ dàng nhận ra các design pattern này. Khi sử dụng design pattern, chương trình thực sự trở nên đơn giản, giảm được thời gian và công sức suy nghĩ ra cách giải quyết cho những vấn đề đã có lời giải sẵn bởi các design pattern.

Pattern là gì?

Pattern mô tả một giải pháp chung đối với một vấn đề nào đó trong thiết kế thường được “lặp lại” trong nhiều dự án. Nói cách khác, một pattern có thể được xem như một “khuôn mẫu” có sẵn áp dụng được cho nhiều tình huống khác nhau để giải quyết một vấn đề cụ thể. Trong bất kỳ hệ thống phần mềm hướng đối tượng nào chúng ta cũng có thể bắt gặp các vấn đề lặp lại đó.

Tiềm năng ứng dụng của pattern là rất lớn. Các thiết kế dựa trên pattern được sử dụng khá nhiều ở các phần mềm mã nguồn mở, trong nền tảng J2EE hoặc .NET,.. Trong các dạng ứng dụng này, có thể dễ dàng nhận ra một số tên lớp chứa các tiền tố hoặc hậu tố như Factory, Proxy, Adapter,…

Pattern – tái sử dụng ý tưởng

Pattern cho phép các nhà thiết kế có thể cùng ngồi lại với nhau và cùng giải quyết một vấn đề nào đó mà không phải mất nhiều thời gian tranh cãi. Trong rất nhiều trường hợp, dự án phần mềm thất bại là do các nhà phát triển không có được sự hiểu biết chung trong các vấn đề về kiến trúc phần mềm. Ngoài ra, pattern cũng cung cấp các thuật ngữ chung trong thiết kế. Nói một cách đơn giản, khi đề cập đến một pattern nào đấy, bất kỳ ai biết pattern đó đều có thể nhanh chóng hình dung ra "bức tranh" của giải pháp. Nếu áp dụng pattern hiệu quả thì việc bảo trì phần mềm cũng được tiến hành thuận lợi hơn, nắm bắt kiến trúc hệ thống nhanh hơn.

Pattern hỗ trợ tái sử dụng kiến trúc và mô hình thiết kế phần mềm theo quy mô lớn

Cần có sự phân biệt design pattern với framework. Framework hỗ trợ tái sử dụng mô hình thiết kế và mã nguồn ở mức chi tiết hơn. Trong khi đó, design pattern được vận dụng ở mức tổng quát hơn, giúp các nhà phát triển hình dung và ghi nhận các cấu trúc tĩnh và động cũng như quan hệ tương tác giữa các giải pháp trong quá trình thiết kế ứng dụng.

Pattern đa tương thích

Pattern không phụ thuộc vào ngôn ngữ lập trình, công nghệ hoặc các nền tảng lớn như J2EE của Sun hay Microsoft.NET Framework.

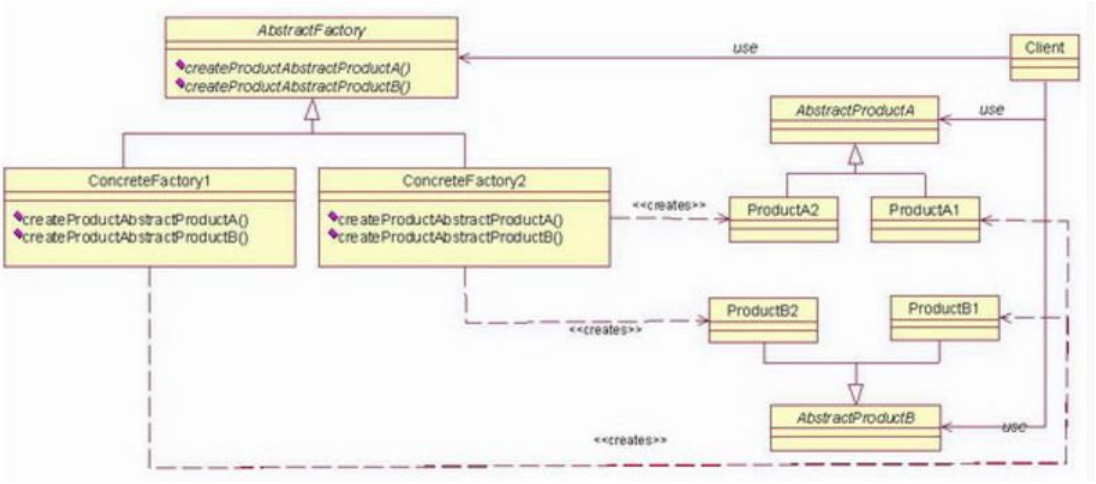
Pattern được chia ra làm 3 nhóm chính sau:

Nhóm kiến tạo (Creational Pattern): gồm Factory, Abstract Factory, Singleton, Prototype, Builder,… liên quan đến quá trình khởi tạo đối tượng cụ thể từ một định nghĩa trừu tượng (abstract class, interface).

Nhóm cấu trúc tĩnh (Structural Pattern): gồm Proxy, Adapter, Wrapper, Bridge,… liên quan đến vấn đề làm thế nào để các lớp và đối tượng kết hợp với nhau tạo thành các cấu trúc lớn hơn.

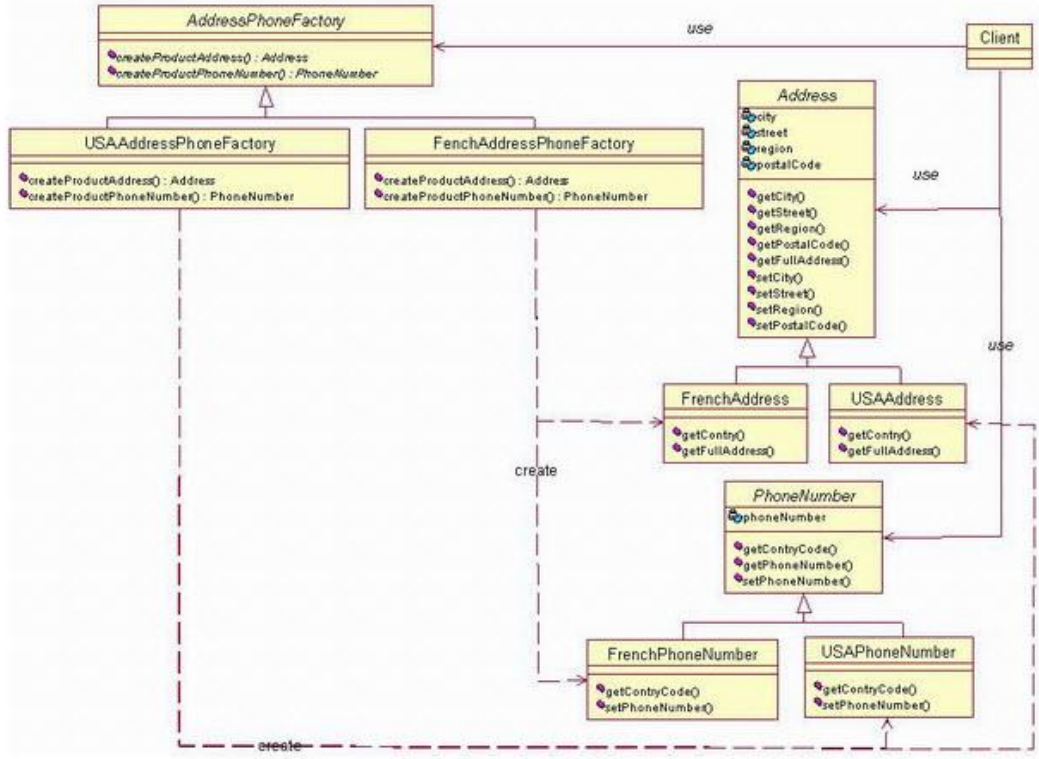
Nhóm tương tác (Behavioral Pattern): gồm Observe, State, Command,… mô tả cách thức để các lớp hoặc đối tượng có thể giao tiếp với nhau.

Ví dụ: Abstract Factory Method Pattern có ý nghĩa là đóng gói một nhóm những lớp đóng vai trò "sản xuất" (Factory) trong ứng dụng, đây là những lớp được dùng để tạo lập các đối tượng. Các lớp sản xuất này có chung một giao diện lập trình được kế thừa từ một lớp cha thuần ảo gọi là "lớp sản xuất ảo".



Hình 50. Cấu trúc mẫu Abstract Factory Method Pattern

Giả sử ta cần viết một ứng dụng quản lý địa chỉ và số điện thoại cho các quốc gia trên thế giới. Địa chỉ và số điện thoại của mỗi quốc gia sẽ có một số điểm giống nhau và một số điển khác nhau. Sơ đồ lớp được xây dựng như sau:



Hình 51. Sơ đồ lớp quản lý địa chỉ và số điện thoại

4.2 THỰC HIỆN

4.2.1 Vai trò của lập trình

Lập trình được xem là một thành phần đơn giản nhưng mất nhiều thời gian, công sức và chi phí trong các dự án phát triển phần mềm. Giai đoạn lập kế hoạch, phân tích, thiết kế và quản trị giữ vai trò quyết định sự thành công hay thất bại của một dự án. Trong khi đó giai đoạn thực thi không ảnh hưởng lớn nhưng nó bị phụ

thuộc vào các giai đoạn trước đó. Bởi vì người lập trình không thể viết code tốt nếu các giai đoạn trước làm không tốt.

Sản phẩm phần mềm được xem là tốt khi các giai đoạn phân tích, thiết kế, lập trình tốt, kiểm thử chặt chẽ. Trong đó, kỹ thuật lập trình được xem là tốt thì trong lập trình phải đạt được tính chuyên nghiệp (tuân thủ các chuẩn), ổn định và hiệu quả. Khi lập trình đạt hiệu quả thì sản phẩm có giá thành thấp hơn, tốc độ phát triển cao hơn, dễ bảo trì, chất lượng cao.

Trong giai đoạn thực thi, lập trình viên có thể làm việc với nhiều loại ngôn ngữ lập trình. Mỗi loại ngôn ngữ lập trình có những đặc điểm và ưu thế riêng, sau đây là một số dạng ngôn ngữ lập trình:

- **Lập trình tuần tự** (tuyến tính): độ ghép nối cao, chương trình khó hiểu, khó sửa, dễ sinh lỗi (assembly, basic,…)
- **Lập trình có cấu trúc** (thủ tục): Dễ hiểu hơn, an toàn hơn (Fortran, Pascal, C,…)
- **Lập trình hướng chức năng**: Dựa trên nguyên tắc ghép nối dữ liệu, loại bỏ hiệu ứng phụ, nâng cao tính tái sử dụng (Lisp)
- **Lập trình hướng đối tượng**: Cục bộ hơn, dễ tái sử dụng hơn, thuận tiện cho các ứng dụng lớn (C++, Java, C#)
- Kỹ thuật thế hệ thứ 4.

Phong cách lập trình tốt được thể hiện ở nhiều yếu tố như cách đặt tên hàm và biến, cách xây dựng câu lệnh, cấu trúc chương trình, cách viết chú thích, mã nguồn dễ hiểu, dễ sửa đổi, an toàn, ít lỗi…Cụ thể như:

- Hiểu rõ ngôn ngữ
- Sử dụng tên biến thích hợp và có nghĩa
- Chú thích tự thân nghĩa là tên biến phải được diễn giải ngay từ đầu, về sau không cần chú thích thêm.
- Nên có các chú thích bên trong module
- Các vấn đề về sử dụng tham số
- Mã lệnh dễ đọc, sử dụng các cặp dấu ngoặc, canh đầu dòng
- Nên có các dòng trắng để phân biệt các công việc.
- Thông tin tối thiểu của một module:
    - Tên module
    - Mô tả vắn tắt các công việc module phải thực hiện
    - Tên của lập trình viên
    - Ngày viết module

- Ngày module được chấp thuận, chấp thuận bởi ai
- Các tham số
- Danh sách các tên biến (nên theo thứ tự chữ cái), cách sử dụng
- Tên các tập tin mà module có truy xuất
- Tên các tập tin bị thay đổi bởi modul
- Nhập xuất của module
- Các khả năng lỗi xảy ra
- Tên tập tin sẽ được sử dụng để kiểm thử
- Các lỗi đã biết
- Danh sách các cập nhật đã được thực hiện với ngày tương ứng, người thực hiện.

Chú thích

Mọi điều phải được chú thích trong chương trình như mục đích sử dụng của các biến; chức năng của khối lệnh, câu lệnh; chú thích các mô đun về mục đích, chức năng của mô đun, tham số giá trị trả lại, cấu trúc, thuật toán,..

Đặt tên

Đặt tên biến, tên hàm có nghĩa, gợi nhớ. Nên Sử dụng các ký hiệu, từ tiếng Anh có nghĩa, ví dụ: Dùng DateOfBirth hoặc date_of_birth, chứ không dùng dateofbirth; tránh đặt tên quá dài.

Câu lệnh

Các câu lệnh phải mô tả cấu trúc như thụt lề, dễ đọc, dễ hiểu, làm đơn giản các lệnh, mỗi lệnh trên một dòng, hạn chế truyền tham số là kết quả của hàm, biểu thức.

Tránh các cấu trúc phức tạp

Các lệnh if lồng nhau

Điều kiện phủ định if not

Ví dụ các lệnh if lồng nhau:

```
if (vido>=30 && kinhdo>120)
{
    if (vido=60 && kinhdo<=150)
        mapSquareNo = 1;
   else if (vido<=90 && kinhdo<=150)
        mapSquareNo = 2;
        else
        System.out.println
            (“Not on the map”);
}
else
    System.out.println(“Not on the map”);
```

Hình 52. Định dạng tốt

```
if (vido>=30 && kinhdo>120) { if (vido<=60 && kinhdo<=150)
mapSquareNo = 1;   else if (vido<=90 && kinhdo<=150) mapSquareNo = 2;    else
System.out.println(“Not on the map”);} else System.out.println(“Not on the map”);
```

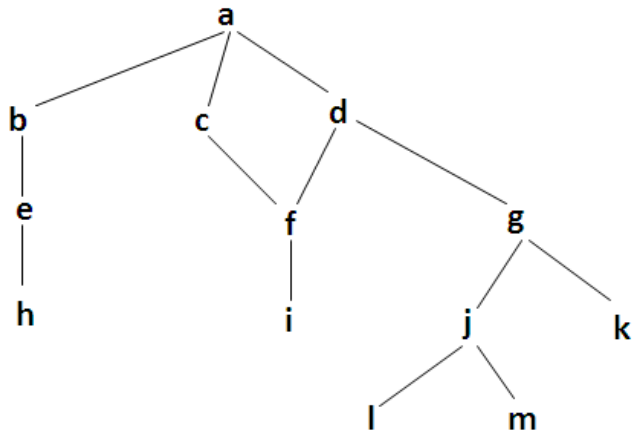
Hình 53. Định dạng không đạt

Viết mã lệnh chuẩn

- Thống nhất quy ước đặt tên module, tên biến,…
- Nên sử dụng các qui tắc sau:
  - Độ lồng nhau của lệnh if tối đa là 3
  - Mỗi module có khoảng 35 đến 50 mã lệnh thực thi
  - Không sử dụng lệnh goto, có thể sử dụng để bắt lỗi
- Chịu sự kiểm thử của nhóm SQA
- Có khả năng tái sử dụng

4.2.2 Các chiến lược cài đặt và tích hợp mô đun

- Chiến lược big bang là kiểu tích hợp cùng một lúc tất cả các mô đun và sau đó thực hiện kiểm thử chung. Cách này không mang lại hiệu quả cao, làm tăng sự phức tạp bởi vì việc phát hiện, tìm vết và sửa lỗi sau khi tích hợp là rất khó khăn. Cách này chỉ thích hợp với các chương trình nhỏ
- Chiến lược tích hợp trên-xuống là kiểu tích hợp bắt đầu từ các mô đun ở nút gốc trước, sau đó tích hợp dần xuống các nút lá dựa trên cây mô đun.



Hình 54. Các mô đun được phân chia như hình cây

Nếu mô đun a gọi mô đun b thì a phải được cài đặt và tích hợp trước b.

Theo cách này thì thứ tự của hình trên như sau: a, b, c, d, e, f, g, h, i, j, k,l ,m hay a, b, e, h, c, d, f, i, g,j,k,l

Nếu khi thêm một mô đun mới vào thì chỉ có thể xảy ra lỗi tại các vị trí từ mô đun mới hoặc giao diện giữa mô đun mới và phần còn lại.

❖ Chiến lược tích lợp từ dưới – lên

Các mô đun có thể chia thành 2 dạng: Mô đun logic là tổ hợp các dòng điều khiển quyết định trong sản phẩm như a, b, c, d và có thể j, g. Mô đun hoạt động là mô đun hoạt động thực sự của sản phẩm: e, f, h, i, k, l, m. Các mô đun hoạt động phải được cài đặt trước các mô đun logic.

Nếu mô đun a gọi mô đun b thì b được cài đặt trước a

Theo hình trên, thứ tự duy nhất là l, m, h, i, j, k, e, f, g, b, c, d, a

Nên phân chia các mô đun như sau:

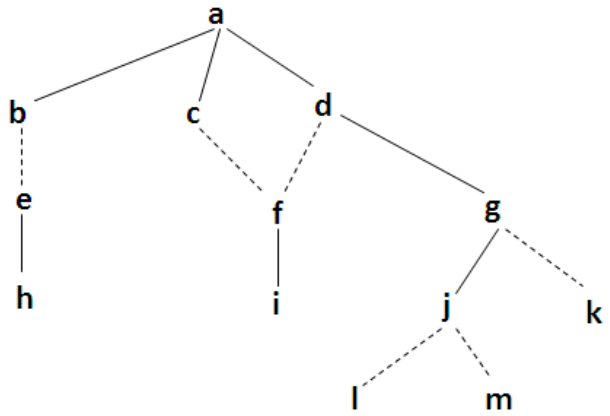
- Người 1: h, e, b
- Người 2: i, f, c
- Người 3: l, m, j, k, g, d

Tích hợp 3 và 2, sau khi tích hợp b, c, d thì cài đặt và tích hợp a

Nếu gặp lỗi ở các mô đun logic thì sẽ khó khăn khi lần vết sửa đổi trở lại trên các mô đun đã thực hiện cài đặt và tích hợp.

❖ Cài đặt và tích hợp tổng hợp

Cài đặt: Trên xuống: a, b, c, d, g, j. Dưới lên: e, f, h, i, k, l, m



Hình 55. Mô đun logic và mô đun hoạt động

Bảng 6. Thông tin so sánh giữa các cách tiếp cận

Cách tiếp cận	**Điểm mạnh**	**Điểm yếu**
Cài đặt trước, tích hợp sau		Không cô lập được lỗi, chậm phát hiện lỗi thiết kế chính
Cài đặt & tích hợp trên xuống	cô lập được lỗi, sớm phát hiện lỗi thiết kế chính	Các module sử dụng lại không được kiểm thử đầy đủ
Cài đặt & tích hợp dưới lên	cô lập được lỗi, các module sử dụng lại được kiểm thử đầy đủ	Chậm phát hiện lỗi thiết kế chính
Cài đặt & tích hợp hỗn hợp	cô lập được lỗi, sớm phát hiện lỗi thiết kế chính, Các module sử dụng lại được kiểm thử đầy đủ	

Câu hỏi cuối chương

1. Trình bày mối quan hệ giữa các giai đoạn phân tích – thiết kế - thực hiện - kiểm thử - bảo trì. Hãy liệt kê các công việc cần được thực hiện ở giai đoạn thiết kế phần mềm. Vì sao nói "Trên thực tế phân tích và thiết kế không có sự tách biệt nhau mà hai giai đoạn này được tiến hành song song và bổ sung cho nhau"?

2. Trình bày khái niệm mô đun. Thông qua hình 38, hãy phân tích và giải thích vì sao phải phân chia mô đun một cách tối ưu (số lượng mô đun nhiều quá hoặc ít quá thì không tốt, các yếu tố ảnh hưởng: giá PM, độ phức tạp, công sức thực hiện,..)?

3. Trình bày những thuận lợi của việc phân chia mô đun trong thiết kế phần mềm.

4. Khi thiết kế phần mềm, việc che giấu thông tin và phân chia mô đun có quan hệ như thế nào? Vì sao trong thiết kế phần mềm cần thực hiện việc che giấu thông tin? Hãy liệt kê các lợi ích của việc che giấu thông tin.

5. Công sức thực hiện, độ phức tạp của mô đun có quan hệ với nhau như thế nào? Phải phân chia mô đun như thế nào để đạt hiệu quả tốt nhất?

6. Trình bày đặc điểm của tính ghép nối và kết dính trong phân chia mô đun. Yếu tố ghép nối và kết dính giữa các mô đun có ý nghĩa gì trong thiết kế? Vì sao khi phân chia mô đun phải tăng độ kết dính và giảm sự liên kết?

7. Giao diện giữ vai trò như thế nào đối với người sử dụng? Hãy gợi ý một số hướng dẫn về các yếu tố màu sắc và những gợi ý khác trong việc thiết kế giao diện.

CHƯƠNG 5
KIỂM THỬ VÀ BẢO TRÌ PHẦN MỀM

Kiểm thử và bảo trì phần mềm là các giai đoạn sau của qui trình phát triển phần mềm nhằm đảm bảo sản phẩm phần mềm dùng được, vận hành tốt với chất lượng đáp ứng yêu cầu mong đợi. Nội dung chương này giới thiệu về một số khái niệm trong giai đoạn kiểm thử. Phần tiếp theo giới thiệu các phương pháp, chiến lược, kỹ thuật hoạt động trong kiểm thử phần mềm. Trong đó tôi chọn phương pháp kiểm thử hợp trắng với kỹ thuật đồ thị dòng chảy để minh họa cho một số ví dụ kiểm thử có sử dụng testcase. Phần cuối của chương giới thiệu khái quát về hoạt động bảo trì phần mềm.

5.1 KIỂM THỬ PHẦN MỀM

5.1.1 Giới thiệu

Nói đến công nghiệp phần mềm người ta nghĩ ngay đến đội ngũ lập trình viên, những người viết mã (code) chương trình. Tuy nhiên, trên thực tế phần mềm là một sản phẩm phức tạp phải được thực hiện qua nhiều công đoạn như đặc tả, phân tích, thiết kế, viết mã (code), kiểm thử, bảo trì,… Người ta thống kê rằng trong qui trình làm phần mềm, có khoảng 40 - 45% công sức dành cho kiểm thử (Trần Cao Đệ & Đỗ Thanh Nghị, 2012). Công việc chính của đội ngũ kiểm thử viên là kiểm thử (test) một cách có hệ thống để tìm lỗi (defect) của chương trình, tức là lỗi trong mã lệnh được viết ra bởi lập trình viên. Nhiều thống kê cũng cho thấy rằng trên thế giới hiện nay tỷ lệ lập trình viên trên kiểm thử viên là 3:1, tức là 3 lập trình viên có 1 kiểm thử viên.

Hoạt động kiểm thử được thực hiện để chỉ ra rằng một chương trình thực hiện được những yêu cầu đặt ra ban đầu và tìm ra những khuyết điểm của chương trình trước khi đưa vào sử dụng. Khi kiểm thử phần mềm, những bộ dữ liệu kiểm tra phải được tạo ra dùng làm dữ liệu đầu vào của chương trình, để thực thi chương trình, kiểm tra kết quả thực thi, kiểm tra lỗi, phát hiện những bất thường hoặc có thêm thông tin về các thuộc tính phi chức năng của chương trình.

Theo IEEE 829, mục đích của kiểm thử là nhằm phát hiện lỗi phần mềm, để chứng tỏ phần mềm thực hiện đúng các chức năng mong đợi và nhằm xác lập độ tin cậy của cái mà chương trình muốn thực hiện. Một kiểm thử tốt phải kiểm tra được những hành vi bất thường của chương trình. Kiểm thử chỉ có thể chỉ ra một chương trình có lỗi nhưng không thể chứng tỏ là chương trình không có lỗi. Tuy nhiên, qua kiểm thử người ta sẽ thu được thông tin về tình trạng lỗi của phần mềm, tức là thu được hình ảnh về chất lượng của sản phẩm được kiểm thử.

Có thể nói rằng lỗi phần mềm là một phần hiển nhiên của quá trình phát triển, dù cố gắng đến mức nào thì ngay cả lập trình viên xuất sắc nhất cũng không thể đảm

bảo lúc nào cũng viết được những đoạn mã không có lỗi. Trung bình, một lập trình viên loại tốt nhất thì cũng có từ 1 đến 3 lỗi trên 100 dòng lệnh. Ngày nay, các phần mềm càng ngày càng trở nên phức tạp và đồ sộ. Nó có thể chứa đến hàng triệu dòng mã lệnh do hàng trăm hàng ngàn thậm chí hàng chục ngàn nhân viên, kỹ sư phát triển. Một sản phẩm tích hợp từ nhiều mô đun, sản phẩm khác, thư viện lập trình của nhiều tổ chức, công ty phát triển phần mềm khác nhau. Vì vậy việc kiểm thử tính đúng đắn của phần mềm càng ngày càng trở nên rất quan trọng và rất phức tạp.

Kiểm thử phần mềm là công cụ đảm bảo chất lượng phần mềm đầu tiên được áp dụng để kiểm soát chất lượng sản phẩm phần mềm trước khi giao hoặc cài đặt sản phẩm cho khách hàng (Trần Cao Đệ & Nguyễn Công Danh, 2014). Trước đây, kiểm thử chỉ được thực hiện ở giai đoạn cuối cùng, tức là sau khi toàn bộ gói phần mềm đã được hoàn thành. Nhưng sau đó, khi tầm quan trọng của việc phát hiện sớm các lỗi phần mềm xâm nhập vào khái niệm đảm bảo chất lượng, các chuyên gia đảm bảo chất lượng phần mềm (SQA) đã được khuyến khích để mở rộng kiểm thử cho các sản phẩm vừa được cài đặt xong, điều này dẫn đến sự hình thành của kiểm thử đơn vị và kiểm thử tích hợp.

Vì sao phải kiểm thử phần mềm?

Qui trình phát triển phần mềm có nhiều giai đoạn được thực hiện tự động hoá bởi các công cụ CASE. Có những giai đoạn được thực hiện bởi con người. Lỗi có thể xảy ra trong tất cả các giai đoạn: phân tích yêu cầu, thiết kế, mã hoá,… Do đó, phải kiểm thử chương trình trước khi chính thức sử dụng. Việc kiểm thử phải được thực hiện một cách có kế hoạch và có phương pháp để tìm ra lỗi với chi phí thấp mang lại hiệu suất tìm lỗi cao.

Ví dụ về lỗi trong chương trình qui đổi điểm số 0..10 thành điểm chữ có đặc tả như sau:

Mỗi môn học có hai điểm số thành phần là điểm giữa kỳ và cuối kỳ

Mỗi điểm số thành phần có một hệ số (%) nhất định, trong đó điểm cuối kỳ phải có hệ số từ 50% trở lên

Điểm trung bình là trung bình theo trọng số của các điểm số thành phần, được làm tròn đến 1 chữ số theo qui tắc làm tròn

Điểm trung bình được qui đổi sang điểm chữ theo qui tắc:

Điểm số từ 0 đến cận 4: điểm F

Điểm số từ 4 đến cận 5.5: điểm D

Điểm số từ 7.0 đến cận 8.5: điểm B

Điểm số từ 8.5 đến cận 10: điểm A

Người lập trình viết chương trình như sau:

Char doi(float diemtb){//diemtb đã được tính toán và làm tròn

if(diemtb >=0.0 && diemtb<=4.0) return 'F';

if(diemtb >=4.0 && diemtb<5.5) return 'D';

...

}

Ở đây người lập trình đã có sai sót trong khi viết biểu thức logic đầu tiên, lẽ ra nó phải là *if(diemtb >=0.0 && diemtb<4.0).* Sai sót này đã để lại một lỗi trong chương trình. Có thể trong một thời gian dài không ai có điểm trung bình là 4.0 nên lỗi sẽ không gây ra tác dụng hay thiệt hại gì… cho đến khi có ai đó có diemtb là 4.0, bị qui đổi ra điểm chữ là 'F' oan phải đi khiếu nại.

5.1.2 Các khái niệm

Xác minh và thẩm định

Quá trình kiểm thử có 2 mục tiêu chính:

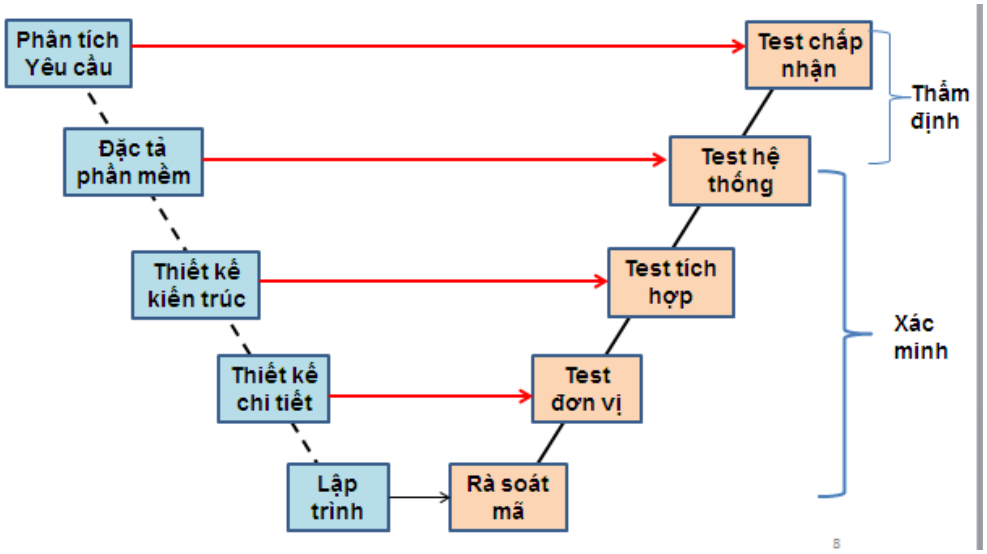
1. Chứng minh cho người phát triển và khách hàng thấy các yêu cầu của phần mềm.

2. Phát hiện các lỗi và khiếm khuyết trong phần mềm: xử lý của phần mềm không chính xác, không mong muốn hoặc không thích hợp với đặc tả kỹ thuật của nó - nói chung là lỗi của phần mềm. Kiểm tra lỗi có liên quan đến hành vi của hệ thống không mong muốn như hệ thống bị treo, những tương tác không mong muốn với hệ thống bên ngoài, tính toán không chính xác hoặc hư hỏng dữ liệu.

Mục tiêu đầu tiên là kiểm thử thẩm định (validation), kiểm tra xem phần mềm có đáp ứng yêu cầu người dùng không, hệ thống thực hiện đúng với mong đợi của người dùng không.

Mục tiêu thứ hai là kiểm tra khuyết điểm. Tuy nhiên ranh giới giữa hai hướng tiếp cận kiểm thử này chỉ mang tính tương đối. Bởi vì trong suốt quá trình kiểm thử thẩm định, kiểm thử viên có thể sẽ tìm thấy những lỗi trong hệ thống và trong suốt quá trình kiểm tra khuyết điểm, một số kiểm thử thấy được chương trình có đáp ứng được các yêu cầu đặt ra ban đầu.

Kiểm thử không thể chứng minh được phần mềm không còn lỗi, cũng không chứng minh được rằng hệ thống sẽ xử lý luôn như thế trong mọi tình huống. Kiểm thử chỉ có thể cho thấy sự hiện diện của lỗi, chứ không chỉ ra sự vắng mặt của lỗi.



Hình 56. Mô hình kiểm thử chữ V

Xác minh và thẩm định liên quan đến việc kiểm tra phần mềm có được phát triển như đặc tả của nó và cung cấp những chức năng mong đợi đến cho khách hàng không. Hai hoạt động này cốt yếu để đảm bảo chất lượng phần mềm và được bắt đầu ngay từ những yêu cầu ban đầu và tiếp tục thực hiện xuyên suốt quá trình phát triển phần mềm.

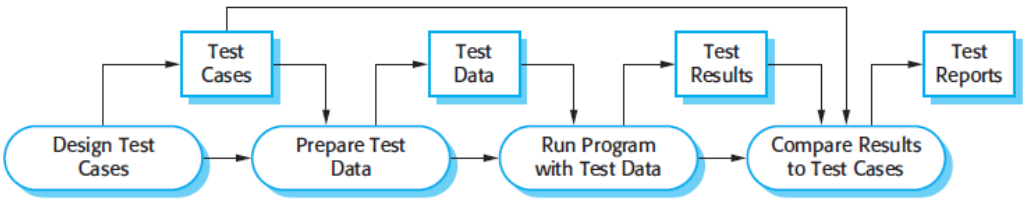
Mục đích của xác minh là kiểm tra phần mềm có đáp ứng các yêu cầu chức năng và phi chức năng của nó không. Thẩm định là quá trình tổng quát hơn. Mục đích của thẩm định là đảm bảo phần mềm đáp ứng mong đợi của khách hàng. Nó vượt xa kiểu kiểm tra thực thi một cách đơn giản bằng sự đặc tả kỹ thuật để chứng minh rằng phần mềm làm được những gì mà khách hàng mong đợi. Thẩm định là chính yếu bởi vì những đặc tả yêu cầu không luôn luôn phản ánh những mong ước, yêu cầu thực sự của khách hàng và người sử dụng.

Cơ sở của hoạt động xác minh và thẩm định bao gồm bản đặc tả yêu cầu, mã nguồn. Riêng hoạt động xác minh cần thêm bản thiết kế. Hai hoạt động chính của thẩm định và xác minh là: rà soát và kiểm thử. Rà soát là xem xét, đánh giá sản phẩm được tiến hành mỗi giai đoạn để phát hiện ra những khiếm khuyết cần sửa trước khi sang giai đoạn sau.

☙ *Khái niệm kiểm thử*

Theo Daniel Galin định nghĩa kiểm thử phần mềm như sau:

Kiểm thử phần mềm là một qui trình hình thức được thực hiện bởi nhóm kiểm thử chuyên nghiệp, trong đó một đơn vị phần mềm, một tích hợp của nhiều đơn vị phần mềm, hoặc toàn bộ gói phần mềm được kiểm tra bằng cách chạy các chương trình trên máy tính. Tất cả các kiểm thử được thực hiện dựa theo các thủ tục kiểm thử đã được phê duyệt cho các ca kiểm thử đã được phê duyệt.



Hình 57. Qui trình kiểm thử (**Iam Sommerville, trang 210, 2011)**

Hình trên là mô hình trừu tượng về qui trình kiểm thử truyền thống. Những ca kiểm thử (test cases) là những đặc tả đầu vào (input) để kiểm thử và đầu ra (output) mong đợi – kết quả kiểm thử. Dữ liệu kiểm thử là dữ liệu đầu vào được tạo ra để kiểm thử hệ thống. Đôi khi dữ liệu này có thể được tạo ra một cách tự động nhưng việc tạo ra ca kiểm thử tự động thì không thể được, chỉ có thể tự động ở việc thực thi kiểm thử. Kết quả mong đợi được so sánh với kết quả dự đoán để giúp kiểm thử viên tìm ra lỗi và những bất thường trong chương trình.

Trên thực tế qui trình kiểm thử có sự kết hợp giữa kiểm thử thủ công và tự động. Khi kiểm thử thủ công, kiểm thử viên chạy chương trình với dữ liệu kiểm thử và so sánh với kết quả mong đợi. Họ ghi nhận lại và báo cáo những khác biệt so với người phát triển chương trình. Khi kiểm thử tự động, việc kiểm thử được mã hóa thành chương trình và chạy kiểm thử mỗi khi cần kiểm thử hệ thống. Cách này nhanh hơn kiểm thử thủ công đặc biệt là khi kiểm thử hồi quy – nghĩa là chạy lại kiểm thử trước đó để kiểm tra sự thay đổi chương trình mà việc kiểm tra đó chưa được thông báo về các lỗi mới.

Việc dùng cách kiểm thử tự động trong thời gian qua được gia tăng đáng kể. Tuy nhiên, việc kiểm thử không bao giờ có thể làm tự động một cách hoàn toàn được, còn tùy thuộc vào các vấn đề được xem xét. Các trường hợp kiểm thử tự động mang lại hiệu quả cao như đồ họa, giao diện hay các chương trình không quan tâm đến hiệu ứng phụ.

5.1.3 Phương pháp kiểm thử

Kiểm thử phần mềm có thể dựa trên hai phương pháp kiểm thử phổ biến: Kiểm thử hộp trắng (white box) và kiểm thử hộp đen (black box).

❖ ***Kiểm thử hộp trắng*** (kiểm thử cấu trúc)(white-box): Kiểm tra các đường tính toán bên trong để xác định lỗi. Có khi phương pháp này được gọi tên là kiểm thử hộp thủy tinh để diễn tả đặc tính cơ bản của nó là đi điều tra tính đúng đắn của cấu trúc mã lệnh

❖ ***Kiểm thử hộp đen*** (kiểm thử chức năng) (**black-box**): Dùng để xác định các lỗi (bugs) bằng cách chỉ dựa vào "sự làm việc sai chức năng" của phần mềm khi chúng bị lộ ra tại các đầu ra bị sai của nó. Trong trường hợp các đầu ra được nhận thấy là đúng, kiểm thử hộp đen không quan tâm đến các đường tính toán và xử lý bên trong đã được thực hiện.

Mỗi phương pháp kiểm thử đều có khả năng tìm ra những nhóm lỗi khác nhau. Trong nhiều trường hợp, cả hai phương pháp trên có thể áp dụng kết hợp, nhưng trong vài trường hợp chỉ sử dụng một trong hai phương pháp kiểm thử là đủ. Do phải cân nhắc về chi phí, hầu hết các kiểm thử được thực hiện ngày nay là kiểm thử hộp đen vì nó tương đối ít tốn kém.

☙ ***Kiểm thử hộp trắng***

Kiểm nghiệm phần mềm cần xác định các đường thực thi độc lập cơ bản vì: Tất cả các đường thực thi độc lập được thử qua ít nhất một lần.

Qui trình kiểm nghiệm white-box nhằm bảo đảm số phép thử là ít nhất đủ để phát hiện các lỗi. Kiểm thử theo cách này phải thực hiện những công việc như xây dựng đồ thị dòng chảy, xác định những đường thực thi độc lập cơ bản, chuẩn bị những bộ testcase tương ứng với các đường đó, thực thi từng bộ dữ liệu một và so sánh kết quả.

Đối với kiểm thử hộp trắng, yêu cầu đặt ra như sau:

- Mọi con đường độc lập trong một mô đun cần được thực hiện ít nhất 1 lần
- Mọi ràng buộc lôgic được thực hiện cả phía đúng (true) & phía sai (false)
- Tất cả các vòng lặp ở biên và cả các biên vận hành phải được thực hiện
- Mọi cấu trúc dữ liệu nội tại được dùng để bảo đảm hiệu lực thi hành của nó.

Các bước kiểm thử hộp trắng

1. Chuẩn bị test case:

- Xây dựng đồ thị dòng chảy
- Xác định các đường thực thi độc lập cơ bản
- Xây dựng các test case tương ứng với đường thực thi độc lập cơ bản

2. Vận hành kiểm thử
3. So sánh kết quả
4. Sửa lỗi.

☙ ***Đồ thị dòng chảy***

❖ **Các đường thực thi độc lập cơ bản**:

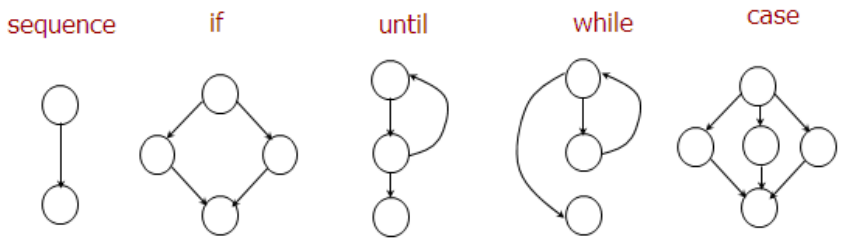
Từ node bắt đầu đến node kết thúc, các đường thực thi cơ bản được liệt kê theo một thứ tự nào đó để đảm bảo rằng: đường đang liệt kê ít nhất đi qua một cạnh chưa được duyệt qua bởi các đường đã liệt kê trước đó.

Tổng số đường thực thi cơ bản độc lập nhau được tính bằng công thức:

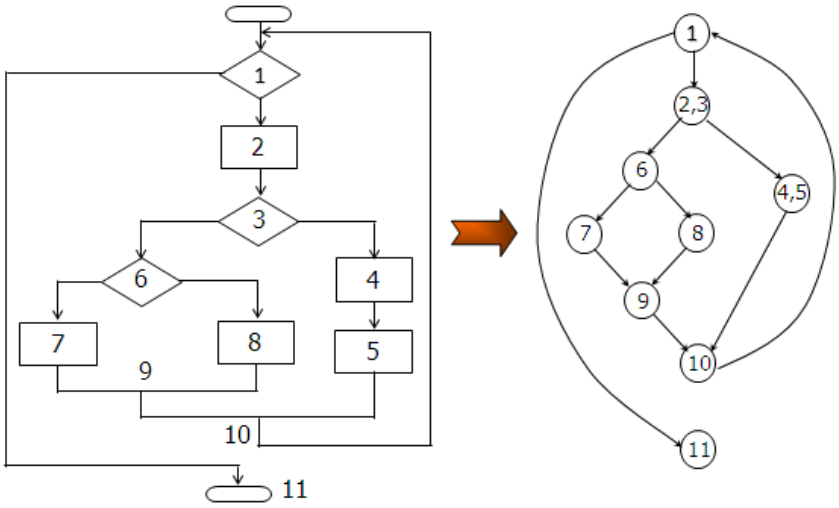
V = P + 1; trong đó P là số node phân nhánh (predicate)

❖ **Đồ thị dòng chảy**

- Mỗi node hình tròn biểu diễn một hoặc một vài tác vụ (hơi khác so với lưu đồ thuật giải)
- Cạnh có hướng miêu tả đường thực thi
- Đồ thị dòng chảy được xây dựng từ lưu đồ thuật giải

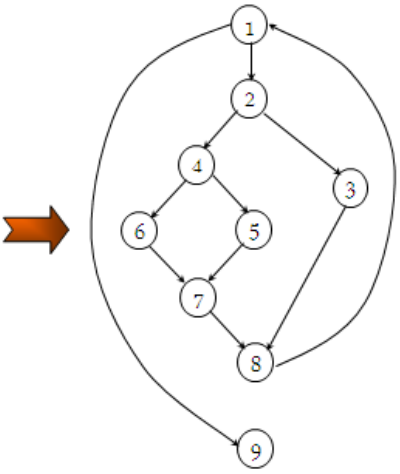


Hình 58. Đồ thị dòng chảy theo cấu trúc lệnh cơ bản



Hình 59. Chuyển từ lưu đồ sang đồ thị dòng chảy

```
procedure: DoSomething
1:      do while x=0
2:      if y=0 then
3:        z=0;
4:      elseif k=0 then
5:        z=1;
6:      else x=1;
7:      endif;
          endif;
8:    enddo
9: end
```



Hình 60. Chuyển từ các dòng lệnh sang đồ thị dòng chảy

Ví dụ:

Tổng số đường :

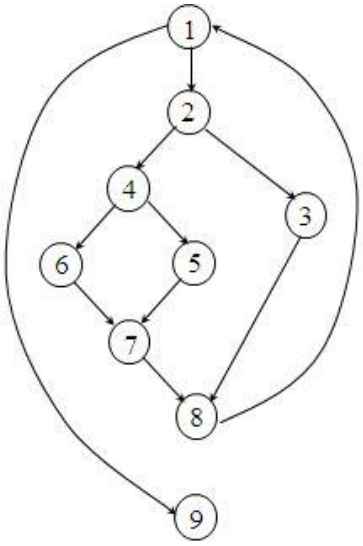
V = 3 + 1 = 4

Đường 1: 1-9

Đường 2: 1-2-3-8-1…

Đường 3: 1-2-4-5-7-8-1…

Đường 4: 1-2-4-6-7-8-1…



Hình 61. Ví dụ đồ thị

Chú ý: dấu 3 chấm (…) mang ý nghĩa "không quan tâm", từ đó có thể đi theo bất kỳ cạnh nào bởi vì các cạnh sau đó đã được duyệt qua rồi

Ví dụ: Đối với chương trình con AnalyzeTriangle

Tổng số đường :

V = 6 + 1 = 7

Đường 1: 1-3-12

Đường 2: 1-2-3-12

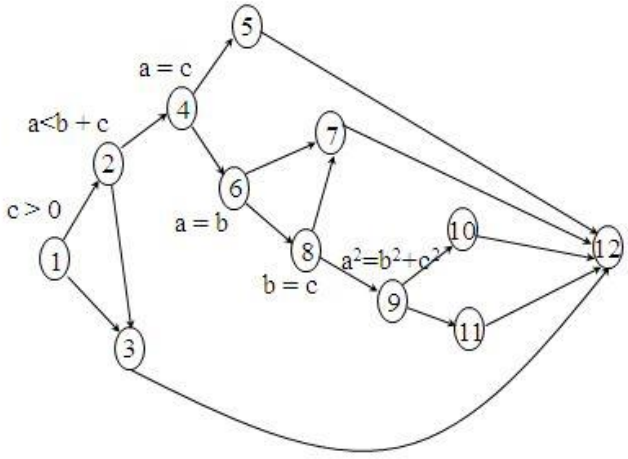
Đường 3: 1-2-4-5-12

Đường 4: 1-2-4-6-7-12

Đường 5: 1-2-4-6-8-7-12

Đường 6: 1-2-4-6-8-9-10-12

Đường 7: 1-2-4-6-8-9-11-12



Hình 62. Ví dụ đồ thị dòng chảy xác định tam giác

Thiết lập một test-case cho mỗi đường thực thi cơ bản

Dựa vào thuật giải để tìm ra một dữ liệu input, sau đó tính ra dữ liệu output hay đáp ứng mong đợi của thuật giải

Chú ý: có thể không tạo ra được test-case cho một đường thực thi nào đó

Ví dụ Sinh test-case cho chương trình con AnalyzeTriangle

Test-case cho đường 1:

Input: a = 3, b = 2, c = 0

Output mong đợi: type = "Error"

Test-case cho đường 2:

Input: a = 17, b = 5, c = 4

Output mong đợi: type = “Error”

Test-case cho đường 3:

Input: a = 6, b = 6, c = 6

Output mong đợi: type = “Equilateral”

☙ ***Nguyên lý kiểm thử***

- Việc kiểm nghiệm nên hướng về yêu cầu của khách hàng
- Nên được hoạch định trước một thời gian dài.
- Áp dụng nguyên lý Pareto (nguyên tắc 80-20):
    - 80% lỗi có nguyên nhân từ 20% các module
    - cô lập và kiểm tra những module khả nghi nhất.
- Nên tiến hành từ nhỏ đến lớn: bắt đầu từ những module riêng biệt rồi sau đó tích hợp các module lại.
- Không thể kiểm nghiệm triệt để một phần mềm.
- Nên được thực hiện bởi những đối tượng **KHÔNG** tham gia vào quá trình phát triển phần mềm.

☙ ***Các chiến lược kiểm thử***

Kiểm thử bigbang

Kiểm thử bigbang là tích hợp đồng thời một lúc tất cả các môđun lại với nhau. Cách kiểm thử này chỉ thích hợp với các chương trình nhỏ, rất phức tạp và không hiệu quả.

Kiểm thử hồi quy

Việc kết hợp các môđun lại với nhau có thể ảnh hưởng đến vòng lặp điều khiển, cấu trúc dữ liệu hay I/O chia sẻ trong một số môđun. Điều đó làm lộ ra một số lỗi không thể phát hiện được khi tiến hành kiểm nghiệm theo đơn vị

→ Phải kiểm thử hồi quy khi tích hợp

Kiểm nghiệm hồi quy có thể được tiến hành thủ công bằng cách thực hiện lại các test-case đã tạo ra. Hoặc có thể dùng một công cụ capture-playback để thực hiện tự động.

Kiểm thử tính năng

- Kiểm thử tính năng hiểu theo cách đơn giản nhất là kiểm tra các chức năng của phần mềm đáp ứng được nhu cầu của khách hàng đã được xác định trong văn bản đặc tả yêu cầu của phần mềm.
- Áp dụng kỹ thuật black-box
- Kiểm thử tính năng bao gồm:

- Xem xét lại cấu hình phần mềm theo lược đồ triển khai
- Kiểm thử alpha
- Kiểm thử beta

Kiểm thử alpha

Được tiến hành ngay tại nơi sản xuất phần mềm, trong môi trường được điều khiển. Nhà phát triển phần mềm sẽ quan sát người sử dụng dùng sản phẩm và ghi nhận lại những lỗi phát sinh để sửa chữa. Dữ liệu thường dùng trong cách này là dữ liệu mô phỏng.

Kiểm thử beta

Phần mềm được kiểm tra bên ngoài phạm vi của đơn vị sản xuất, trong môi trường thực. Khách hàng trực tiếp sử dụng và ghi nhận các vấn đề họ gặp phải trong quá trình sử dụng kiểm thử và báo cáo định kỳ cho nhà phát triển sửa chữa.

☙ ***Nghệ thuật gỡ rối – Debug***

Gỡ rối là một quá trình nhằm loại bỏ các lỗi được phát hiện trong quá trình kiểm tra. Gỡ rối được thực hiện như là một kết quả của việc kiểm tra: lỗi phát hiện được → tìm kiếm nguyên nhân → sửa lỗi.

Có 3 hình thức gỡ rối: brute force, loại trừ nguyên nhân và theo vết. Nên dùng kết hợp cả 3 hình thức này.

Gỡ rối là công việc khó khăn và dễ gây tâm lý chán nản bởi nguyên nhân gây ra lỗi nhiều khi lại mơ hồ: do timeout, do độ chính xác, do chủ quan lập trình... Khả năng gỡ rối gần như là bẩm sinh của mỗi người.

Tìm lỗi – Brute Force

Là phương pháp phổ biến nhất nhưng lại ít hiệu quả nhất cho việc phát hiện nguyên nhân gây lỗi phần mềm. Triết lý của phương pháp này là: "Hãy để máy tính tìm ra lỗi". Có 3 cách thực hiện:

- Lấy dữ liệu trong bộ nhớ để xem xét.
- Dùng run-time trace để tìm lỗi.
- Dùng lệnh WRITE để xuất dữ liệu cần kiểm tra ra màn hình.

❖ Áp dụng phương pháp này khi tất cả các phương pháp khác đều thất bại.

Loại trừ nguyên nhân

Phương pháp này dựa trên nguyên tắc phân chia nhị phân. Cách thực hiện:

- Khi một lỗi được phát hiện, cố gắng đưa ra một danh sách các nguyên nhân có thể gây ra lỗi.
- Danh sách này được nghiệm lại để loại bỏ dần các nguyên nhân không đúng cho đến khi tìm thấy một nguyên nhân khả nghi nhất.

- Khi đó dữ liệu kiểm nghiệm sẽ được tinh chế lại để tiếp tục tìm lỗi.

Theo vết

Theo vết là một phương pháp gỡ lỗi khá phổ biến có thể dùng thành công trong các chương trình nhỏ nhưng khó áp dụng cho đối với các chương trình rất lớn.

Cách thực hiện: bắt đầu tại dòng mã nguồn có triệu chứng lỗi thực hiện lần ngược trở lại từng dòng mã nguồn cho đến khi tìm thấy dòng gây ra lỗi.

5.1.4 Công cụ kiểm thử

☙ ***Kiểm thử đơn vị với JUnit***

Junit (Trần Cao Đệ & Đỗ Thanh Nghị, 2012) là thư viện phần mềm tự do được dùng cho kiểm thử tự động chức năng của chương trình theo đơn vị, viết bằng ngôn ngữ lập trình Java. JUnit giúp giảm bớt áp lực lo lắng và giảm tải công việc kiểm thử chương trình của kiểm thử viên. Công cụ nhằm phát hiện lỗi trong chương trình, từ đó nâng cao năng suất và đảm bảo chất lượng phần mềm.

☙ ***Kiểm thử chức năng với Quicktest Pro***

Quicktest Pro là công cụ kiểm thử tự động chức năng chương trình ứng dụng chuẩn Windows, ứng dụng Web, ActiveX hay Visual Basic. Quicktest Pro là sản phẩm của hãng Mercury, một sản phẩm khá tốt và mạnh, bao gồm nhiều chức năng điển hình của một cộng cụ kiểm thử dùng để kiểm tra chức năng tự động và cho phép kiểm tra hồi qui một cách tự động. Công cụ này cho phép kiểm tra viên bổ sung các kiểm thử bằng cách tạo tập tin mô tả cho nó mà không cần phải chỉnh sửa hay bổ sung bất cứ kịch bản nào. Nó có thể phù hợp trong tình huống chuyển giao công việc mà người mới tiếp nhận chưa có thời gian hoặc không hiểu kịch bản vẫn có thể thực hiện kiểm tra chức năng phần mềm theo đúng yêu cầu.

Các bước kiểm thử chức năng với Quicktest Pro bao gồm: thu thập các kiểm thử, sinh ra các bó kiểm thử, thêm vào các kiểm thử cần thực hiện, tạo dữ liệu cho kiểm thử và cuối cùng là thực thi kiểm thử.

5.2 BẢO TRÌ PHẦN MỀM

5.2.1 Giới thiệu bảo trì phần mềm

Có những hệ thống cần hoạt động liên tục và không được phép dừng nếu bắt gặp một lỗi nào đó. Ví dụ phần mềm điều khiển các chuyến bay hay hệ thống tính hiệu tàu hỏa không được phép dừng nếu có một lỗi xảy ra, hoặc chính phủ điều chỉnh các qui định buộc các phần mềm liên quan phải được điều chỉnh theo,... Lỗi không mong đợi của phần mềm có thể đe dọa đến đời sống và có thể gây ra những hậu quả rất xấu, nhất là hiện nay máy tính và phần mềm được ứng dụng trong rất nhiều lĩnh vực. Vì vậy, các hoạt động bảo trì được thực hiện nhằm giúp cho hệ thống sẵn sàng hoạt động. Chúng bao gồm sửa lỗi, phục hồi lại hệ thống không hoạt động hoặc cung

cấp các thay đổi phần cứng và phần mềm. Khi đó đòi hỏi người bảo trì phải nhìn lại sản phẩm từ phân tích, thiết kế,… cho đến những ràng buộc trước đây của sản phẩm.

Theo định nghĩa ở chương 1, phần mềm bao gồm không chỉ có mã nguồn (tập các câu lệnh) mà còn bao gồm các tài liệu như tài liệu đặc tả, thiết kế, kiểm thử, hướng dẫn người dùng,…Đây chính là các cơ sở để người làm bảo trì có thể nhìn lại sản phẩm trong quá trình bảo trì.

Theo định nghĩa của ISO/IEC, **bảo trì phần mềm** là sản phẩm phần mềm phải trải qua sự sửa đổi về mã lệnh và các tài liệu liên quan đến một vấn đề hay nhu cầu cải tiến. Mục đích là sửa đổi sản phẩm phần mềm hiện tại mà vẫn giữ được tính toàn vẹn của nó (Huỳnh Xuân Hiệp & Phan Phương Lan, 2014).

Bảo trì phần mềm phải được thực hiện nhằm mục đích:

- Khắc phục lỗi.
- Cải thiện thiết kế.
- Thực hiện các cải tiến.
- Giao tiếp 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 suất tới mức không thể chấp nhận được.

5.2.2 Phân loại bảo trì

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

- 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 đó 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.

5.2.3 Các vấn đề trong bảo trì

Bốn vấn đề then chốt trong bảo trì phần mềm cần quan tâm là các vấn đề về kỹ thuật, các vấn đề trong quản lý, các phép đo và sự ước lượng chi phí bảo trì. Hiểu được các vấn đề về kỹ thuật và quản lý, tổ chức bảo trì sẽ có sự chuẩn bị đầy đủ để thực hiện và hoàn thành công việc một cách hiệu quả, đáp ứng được yêu cầu thay đổi, đúng thời gian và trong phạm vi ngân sách. Ngoài ra, nhờ các phép đo, tổ chức bảo trì có thể đánh giá những phương pháp và công cụ nhằm chọn ra cái phù hợp nhất. Đồng thời, tổ chức bảo trì có thể thực hiện các dự báo về những khía cạnh khác nhau của phần mềm và quy trình, kiểm soát được quy trình, cải thiện về chất lượng và hiệu quả công việc.

Các kỹ thuật bảo trì phần mềm bao gồm hiểu chương trình, tái cấu trúc, kỹ thuật đảo ngược, di cư và nghĩ hưu. Trong đó việc đọc và hiểu chương trình là để thực hiện thành công các yêu cầu thay đổi. Điều này đòi hỏi bảo trì viên phải có kiến thức về những điểm đặc trưng của một phần mềm như phạm vi vấn đề, ảnh hưởng của sự thực hiện, quan hệ nhân – quả, quan hệ sản phẩm – môi trường, và các điểm đặc trưng hỗ trợ việc ra quyết định. Bảo trì viên cũng phải quan tâm đến những yếu tố ảnh hưởng đến việc có được các loại kiến thức trên như các chiến lược hiểu chương trình, quyết định nghiệp vụ, chất lượng của các tài liệu, sự tổ chức và trình bày, các vấn đề thực thi – thực tiễn lập trình, và các công cụ hỗ trợ.

Các công cụ hỗ trợ cho kỹ thuật đảo ngược và hiểu chương trình như bộ cắt lát chương trình (program slicer), bộ phân tích tĩnh (static analyser), bộ phân tích động (dynamic analyser), bộ tham khảo chéo (cross referencer), bộ phân tích dòng dữ liệu (data flow analyser), công cụ chuyển đổi (transformation tool),…

Ví dụ, Radare2 và ODA là các công dùng cho dịch ngược thuộc loại (disassembler); Java Decompile và .NET Reflector là các công cụ dịch ngược thuộc loại (decompiler).

Câu hỏi cuối chương

1. Vì sao phải kiểm thử phần mềm? Hãy trình bày hai mục tiêu của quá trình kiểm thử phần mềm. Xác minh và thẩm định là hai hoạt động để đảm bảo chất lượng phần mềm, vậy mục đích của hai hoạt động này là gì?

2. Vì sao kiểm nghiệm phần mềm cần xác định các đường thực thi độc lập cơ bản? Từ một mô đun chương trình làm thế nào để xác định được các đường độc lập cơ bản, cách thức thực hiện ra sao? Liệt kê qui trình kiểm nghiệm white box theo cách thức này (từ công việc bắt đầu đến công việc kết thúc kiểm nghiệm).

3. Vì sao nói kiểm nghiệm phần mềm không khẳng định được phần mềm không còn khiếm khuyết, chỉ khẳng định được phần mềm có lỗi và giảm thiểu lỗi? Cách thức kiểm nghiệm như thế nào để nhanh chóng tìm ra các lỗi, không dư thừa, không phức tạp? Ai là người kiểm nghiệm thích hợp? vì sao?

4. Vì sao phần mềm luôn luôn có lỗi trước kiểm nghiệm? Thế nào là kiểm nghiệm black-box và kiểm nghiệm white-box? Vì sao phải kết hợp các cách kiểm nghiệm phần mềm?

5. Qui trình kiểm nghiệm white-box là phải thực hiện những công việc gì? Xác định số đường thực thi độc lập cơ bản để làm gì? Liệt kê các chiến lược kiểm thử.

TÀI LIỆU THAM KHẢO

CHAOS Manifesto. (2013). *Think big, act small, The Standish Group International*.

Christopher Diggins. (k.n.). *Agile, Scrum, and Customer Oriented Development*. https://www.codeproject.com/Articles/573908/Agile-Scrum-and-Customer-Oriented-Development

Huỳnh Xuân Hiệp & Phan Phương Lan. (2014). *Giáo trình bảo trì phần mềm*. Cần Thơ: Nhà xuất bản Đại học Cần Thơ.

Iam Sommerville. (2011). *Software Engineering (9th)*. Addison Wesley

Lê Văn Tường Lân. (2009). *Giáo trình công nghệ phần mềm*. Đại học Sư phạm Huế.

Ngô Trung Việt & Nguyễn Kim Ánh. (2003). *Nhập môn kỹ nghệ phần mềm*. Hà nội: Nhà xuất bản Khoa học và kỹ thuật Hà Nội.

Nguyễn Văn Vỵ & Nguyễn Việt Hà. (2009). *Giáo trình kỹ nghệ phần mềm*. Hà nội: Nhà xuất bản Giáo dục Hà Nội.

Nguyễn Văn Hòa & Hồ Nhã Phong. (2016). *Giáo trình quản lý dự án phần mềm*. Cần Thơ: Nhà xuất bản Đại học Cần Thơ.

Trần Cao Đệ & Đỗ Thanh Nghị. (2012). *Giáo trình kiểm thử phần mềm*. Cần Thơ: Nhà xuất bản Đại học Cần Thơ.

Trần Cao Đệ & Nguyễn Công Danh.(2013). *Quản lý dự án phần mềm trong thực tiễn*. Cần Thơ: Nhà xuất bản Đại học Cần Thơ.

Trần Cao Đệ & Nguyễn Công Danh. (2014). *Giáo trình Đảm bảo chất lượng phần mềm*. Cần Thơ: Nhà xuất bản Đại học Cần Thơ.