MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ CÔNG NGHỆ PHẦN MỀM

MỤC ĐÍCH

- *Giới thiệu tổng quan về CNPM*
- *Nắm được một số khái niệm cơ bản*
- *Hiểu được các trách nhiệm đạo đức và nghề nghiệp của người kỹ sư CNPM.*
- *Hiểu được nhân tố con người và sự phân loại nghề nghiệp trong CNPM.*

1.1 Giới thiệu tổng quan về CNPM

1.1.1 Công nghệ phần mềm nhìn từ góc độ lịch sử

- Năm 1946 máy tính điện tử ra đời.
- Những năm 1950 máy tính được thương mại hóa: phần mềm bắt đầu được phát triển.

Máy tính điện tử đầu tiên cho mục đích thương mại là máy UNIVAC-1 được sản xuất năm 1951 ở Mỹ. Từ khoảng những năm 1955 thì bắt đầu có các phần mềm, tức là các chương trình máy tính.

- Những năm 1960: gặp những thất bại về phát triển phần mềm.

Cho đến nay, nếu lấy phương pháp và mức độ phức tạp làm căn cứ, thì có thể chia quá trình phát triển phần mềm thành 3 giai đoạn: 1955 - 1970, 1971 - 1985, 1986 đến nay. Tất nhiên sự phân chia này chỉ là tương đối mà thôi.

Đặc điểm của từng giai đoạn là:

- 1955 - 1970: Tính toán và quản lý rời rạc, quản lý nhỏ. Đặc tả yêu cầu của khách hàng lúc đó còn dùng ngôn ngữ tự nhiên thông thường.

Ví dụ: các câu kiểu như: tôi muốn có tệp dữ liệu chứa những thông tin về bán sản phẩm như số hóa đơn, người bán, mặt hàng,...; Dữ liệu sẽ được nhập từ bàn phím, có kiểm tra rồi mới đưa vào tệp; Hàng tháng tôi muốn lấy thông tin về doanh thu, lãi, số hàng bán....

Thiết kế thời ấy không được soạn thảo ra giấy mà chỉ hình thành trong tư duy của người lập trình. Người lập trình vừa viết chương trình vừa suy nghĩ về cách tổ chức dữ liệu, cách sử dụng các thuật toán sao cho chương trình chạy tốt, đáp ứng được yêu cầu của khách hàng.

Giai đoạn này các chương trình nhỏ, tính toán chuyên dụng (chỉ gồm khoảng trên dưới vài trăm dòng lệnh). Phương pháp lập trình được sử dụng thời đó là **lập trình tuyến tính**, tức là cách viết chương trình trong đó phần lớn các lệnh được đặt theo trình tự thực hiện của chúng, nghĩa là lệnh cần thực hiện trước sẽ được viết trước, lệnh thực hiện sau được viết sau.

Các hoạt động xử lý của chương trình là xử lí số, theo lô,... Ngôn ngữ được sử dụng trong giai đoạn này là ngôn ngữ mã máy, hợp ngữ và chúng mang tính đặc thù theo từng máy.

Các tiêu chí đánh giá gồm tính nhanh, giải được bài toán lớn (dùng bộ nhớ hiệu quả). Các công nghệ giai đoạn này như Bóng điện tử (các tính toán là chậm, bộ nhớ nhỏ), ...

- **1971 - 1985:** Lúc này đã có nhu cầu xây dựng các phần mềm thời gian thực (nghĩa là tính toán và sử dụng ngay kết quả, ví dụ tính toán trong lò phản ứng hạt nhân phải có ngay kết quả để điều khiển kịp thời).

Lúc này có nhu cầu xây dựng các mạng cục bộ và kết nối các cơ sở dữ liệu. Đặc tả thời đó chú trọng vào đặc tả đầu vào, đầu ra; dữ liệu và các luồng dữ liệu (data flow).

Ví dụ: những tệp dữ liệu lưu trữ những thông tin gì, trong quá trình xử lý thì dữ liệu được tính toán và di chuyển ra sao. Đầu ra của tệp dữ liệu này sau khi xử lý lại có thể trở thành đầu vào của tệp khác...

Thiết kế thời đó chú trọng tới cấu hình hệ thống (system configuration). Vấn đề cấu trúc đối với dữ liệu và giải thuật cũng được chú ý, vì vấn đề lớn cần phải được **mô tả có cấu trúc** cho dễ hiểu.

Các chương trình tiêu biểu thời đó có thể kể tới **hệ điều hành DOS, UNIX**...Lúc đó cũng đã có những cơ sở dữ liệu có thể truy cập từ xa. Do nhu cầu thực tế, các ngôn ngữ lập trình ra đời giai đoạn này đã có khả năng hỗ trợ phương pháp lập trình có cấu trúc, nghĩa là chương trình được **phân chia thành các chương trình con**, mỗi chương trình con có tên và các chức năng xử lý riêng, có thể giao tiếp với các chương trình con khác thông qua các đối số và kiểu trả về (nếu có).

Ở giai đoạn này phần mềm là thường là sản phẩm đa nhiệm, đa người dùng. Các hoạt động xử lý là xử lý số, ký tự, theo lô và thời gian thực. Giai đoạn này xuất hiện hình thức lưu trữ thông tin trực tuyến (CSDL).

Ngôn ngữ lập trình sử dụng trong giai đoạn này là các ngôn ngữ có cấu trúc như PL1, Algol 60, Fortran, COBOL, ..

Các tiêu chí đánh giá phần mềm gồm tính nhanh, giải được bài toán lớn, nhiều người dùng. Công nghệ: bán dẫn (tính toán nhanh hơn, bộ nhớ khá,...), cơ sở dữ liệu. Yêu cầu bảo trì: Sửa lỗi, thích nghi.

- Từ **1986 đến nay**: Đây là thời kỳ của máy vi tính PC, thời nối mạng tầm rộng, mạng toàn cầu Internet.

Đặc tả dự án được biết nhiều là hướng đối tượng, công cụ CASE (Computer Aided Software Engineering).

Trong thiết kế người ta chú ý đến môđun (module), đối tượng (object), giao thức (protocol) và giao diện (interface). Giao thức hay giao diện nói về sự trao đổi giữa các đối tượng.

Khi cài đặt trên máy tính người ta thường dùng ngôn ngữ hướng đối tượng, thế hệ 4, lập trình trực quan. Phần mềm lớn, tinh vi, tin cậy, hướng người dùng. Gồm các hệ chuyên gia, trí tuệ nhân tạo, phần mềm nhúng, các dịch vụ Web được sử dụng rộng rãi. Internet này cảng được mở rộng. Cơ sở dữ liệu hướng đối tượng, kho dữ liệu phát triển.

Ngoài các phần mềm được viết theo đặt hàng, người ta chú trọng đến các phần mềm đóng gói như: Netscape, Internet Explorer, Word, Excel...

Các tiêu chí đánh giá phần mềm gồm: Tính tiện dụng, độ tinh vi, tính tin cậy, tính dễ bảo trì. Công nghệ: CSDL quan hệ, vi mạch siêu tích hợp, internet, mạng không dây tốc độ cao, hướng đối tượng, Web

Ngày nay phần mềm ngày càng lớn, càng tích hợp. Phần mềm ngày nay còn được truy cập ở khắp nơi trên thế giới thông qua Internet.

Người ta cho rằng việc xây dựng một phần mềm cũng cần tuân theo những nguyên tắc và kỹ luật giống như khi xây dựng một chiếc cầu hay thực hiện những công việc kỹ nghệ khác. Nếu một chiếc cầu bị lỗi khi xây dựng và do đó bị sập thì tác hại gây ra rất lớn. Một phần mềm quan trọng như phần mềm điều khiển hệ thống tên lửa đạn đạo của một nước mà có lỗi, cho kết quả tính toán sai thì hậu quả nó gây ra cũng thật là kinh khủng. Chính vì vậy một nhóm nghiên cứu của NATO trong **năm 1967** đã sử dụng thuật ngữ **"Software engineering"**. Họ muốn rằng khi xây dựng phần mềm thì các kỹ sư cũng cần áp dụng các nguyên tắc của kỹ nghệ truyền thống.

Tuy nhiên, có nhiều sự khác biệt giữa sản phẩm của kỹ nghệ truyền thống và công nghệ phần mềm. Ví dụ như chiếc cầu và hệ điều hành chẳng hạn. Sự cố cầu sập rất ít khi xảy ra và nếu xảy ra thì cách khắc phục thường là xây dựng lại. Còn hệ điều hành nếu có sự cố có khi chỉ cần tắt máy khởi động lại là lại chạy tốt. Phần mềm có thể sửa lại, có khi là tới 50% để có thể chạy trên máy tính có cấu hình khác hoặc thực hiện thêm các chức năng khác. Còn chiếc cầu muốn sửa lại 50% thì rất khó, nhiều khi xây dựng mới còn dễ thực hiện hơn.

1.1.2 Từ góc độ kinh tế

Trong kỹ nghệ truyền thống, khi có nhiều cách thức để sản xuất một sản phẩm nào đó, ví dụ chiết xuất dầu lửa từ than đá chẳng hạn, các kỹ sư thường chọn phương án rẻ nhất. Khi xây dựng phần mềm cũng vậy, người ta thường lựa chọn cách thức chi phí ít nhất. Một phương pháp lập trình mới có thể viết chương trình nhanh hơn, nhưng chi phí huấn luyện sử dụng và công bảo trì có thể lớn hơn khiến người ta phải cân nhắc khi lựa chọn.

1.1.3 Khủng hoảng phần mềm

- Phần mềm được viết ngay từ khi xuất hiện các hệ máy tính và ngôn ngữ lập trình đầu tiên.

- Công nghệ phần cứng phát triển nhanh và mạnh. Tính năng và tiềm lực của nó ngày càng tăng. Các ứng dụng dựa trên phần cứng cũng phải phát triển theo, các phần mềm ngày càng trở nên phức tạp và khó hiểu.

- Trên thực tế nhu cầu ứng dụng công nghệ thông tin ngày càng cao. Mọi ngành nghề kinh tế, mọi hoạt động xã hội và đời sống kinh tế đều cần đến các phần mềm hỗ trợ. Dẫn đến các phần mềm ngày càng lớn và phức tạp.

=> Trên thực tế các sản phẩm phần mềm không đáp ứng kịp các yêu cầu của người sử dụng. Việc sản xuất phần mềm gặp thất bại quá nhiều. Hầu hết các sản phẩm phần mềm đều rơi vào tình trạng:

+ Không đáp ứng kịp các nhu cầu của người sử dụng

+ Vượt quá chi phí và thời hạn.

+ Tiềm ẩn nhiều lỗi trong các sản phẩm phần mềm

+ Không đảm bảo chất lượng.

Các dữ liệu quan sát được cho thấy:

- Cứ có 6 đề án được triển khai thì có 2 đề án bị thất bại
- 35% số dự án phần mềm thất bại vì các lý do: thời hạn, chi phí, chất lượng (không đáp ứng được nghiệp vụ, khó sử dụng, không tin cậy…)
- 45% : đã được phân phối, không được sử dụng
- 27% : không được phân phối
- 17% : bị hủy bỏ
- 6% : được sử dụng sau khi đã sửa đổi
- 5% : được sử dụng ngay sau khi phân phối

- Trung bình thời gian thực hiện thực tế bị kéo dài 50% (cá biệt lên tới 200 – 300%)
- 3/4 các hệ thống lớn có lỗi khi thực thi
- Quá trình phân tích yêu cầu (5% công sức): để lại 55% lỗi, có 18% phát hiện được
- Quá trình thiết kế (25% công sức):để lại 30% lỗi, có 10% phát hiện được
- Quá trình mã hóa, kiểm tra và bảo trì : Để lại 15% lỗi, có 72% phát hiện được

Để giải quyết vấn đề trên một hội nghị thế giới đã được triệu tập để bàn về cách giải quyết – Hội nghị của NATO về khủng hoảng phần mềm diễn ra 10/1968, đã:

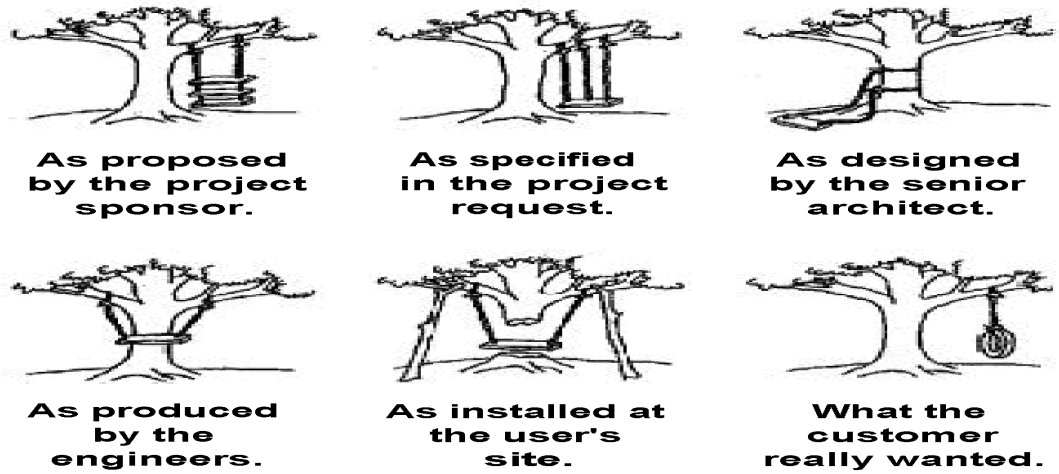
Xác định các nguyên nhân chính gây ra khủng hoảng phần mềm, đó là việc sản xuất các sản phẩm phần mềm theo phương pháp thủ công. Phương pháp này không thích hợp cho việc phát triển các sản phẩm phần mềm lớn và phức tạp. Phương pháp thủ công thể hiện như sau:

- Làm theo cảm tính: Dựa chủ yếu vào kinh nghiệm, không có phương pháp đủ tốt

- Phương tiện thô sơ: Chủ yếu là ngôn ngữ lập trình

- Làm đơn lẻ: Do một hoặc một số cá nhân thực hiện

Ví dụ Khủng hoảng phần mềm:



Khắc phục khủng hoảng phần mềm:

Xây dựng phần mềm theo công nghệ ~ công nghiệp hóa quá trình sản xuất phần mềm. Từ đó Khái niệm công nghệ phần mềm được đưa ra. Và từ đó công nghệ phần mềm thực sự trở thành một nghành nghiên cứu không thể thiếu được trong lĩnh vực CNTT.

Để đáp ứng đòi hỏi của phát triển phần mềm cần có lý thuyết, kỹ thuật, phương pháp, công cụ đủ tốt để điều khiển tiến trình phát triển hệ thống phần mềm. Công nghệ phần mềm nhằm nghiên cứu tất cả các khía cạnh liên quan đến việc sản xuất các sản phẩm phần mềm chuyên nghiệp. Nó liên quan tới lý thuyết, quy trình, phương pháp và công cụ cần để phát triển phần mềm.

Mục tiêu của công nghệ phần mềm: Sản xuất phần mềm độc lập, đúng hạn, phù hợp kinh phí và đáp ứng mọi yêu cầu người sử dụng.

Để xây dựng hệ thống phần mềm tốt ta cần

- Xác định đúng đắn tiến trình phát triển phần mềm
  - Các pha của hoạt động
  - Sản phẩm của mỗi pha
- Phương pháp và kỹ thuật áp dụng trong từng pha và mô hình hóa sản phẩm của chúng
- Công cụ phát sinh ra sản phẩm

Các sản phẩm phần mềm ngày nay đã đóng góp một phần lớn cho nền kinh tế của đất nước và đạt được các kết quả nhất định. Mặc dù CNPM đã qua gần 2 thập kỷ phát triển, đã đưa ra nhiều nguyên lý, công cụ, phương pháp tiến trình.... để phát triển các sản phẩm phần mềm. Tuy nhiên các dự án phát triển phần mềm vẫn tốn chí phí cao, sản phẩm phần mềm chưa đảm bảo chất lượng và còn tiềm ẩn nhiều lỗi, các chi phí phần lớn vẫn dành cho hoạt động kiểm thử đánh giá phần mềm. Một số nguyên nhân (Những khó khăn trong phát triển phần mềm):

- Năng lực máy tính ngày càng mạnh
- Các hệ thống được liên kết lại ngày càng lớn
- Thế giới thay đổi nhanh (cả nghiệp vụ lẫn công nghệ)
- Ham muốn của người dùng ngày càng nhiều.

=> Yêu cầu tiến hóa phần mềm là tất yếu

- Quá trình phát triển phần mềm chưa được thống nhất
- Chưa đạt được chuẩn cho việc đo lường hiệu xuất và sản phẩm
- Độ phức tạp phần mềm quá cao.
- Làm việc nhóm không đúng kỷ luật gây ra các lỗi
- Không có phương pháp mô tả rõ ràng định nghĩa yêu cầu của người dùng

- Tư liệu đặc tả đã cố định thời gian dài, do vậy khó đáp ứng nhu cầu thay đổi trong thời gian đó
- Phương pháp luận thiết kế không nhất quán thì sẽ dẫn đến suy giảm chất lượng phần mềm
- Không có chuẩn về làm tư liệu quy trình sản xuất phần mềm, thì những đặc tả không rõ ràng sẽ làm giảm chất lượng phần mềm
- Không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn
- Coi trọng việc lập trình hơn khâu thiết kế
- Coi thường việc tái sử dụng phần mềm, thì năng suất lao động sẽ giảm
- Không chứng minh được tính đúng đắn của phần mềm
- Chuẩn về một phần mềm tốt không thể đo được một cách định lượng
- Khi đầu tư nhân lực lớn vào bảo trì sẽ làm giảm hiệu suất lao động của nhân viên
- Quản lý dự án lỏng lẻo kéo theo quản lý lịch trình cũng không rõ ràng
- Không có tiêu chuẩn để ước lượng nhân lực và dự toán sẽ làm kéo dài thời hạn và vượt kinh phí của dự án

*** Thách thức đối với phần mềm**

- ✓ Phần mềm làm ra nhỏ hơn rất nhiều so với nhu cầu
- ✓ Việc khai thác phần mềm nhỏ hơn rất nhiều so với tiềm lực phần cứng
- ✓ Bảo trì những hệ thống phần mềm cũ, lạc hậu để sử dụng là cực kỳ khó khăn
- ✓ Về mặt công nghệ: Cần có các công nghệ, công cụ hiện đại để phát triển phần mềm
- ✓ Về mặt quản lý: Cần có phương pháp thích hợp (CMM, CMMI, RMM).

1.2 Một số khái niệm cơ bản

1.2.1 Phần cứng (hardware)

Là các thiết bị cấu kiện mang tính vật lý có thể tiếp xúc bằng tay được như máy in, ổ đĩa, máy quét, các mạch tích hợp (IC)....

1.2.2 Phần mềm (software)

Nghĩa đơn giản nhất của **phần mềm** (software) là các chương trình chứa các dòng lệnh chỉ thị cho máy tính thực hiện một công việc nào đó.

Đôi khi người ta gọi phần mềm là chương trình, cho dù thực tế thì phần mềm

gồm nhiều chương trình và dữ liệu. Ví dụ: chương trình quản lý nhân sự, chương trình quản lý tín dụng...

Trong công nghệ phần mềm, người ta hiểu **phần mềm**: ***không chỉ là các chương trình, dữ liệu, mà còn gồm cả các tài liệu liên quan như các bản đặc tả, thiết kế, hướng dẫn sử dụng, các thông tin cấu hình cần thiết để làm cho các chương trình này có thể vận hành một cách đúng đắn.***

Một hệ thống phần mềm bao gồm ba phần:

- ✓ Các chương trình máy tính riêng lẻ:
  - Các file mã nguồn
  - Các file mã máy (file text)
- ✓ Các cấu trúc dữ liệu:
  - Cấu trúc làm việc (bộ nhớ trong)
  - Cấu trúc lưu trữ (bộ nhớ ngoài)
- ✓ Các tài liệu liên quan:
  - Tài liệu hướng dẫn sử dụng *(dành cho người dùng cuối)*
  - Tài liệu tham khảo kỹ thuật *(dành cho người bảo trì phần mềm)*
  - Tài liệu phát triển *(dành cho nhà phát triển)*

Ngoài ra cần một Website cho người dùng để download thông tin sản phẩm hiện thời.

Các sản phẩm phần mềm có thể được chia thành 2 dạng:

+ ***Sản phẩm đại trà (Generic Product***): được phát triển để bán ra ngoài thị trường, đối tượng người sử dụng tương đối đa dạng và phong phú. Những sản phẩm phần mềm thuộc loại này thường là những phần mềm dành cho máy PC.

+ ***Sản phẩm theo đơn đặt hàng (Bespoke Product hoặc Customised Product)***: được phát triển cho một khách hàng riêng lẻ theo yêu cầu. Ví dụ: Những hệ thống phần mềm chuyên dụng, hỗ trợ nghiệp vụ cho một doanh nghiệp riêng lẻ.

Sự phân biệt này ngày càng trở nên mờ nhạt, vì ngày càng nhiều công ty phát triển phần mềm bắt đầu bởi các sản phẩm đại trà và tùy biến nó theo các yêu cầu của khách hành riêng lẻ.

a) Đặc trưng của phần mềm

- Phần mềm không "hỏng đi" nhưng thoái hoá theo thời gian

- Phần lớn phần mềm vẫn được xây dựng theo đơn đặt hàng của khách
- Sự phức tạp và tính thay đổi luôn là bản chất của phần mềm
- Phần mềm được phát triển theo nhóm
- Phần mềm cũng là một thứ hàng hóa như những thứ hàng hoá khác
- phần mềm mới có thể được tạo ra bằng cách phát triển các chương trình mới, cấu hình lại những phần mềm đại chúng hoặc sử dụng lại phần mềm đã có

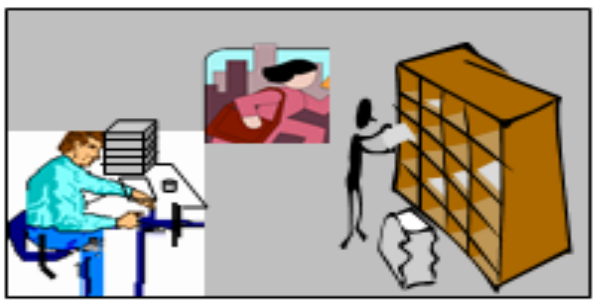
b) Vai trò của phần mềm

Các phần mềm máy tính ngày nay đóng vai trò quan trọng trong mọi lĩnh vực đời sống, kinh tế, xã hội của mọi quốc gia trên thế giới:

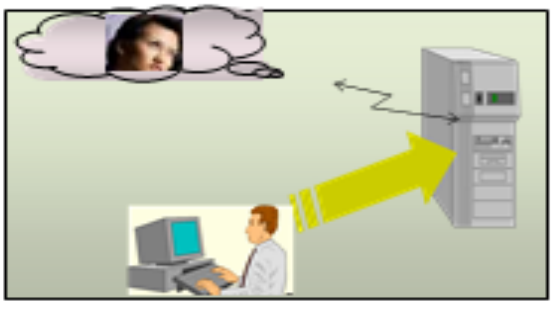
Các phần mềm là *linh hồn* của các hệ thống máy tính. Chúng có vai trò nền tảng của mọi hoạt động xã hội. Mọi nền kinh tế đều phụ thuộc rất lớn vào phần mềm.

- Số % thu, chi từ phần mềm chiếm đáng kể trong tổng GNP của mọi quốc gia:
  - ✓ *2006 ấn độ xuất gần **30 tỉ USD** phần mềm*
  - ✓ *T*hế giới có >7 triệu kỹ sư CNTT tạo ra **600 tỉ $/năm**
  - ✓ Chi phí cho phần mềm năm 2000 lên tới: **770 tỉ $**
- Phần mềm sai hỏng, kinh tế tổn thất lớn
  - ✓ *Vệ tinh Ariane 5 hỏng do lỗi phần mềm (1996) thiệt hại500 triệu $.*
  - ✓ *Website dùng 1 ngàyy mất hàng triệu $.* [Pankaj Jalote. CMM in practice, Addison-Wesley, tr.1,3,11]

Phần mềm tạo nên sự khác biệt giữa các tổ chức về phong cách và năng xuất



Trời ơi!



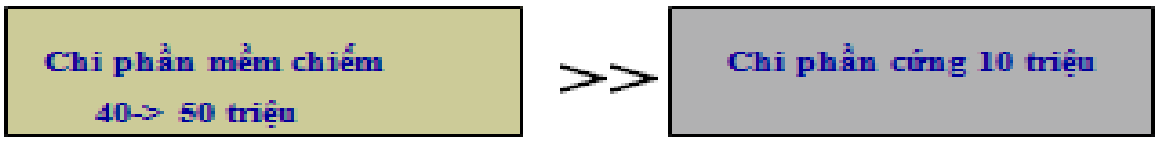
Tuyệt vời!

lao động:

Ngày càng nhiều hệ thống được phần mềm điều khiển, trợ giúp. Tính tự động hóa của các hệ thống này ngày càng tăng và chi phí phần mềm ngày càng lớn so với

chi phí phần cứng.

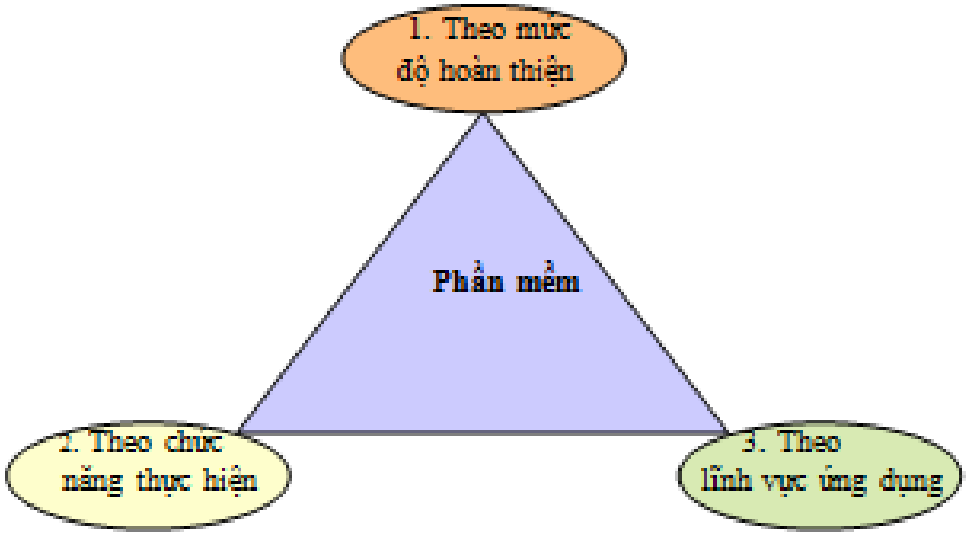
Ví dụ: Với hệ thống siêu thị



Ứng dụng phần mềm có mặt trên mọi lĩnh vực kinh tế, giáo dục, quân sự, trò chơi, ...

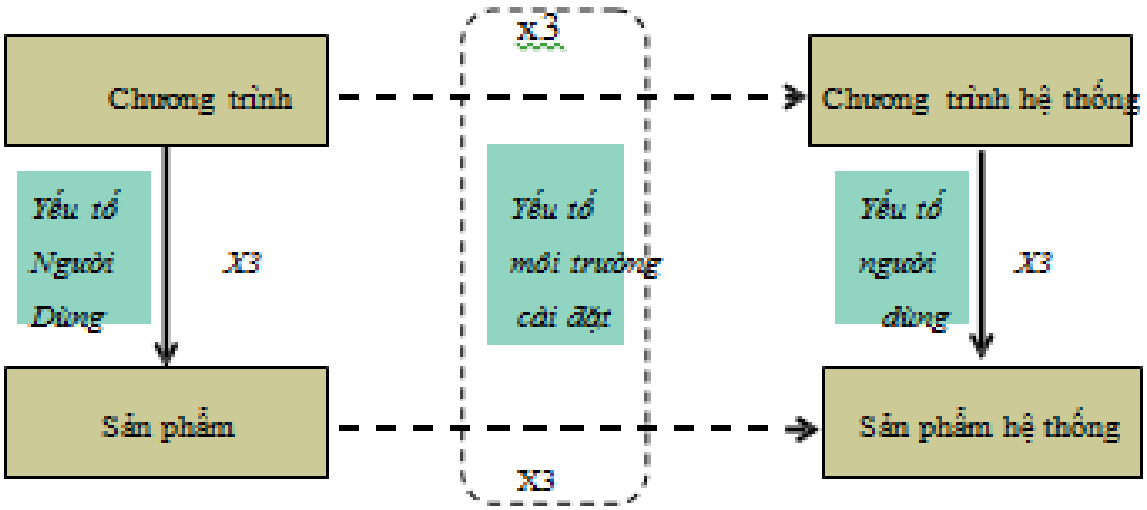
c) Phân loại phần mềm

Một phần mềm có thể được phân loại theo 3 tiêu chí khác nhau:



1. *Phân loại theo mức độ hoàn thiện*

Phần mềm được phân loại theo mức độ hoàn thiện tăng dần từ chương trình đến sản phẩm và cuối cùng là sản phẩm hệ thống.



Tính phức tạp tăng nhanh (9 lần) từ chương trình -> sản phẩm -> hệ thống. Trong đó:

- ✓ **Chương trình**
  - Một người viết, một người dùng (người viết ≡ người dùng).
  - Mục đích chính là thu thập thông tin, xử lý tín hiệu số (dùng một lần)
  - Thường không có tài liệu và không kiểm thử triệt để
- ✓ **Sản phẩm phần mềm**
  - Nhiều người viết, nhiều người dùng,
  - Độ phức tạp cao, đồng bộ, an toàn, an ninh.

    => Kinh nghiệm viết chương trình nhỏ không thể áp dụng cho các sản phẩm lớn.

2. ***Phân loại theo chức năng***

- ✓ Phần mềm hệ thống
  - Điều hành hoạt động máy tính, thiết bị và chương trình (OS, ...)
  - Trợ giúp các tiện ích (tổ chức tệp, nén, dọn đĩa, ...).
- ✓ Phần mềm nghiệp vụ
  - Trợ giúp các hoạt động nghiệp vụ khác nhau
  - Có số lượng lớn, đa dạng
  - Chúng thường được chia làm 2 loại theo cách thức phát triển:
    i. *Sản phẩm đặt hàng*: Thường sản xuất theo đơn đặt hàng (các hệ thống thông tin, hệ thống quản lý, ...). Chúng có đặc trưng đơn chiếc, được sản xuất theo các yêu cầu đắc thù riêng (dễ nhận dạng).

    ii. Sản phẩm chung (sản phẩm đại trà – software packets): Được phát triển để bán rộng rãi trên thị trường (còn gọi là những phần mềm thương mại, ví dụ các phần mềm văn phòng, ...). Chúng thường thỏa mãn các yêu cầu chung, số lượng người dùng lớn.

=> Mỗi loại có cách thức tiếp cận riêng để phát triển, chi phí, thời gian thực hiện cũng khác nhau. Ngày nay sự phân biệt giữa hai loại phần mềm này ngày càng trở nên mờ nhạt, vì ngày càng nhiều công ty phát triển phần mềm bắt đầu bởi các sản phẩm đại trà và tùy biến nó theo các yêu cầu của khách hành riêng lẻ.

- ✓ Phần mềm công cụ (Tools, CASE, ...):

Là những phần mềm trợ giúp cho quá trình phát triển phần mềm như các ngôn ngữ lập trình (trợ giúp các hoạt động soạn thảo mã nguồn, dịch, gỡ rối, ...). Các công cụ trợ giúp cho một hay nhiều giai đoạn phát triển (phân tích, thiết kế, quản lý dự án, kiểm thử, ...) như: Developer2000, Powerdesigner, WINE, Microsoft Project Management, RequisitePro…

*3. **Phân loại theo lĩnh vực ứng dụng***

- PM hệ thống: là những PM đảm nhận công việc tích hợp và điều khiển các thiết bị phần cứng đồng thời tạo ra môi trường thuận lợi để các PM khác nhau và người sử dụng có thể thao tác trên đó như một khối thống nhất mà không cần quan tâm đến các chi tiết kỹ thuật phức tạp bên dưới như: cách thức trao đổi dữ liệu giữa bộ nhớ chính và đĩa, các hiển thị văn bản lên màn hình…

Ví dụ: Trình biên dịch, hệ điều hành, các tiện ích quản lý tệp…..

- PM thời gian thực: Thu thập, xử lý các dữ liệu thời gian thực (dùng điều khiển các hệ thống tự động).

- PM nghiệp vụ: Xử lý các thông tin nghiệp vụ.

- PM khoa học kỹ thuật: Mô phỏng vật lý…

- PM Nhúng: Là một chương trình được viết, biên dịch trên máy tính và được nạp vào một hệ thống khác (gọi tắt là KIT) bao gồm một hoặc nhiều bộ vi xử lý đã được cài sẵn một hệ điều hành, bộ nhớ ghi chép được, các cổng giao tiếp với các phần cứng khác….

+ Mục đích của PM nhúng: Nhằm hỗ trợ cho các sản phẩm phần cứng các chức năng hoàn hảo nhất, phục vụ tốt nhất các nhu cầu của người dùng với sự bảo mật về sản phẩm tốt nhất.

+ Tính chất của PM nhúng: phụ thuộc vào hệ điều hành cài sẵn trên KIT, phụ thuộc vào các tính năng đặc trưng của từng sản phẩm phần cứng có trong KIT, phụ thuộc vào đặc tính của hệ thống.

Ví dụ: Máy ảnh KTS, lò vi ba, máy photocopy, máy in Laser, máy FAX, máy giặt, các bảng quảng cáo sử dụng hệ thống đèn LED….

Ngoài ra còn có thể phân loại phần mềm dựa trên quan điểm của người phát triển như sau:

- PM nguồn đóng: Là sản phẩm hoàn thiện, đã được đóng gói. Khó phát triển.

- PM nguồn mở: Cho phép người sử dụng có thể phát triển dựa trên mã nguồn mở.

1.2.3 Công nghệ (engineering)

Cách thức hay phương pháp để làm một việc gì đó, cụ thể hoặc trừu tượng, có áp dụng các thành tựu của khoa học và được thực hiện một cách có bài bản, hệ thống.

Công nghệ là cách sử dụng các công cụ, các kỹ thuật trong tiến trình giải quyết một vấn đề (công việc) nào đó. Mọi công nghệ (engineering) đều đề cập đến sản xuất sản phẩm theo tiến trình. Tổng quát thì tiến trình (process) xác định ai (Who) làm gì (What) và làm khi nào (When) và làm như thế nào (How) để đạt tới mục đích mong muốn.

1.2.4 Công nghệ phần mềm (software engineering)

- Công nghệ phần mềm là **quy tắc công nghệ** (engineering discipline) có liên quan đến tất cả các khía cạnh của tiến trình phát triển phần mềm như: sản xuất ntn? Đánh giá, đảm bảo chất lượng, quản lý ra sao....

- Tiến trình phát triển phần mềm (Software Development Process - SDP) là tiến trình xây dựng sản phẩm phầm mềm hay nâng cấp phần mềm đang có. Nó mô tả tập các hoạt động cần thiết để chuyển đổi từ yêu cầu người sử dụng sang hệ thống phần mềm

*** Định nghĩa CNPM cổ điển (Fritz Bauer)**

"*Công nghệ phần mềm là sự thiết lập và sử dụng các nguyên tắc khoa học nhằm mục đích tạo ra các sản phầm phần mềm một cách kinh tế mà các sản phầm phần mềm lại hoạt động một cách hiệu quả và tin cậy trên các máy tính*"

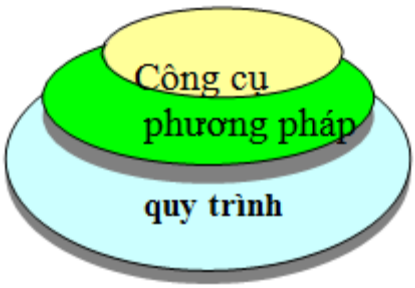
*** Định nghĩa khác về CNPM:**

- *CNPM là các quy trình đúng kỷ luật và có định lượng được áp dụng cho sự phát triển, thực thi và bảo trì các hệ thống thiên về phần mềm.*

- *CNPM tập trung vào quy trình, sự đo lường, sản phẩm, tính đúng thời gian và chất lượng.*

Theo quan điểm của nhiều nhà nghiên cứu, có thể nhìn ***công nghệ phần mềm*** là một ***mô hình được phân theo ba tầng*** mà tất cả các tầng này đều nhằm tới mục tiêu ***chất lượng, chi phí, thời hạn*** phát triển phần mềm.

Mô hình được phân theo ba tầng của công nghệ phần mềm được mô tả như sau:



Tầng quy trình (process) liên quan tới vấn đề quản trị phát triển phần mềm như lập kế hoạch, quản trị chất lượng, tiến độ, chi phí, mua bán sản phẩm phụ, cấu hình phần mềm, quản trị sự thay đổi, quản trị nhân sự (trong môi trường làm việc nhóm), việc chuyển giao, đào tạo, tài liệu;

Tầng phương pháp (methods) hay cách thức, công nghệ, kỹ thuật để làm phần mềm: liên quan đến tất cả các công đoạn phát triển hệ thống như nghiên cứu yêu cầu, thiết kế, lập trình, kiểm thử và bảo trì bằng cách đưa ra một số phương pháp hỗ trợ các công đoạn này. Phương pháp dựa trên những nguyên lý cơ bản nhất cho tất cả các lĩnh vực công nghệ kể cả các hoạt động mô hình hoá và kỹ thuật mô tả.

Tầng công cụ (tools) liên quan đến việc cung cấp các phương tiện hỗ trợ tự động hay bán tự động cho các tầng quá trình và phương pháp (công nghệ).

Như vậy:

Qua sơ đồ trên, ta thấy rõ công nghệ phần mềm là một khái niệm đề cập không chỉ tới các công nghệ và công cụ phần mềm mà còn tới cả cách thức phối hợp công nghệ, phương pháp và công cụ theo các quy trình nghiêm ngặt để làm ra sản phẩm có chất lượng.

Nhiệm vụ cơ bản của kỹ nghệ phần mềm là làm chủ độ phức tạp trong tiến trình phát triển phần mềm.

Kỹ sư phần mềm (software engineer): là một người biết cách áp dụng rộng rãi những kiến thức về CNPM, mục tiêu của kỹ sư phần mềm là sản xuất ra các sản phẩm có chất lượng cao và phù hợp với các quy trình phát triển chuẩn mực. **Công việc của người kỹ sư phần mềm là**: đánh giá, lựa chọn, sử dụng những cách tiếp cận có tính hệ thống, chuyên biệt, rõ ràng trong việc phát triển, đưa vào ứng dụng, bảo trì, và thay thế phần mềm.

Do đặc điểm nghề nghiệp, người kỹ sư phần mềm phải có những kỹ năng cơ bản như:

- Định danh, đánh giá, cài đặt, lựa chọn một phương pháp luận thích hợp và các công cụ CASE.

- Biết cách sử dụng các mẫu phần mềm (prototyping).
- Biết cách lựa chọn ngôn ngữ, phần cứng, phần mềm.
- Quản lý cấu hình, lập sơ đồ và kiểm soát việc phát triển của các tiến trình.
- Lựa chọn ngôn ngữ máy tính và phát triển chương trình máy tính.
- Đánh giá và quyết định khi nào loại bỏ và nâng cấp các ứng dụng.

1.2.5 Sự khác biệt giữa công nghệ phần mềm và khoa học máy tính?

Khoa học máy tính đề cấp tới lý thuyết và những vấn đề cơ bản liên quan đến MT; còn công nghệ phần mềm đề cập tới các hoạt động xây dựng và đưa ra một phần mềm hữu ích. Khi sự phát triển của phần mềm trở nên mạnh mẽ thì các lý thuyết của khoa học máy tính không đủ để đóng vai trò là nền tảng hoàn thiện cho công nghệ phần mềm.

1.2.6 Sự khác nhau giữa kỹ nghệ phần mềm và kỹ nghệ hệ thống

Kỹ nghệ (engineering): Là việc sử dụng phối hợp các công nghệ cần thiết để tạo ra các sản phẩm của một ngành nào đó.

Quá trình phát triển của Kỹ nghệ phần mềm

**** Đề xướng, hình thành năm 1968.***

Các kết quả nghiên cứu nổi bật đạt được những năm 70s là phương pháp lập trình có cấu trúc:

- ✓ Khái niệm về tính mô đun
- ✓ Khái niệm về sơ đồ khối, lập trình top – down
- ✓ Lập trình có cấu trúc (Dijkstra),
- ✓ Phương pháp chia mô đun cho chương trình
- ✓ Trừu tượng hóa dữ liệu (Liskov)

**** Tăng trưởng (nửa đầu những năm 1980)***

Giai đoạn này xuất hiện các phương pháp phát triển hệ thống:

- ✓ Công nghệ CSDL (mô hình quan hệ)
- ✓ Phân tích, thiết kế hướng cấu trúc (các biểu đồ luồng, ...),
- ✓ Các bộ công cụ phát triển như: Công cụ trợ giúp phân tích, thiết kế. Bộ khởi tạo chương trình, kiểm thử các ngôn ngữ bậc cao.

- ✓ Bắt đầu quan tâm đến hoạt động quản lý. Đề cập đến các độ đo phần mềm, quản lý theo thống kê

**** Phát triển (từ giữa những năm 1980 đến nay)***

Hoàn thiện công nghệ cấu trúc, ra đời công nghệ đối tượng:

- ✓ Nhiều mô hình hướng cấu trúc được triển khai và chuẩn hóa,
- ✓ Các CASE được bổ sung hoàn thiện, đạt mức tự động hóa cao
- ✓ Ngôn ngữ thế hệ thứ 4 ra đời như LIPS, PROLOG, ...
- ✓ Công nghệ hướng đối tượng bắt đầu phát triển như quy trình RUP, UML. Kho dữ liệu, CSDL hướng đối tượng, đa phương tiện, ...
- ✓ Các công cụ đầy đủ xuất hiện như ROSE, JIBUILDER, ...
- ✓ Sử dụng lại chiếm vị trí quan trọng trong phát triển phần mềm. Sử dụng lại thành phần, mẫu, Framework, ...
- ✓ Công nghệ Web phát triển: Các Web services, ...
- ✓ Phát triển các mô hình quản lý: Các chuẩn hóa quản lý được công nhận như CMM, ISO9000-03. Nhiều mô hình tổ chức làm phần mềm được đề xuất. Nhiều công cụ trợ giúp cho hoạt động quản lý dự án được hoàn thiện.

Kỹ nghệ hệ thống liên quan đến tất cả các khía cạnh về phát triển các hệ thống dựa trên máy tính, bao gồm:

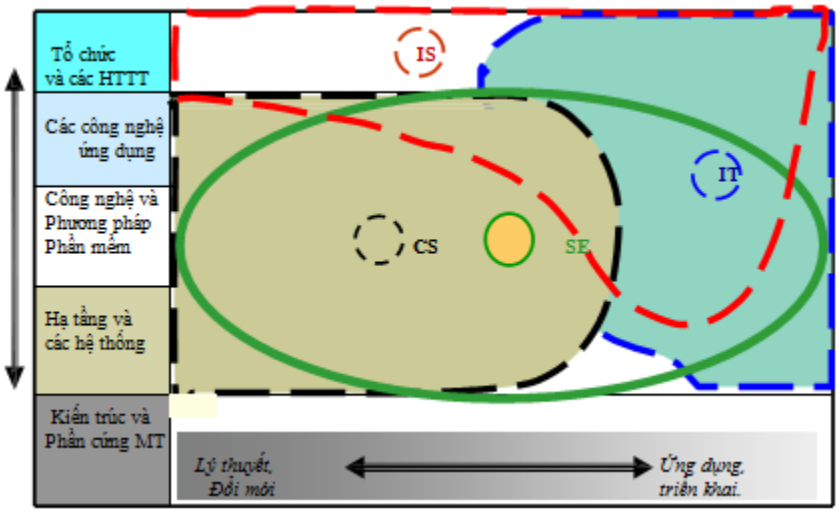
- Kỹ nghệ phần cứng
- Kỹ nghệ phần mềm
- Kỹ nghệ tiến trình

Kỹ nghệ phần mềm là một phần của tiến trình kỹ nghệ hệ thống, nó liên quan đến việc phát triển:

- Các cơ sở hạ tầng phần mềm
- Phần mềm điều khiển
- các ứng dụng và các CSDL trong hệ thống.

Phân biệt các lĩnh vực tính toán liên quan đến kỹ nghệ phần mềm

Các lĩnh vực tính toán liên quan đến Kỹ nghệ phần mềm được biểu diễn như sau [computing curricula 11/2004 -ACM, AIS, IEEE]:



Mỗi lĩnh vực tính toán được giới hạn bằng một vùng biên có mầu sắc như sau:

- ✓ CS: Khoa học máy tính
- ✓ IS: Hệ thống thông tin
- ✓ IT: Công nghệ thông tin
- ✓ **SE**: Kỹ nghệ phần mềm

Từ sơ đồ biểu diễn trên ta thấy, Kỹ nghệ phần mềm chiếm một phần lớn ở trung tâm, đó chính là lý do nó là ngành phức tạp và quan trọng.

1.2.7 Tiến trình phần mềm là gì?

Tiến trình phần mềm là một ***tập hợp các hoạt động có cấu trúc*** mà mục đích của nó là ***xây dựng, phát triển và tiến hóa phần mềm***. Một tiến trình cụ thể phải trả lời được câu hỏi: Làm gì? Khi nào làm? Ai làm? Làm như thế nào? Bằng gì? ở đâu? Kết quả? Tiêu chí đánh giá?

Đặc trưng của tiến trình phần mềm:

Gắn với mỗi dự án

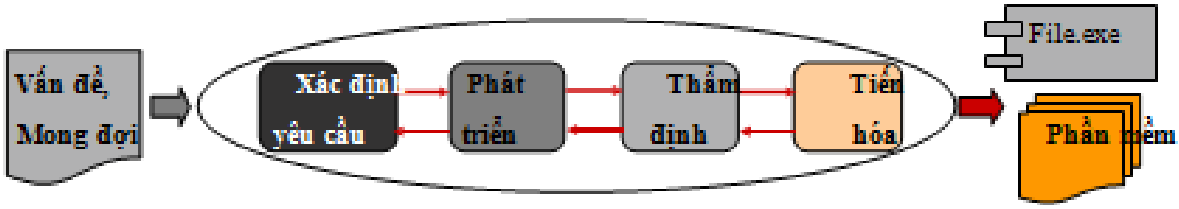
Có cấu trúc xác định: công việc gì, trình tự, công cụ, phương pháp

Sản phẩm cuối cùng là phần mềm bàn giao.

Các hoạt động chính của mọi tiến trình phần mềm:

- ✓ *Xác định yêu cầu*: Xác định rõ các yêu cầu sản phẩm
- ✓ *Phát triển*: Tạo ra sản phẩm
- ✓ *Thẩm định*: Phần mềm có đáp ứng được các yêu cầu không

✓ *Tiến hóa phần mềm*: Thay đổi phần mềm nhằm đáp ứng các yêu cầu thay đổi (người dùng, môi trường).



Những loại hệ thống khác nhau sẽ cần những quy trình phát triển khác nhau. Ví dụ, hệ thống thời gian thực yêu cầu phải hoàn thành đặc tả hệ thống trước khi chuyển sang giai đoạn xây dựng nó. Nhưng với hệ thống thương mại điện tử, chúng ta có thể vừa đặc tả vừa xây dựng chương trình một cách đồng thời.

=> Vì vậy, nếu ta không sử dụng một quy trình phát triển hệ thống thích hợp thì có thể làm giảm chất lượng của hệ thống và tăng chi phí xây dựng.

1.2.8 Mô hình quy trình phần mềm là gì?

Mô hình quy trình phát triển phần mềm là một thể hiện đơn giản của một quy trình phần mềm, và nó được biểu diễn từ một góc độ cụ thể. Sau đây là một số mô hình quy trình chung cũng được đề xuất như:

- *Mô hình thác nước (waterfall)*

- *Mô hình mẫu thử*

- *Mô hình công nghệ phần mềm dựa thành phần (Component-based software engineering).*

- *Mô hình xoắn ốc*

- *và một số mô hình khác*

1.2.9 Chi phí của công nghệ phần mềm bao gồm những gì?

* Chi phí của một hệ thống phần mềm:

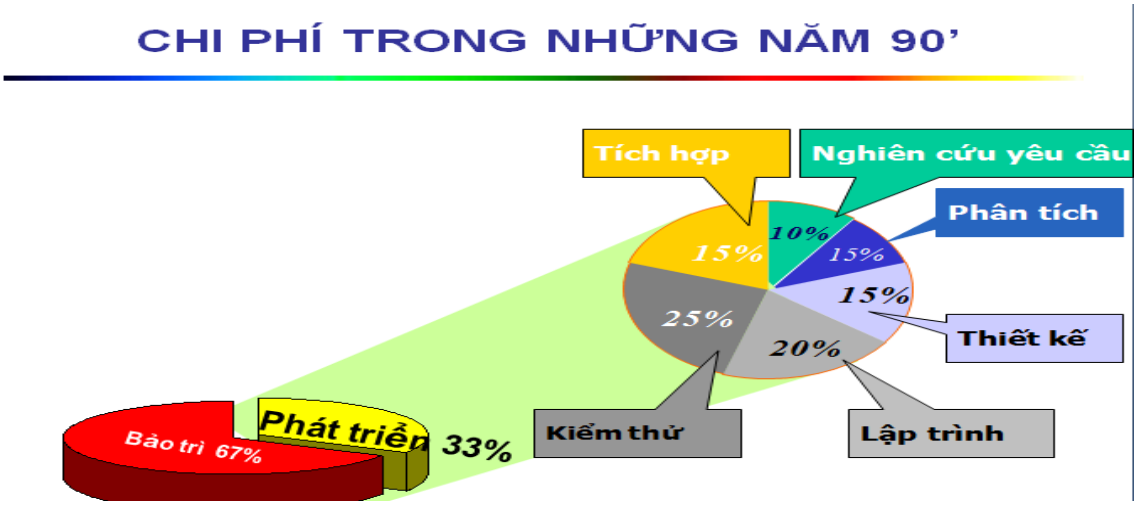
+ Chi phí phần cứng

+ Chi phí phần mềm:

- ✓ Chi phí sản xuất: Đặc tả, Xây dựng, kiểm thử
- ✓ Chi phí vận hành và bảo trì
- ✓ Chi phí cải tiến

Như vậy: Chi phí phần mềm thường chiếm phần lớn chi phí của cả hệ thống máy tính. Chi phí phần mềm trên máy PC thường lớn hơn chi phí phần cứng. Chi phí phần mềm dành cho việc bảo trì phần mềm thường lớn hơn chi phí sản xuất phần

mềm. đặc biệt đối với những hệ thống hoạt động trong thời gian dài, thì chi phí bảo trì thường lớn gấp nhiều lần so với chi phí sản xuất.



Chi phí biến đổi tuỳ thuộc vào từng loại hệ thống được xây dựng và các yêu cầu về đặc điểm của hệ thống như: hiệu năng và độ tin cậy của hệ thống.

Chi phí kỹ nghệ phần mềm là các khoản chi liên quan đến toàn bộ sự phát triển phần mềm. Chi phí phụ thuộc vào:

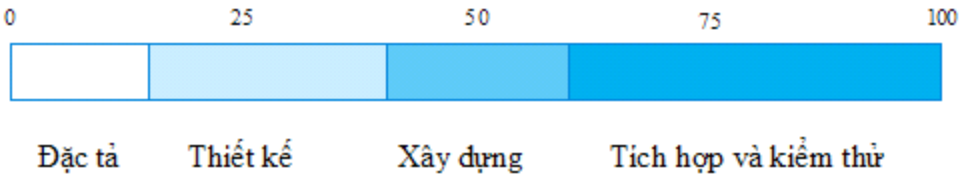
- ✓ Loại hệ thống (là đơn giản hay phức tạp);
- ✓ Yêu cầu đặt ra (nhiều, ít, cao, thấp);
- ✓ Mức độ hoàn thiện (hiệu năng, độ tin cậy, an toàn, ...);
- ✓ Năng lực của tổ chức (nhân lực, công cụ, công nghệ, kỹ năng có được, ...);
- ✓ Loại tiến trình sử dụng.

Chi phí của kỹ nghệ phần mềm được thống kê như sau:

Hình 1.1: Chi phí sản xuất phần mềm

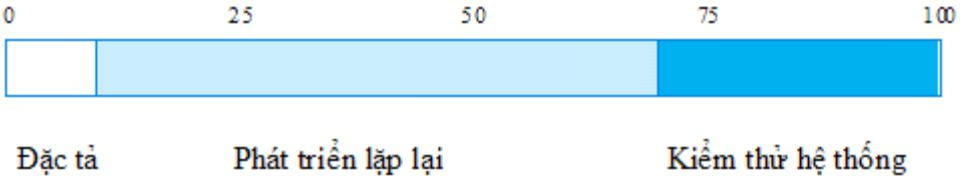
Việc phân bổ chi phí cũng phụ thuộc vào mô hình phát triển hệ thống được sử dụng. Sau đây là bảng so sánh chi phí của 3 mô hình phổ biến nhất, thường được sử dụng:

- ✓ Với mô hình thác nước, chi phí của các pha đặc tả, thiết kế, cài đặt, tích hợp và kiểm thử được xác định một cách riêng rẽ.



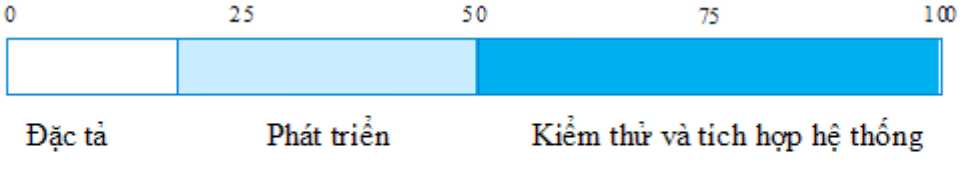
Hình 1.2: Phân bổ chi phí trong mô hình thác nước

- ✓ Với mô hình xoắn ốc, không thể phân biệt rõ chi phí cho từng pha trong quy trình. Chi phí đặc tả giảm vì đây là đặc tả ở bậc cao. Tại mỗi bước lặp, các pha trong quy trình xây dựng hệ thống được thực hiện lại nhằm thực hiện các yêu cầu hệ thống khác nhau ở từng bước lặp. Sau khi đã thực hiện hết các bước lặp, phải có chi phí kiểm thử toàn bộ hệ thống.



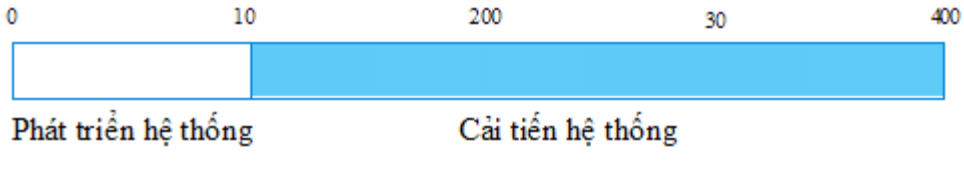
Hình 1.3: Phân bổ chi phí trong mô hình phát triển lặp lại

- ✓ Với mô hình công nghệ phần mềm hướng thành phần, chi phí phụ thuộc nhiều vào việc tích hợp và kiểm thử hệ thống.



Hình 1.4: Phân bổ chi phí trong công nghệ phần mềm hướng thành phần

Ngoài chi phí xây dựng, chúng ta còn phải để một phần lớn chi phí phục vụ cho việc thay đổi phần mềm sau khi nó đã được đưa vào sử dụng. Chi phí cải tiến phần mềm thay đổi phụ thuộc vào từng loại phần mềm.



Hình 1.5: Phân bổ chi phí trên các hệ thống có chu kỳ sống dài

1.2.10 Các phương pháp công nghệ phần mềm là gì?

Các phương pháp công nghệ phần mềm là các cách tiếp cận có cấu trúc chỉ ra cách thức để phát triển phần mềm. Mỗi phương pháp công nghệ phần mềm bao gồm:

- Các mô hình hệ thống

- các ký pháp: Ký hiệu dùng trong các mô hình,

- Các quy tắc/luật: chỉ ra các ràng buộc được áp dụng cho các mô hình hệ thống, ví dụ: mọi thực thể trong mô hình hệ thống phải có một tên duy nhất,

- Các hướng dẫn thiết kế và quy trình.

Để xây dựng phần mềm một cách dễ dàng, đảm bảo chất lượng cao và chi phí hiệu quả. Một số phương pháp công nghệ phần mềm đã được đề xuất như: Phân tích hướng cấu trúc - tập trung vào việc xác định các chức năng cơ bản của hệ thống; phương pháp hướng đối tượng - tập trung vào việc định nghĩa các đối tượng và sự cộng tác giữa chúng ...

1.2.11 CASE (Computer-Aided Software Engineering) là gì?

CASE là các hệ thống phần mềm nhằm mục đích hỗ trợ tự động cho các hoạt động trong quy trình xây dựng phần mềm. Có hai loại CASE:

- Upper-CASE: công cụ để hỗ trợ các hoạt động đầu tiên như đặc tả yêu cầu và thiết kế.

- Lower-CASE: công cụ để hỗ trợ các hoạt động sau như lập trình, gỡ lỗi và kiểm thử.

1.2.12 Thế nào là một phần mềm tốt? (Các đặc tính chủ yếu của hệ thống phần mềm tốt hiện nay)

Phần mềm phải đáp ứng các chức năng theo yêu cầu, có hiệu năng tốt, có khả năng bảo trì, đáng tin cậy, và được người sử dụng chấp nhận.

1.2.13 Những thách thức chính đối với công nghệ phần mềm?

Công nghệ phần mềm trong thế kỷ 21 phải đối mặt với rất nhiều thách thức to lớn. Với mỗi thách thức này, chúng ta phải có những giải pháp cụ thể.

- Sự tinh vi và năng lực của phần cứng đã vượt xa khả năng xây dựng phần mềm để có thể sử dụng được các tiềm năng của nó.
- Khả năng xây dựng các phần mềm mới không giữ được cùng nhịp so với nhu cầu về phần mềm tăng lên nhanh chóng, đặc biệt khi internet phát triển.
- Quy mô và độ phức tạp của các phần mềm mới ngày càng tăng. Khả năng bảo trì các hệ thống phần mềm cũ hiện đang tồn tại rất khó khăn và tốn kém các nguồn tài nguyên vì các thiết kế sơ sài. Phát triển các phần mềm mới phải nhanh chóng và dễ bảo trì trở thành nhu cầu cấp bách.

1.3 Các trách nhiệm đạo đức và nghề nghiệp

Các kỹ sư công nghệ phần mềm không chỉ đơn giản áp dụng các kỹ năng kỹ thuật nghiệp vụ của họ trong nghề nghiệp. Họ còn phải là những người trung thực, tuân theo các nội quy về đạo đức và các trách nhiệm nghề nghiệp nếu họ muốn trở thành các kỹ sư phần mềm chuyên nghiệp.

1.3.1 Các vấn đề về trách nhiệm nghề nghiệp

Các kỹ sư phần mềm cần:

Cẩn mật: Tôn trọng các thông tin bí mật và sự cẩn mật của người lao động và các khách hàng bất chấp có hay không bản hợp đồng về sự cẩn mật được ký kết.

Năng lực: Không được xuyên tạc năng lực của mình, phải biết chấp nhận những công việc nào là vượt quá khả năng của mình

Các quyền sở hữu trí tuệ: Cần nắm vững các luật hiện thời về sở hữu trí tuệ như các bằng sáng chế, bản quyền. Họ cần đảm bảo rằng các quyền sở hữu trí tuệ của khách hành và của người lao động được bảo vệ.

Không được lạm dụng máy tính của người khác: Các kỹ sư phần mềm không được sử dụng các kỹ năng nghề nghiệp của mình để sử dụng sai máy tính của người khác. Việc lạm dụng máy tính giới hạn từ những hành động không đáng kể như chới game trên MT người dùng đến những hành động nghiêm trọng như reo rắc virus trên MT của người khác.

1.3.2 Tập các chuẩn mực đạo đức

ACM và IEEE đã đưa ra một tập các chuẩn mực về đạo đức cho các kỹ sư CNPM. Tập chuẩn mực này gồm 8 chuẩn mực ở dạng ngắn và dạng dài:

1. **Xã hội (Public):** Kỹ sư phần mềm luôn làm việc vì lợi ích xã hội.
2. **Khách hàng và chủ doanh nghiệp** (**Client and employer**): Kỹ sư phần mềm có thái độ làm việc đảm bảo tối đa lợi ích của khách hàng và chủ doanh nghiệp, bên cạnh việc đảm bảo lợi ích xã hội.
3. **Sản phẩm** (**Product**): Kỹ sư phần mềm đảm bảo phần mềm của họ cũng sửa đổi (nâng cấp) phần mềm có liên quan đạt chuẩn chuyên môn cao nhất có thể.
4. **Phán xét (Judgment):** Kỹ sư phần mềm đảm bảo giá trị đạo đức của bản thân và tính độc lập trong các phán xét chuyên môn của mình.
5. **Quản lý (Management):** Các nhà quản lý và lãnh đạo công nghệ phần mềm cần cam kết công khai ủng hộ và đẩy mạnh việc quản lý có đạo đức trong phát triển và quản lý chất lượng phần mềm.
6. **Nghề nghiệp (Profession):** Kỹ sư phần mềm biết quảng bá các giá trị đạo đức và tiếng tăm của nghề nghiệp trong công nghệ phần mềm đi liền với các lợi ích xã hội.
7. **Đồng nghiệp (Colleagues):** Kỹ sư phần mềm luôn công bằng và luôn sẵn sàng hỗ trợ đồng nghiệp của mình.
8. **Bản thân (Self):** Kỹ sư phần mềm không nghừng học tập những kiến thức liên quan đến ngành nghề của mình và sẽ luôn quảng bá và đẩy mạnh yếu tố đạo đức trong nghề nghiệp của họ.

1.4 Nhân tố con người và sự phân hóa nghề nghiệp trong CNPM

1.4.1 Nhân tố con người trong ngành công nghệ phần mềm

Đối với một sản phẩn phần mềm, một người không thể hoàn thành mà là kết quả lao động của một nhóm người-ta gọi là nhóm phát triển phần mềm. Mỗi thành viên trong nhóm không được vị kỷ, thành quả lao động của nhóm được xen như là thành quả chung và phải tuyệt đối trung thành với nhóm.

Mỗi thành viên trong nhóm phải có một số kiến thức cần thiết tuỳ thuộc vào vai trò trong nhóm để phát triển phần mềm.

1.4.2 Phân loại nghề nghiệp

Yêu cầu hiện nay của sự phát triển Công nghệ Thông tin (CNTT) ở Việt nam đòi hỏi cần có những người lao động trong tất cả các ngành kinh tế biết sử dụng hữu hiệu CNTT trong công việc của mình, và đồng thời cần có những người trực tiếp tham gia vào sản xuất, kinh doanh, vận hành về CNTT. Do vậy cần có những lớp người lao động sau:

- Những người biết vận dụng sáng tạo CNTT vào nghiệp vụ chuyên môn.

- Những người tham gia quản lí và vận hành các hệ thống CNTT
- Những người tham gia trực tiếp vào việc phát triển và xây dựng ra các sản phẩm CNTT,...

Việc phân loại nghề nghiệp trong các hệ thống thông tin có thể được phân chia dựa vào các tiêu chuẩn như: mức độ kinh nghiệm, loại hình công việc,...

a) Mức độ kinh nghiệm

1. Sơ cấp: Nhân viên cán bộ ở mức độ sơ đẳng nhất trực tiếp được giám sát chặt chẽ, nhưng họ sẽ được làm những công việc đúng chuyên môn và đây là cấp độ tối thiểu. Thường thì phải mất khoảng hai năm để thực hiện các công việc đẳng cấp này.
2. Trung cấp: Những cán bộ có trình độ trung cấp hầu hết làm việc độc lập, yêu cầu trực tiếp một số các hoạt động. Những người bắt đầu ở mức độ trung cấp có 2 đến 4 năm kinh nghiệm. Thời gian trung bình ở cấp độ này từ 2 – 5 năm.
3. Cao cấp: Các cán bộ ở mức độ này có một trình độ nhất định về công việc và kinh nghiệm kỹ thuật đào tạo, huấn luyện người khác. Những cán bộ có từ 5 – 7 năm kinh nghiệm và có ít nhất là 3 năm để học các kỹ năng.
4. Lãnh đạo: Những nhà lãnh đạo làm việc một mình. Họ kiêm tất cả các nhiệm vụ giám sát. Một người lãnh đạo thường được gọi là những chuyên gia phụ trách các dự án. Những chuyên gia này có kinh nghiệm, kỹ năng cả ở trình độ đại học và có mong muốn được quản lý các vị trí.
5. Chuyên gia kỹ thuật: Chuyên gia kỹ thuật là người có kinh nghiệm rộng rãi trong nhiều lĩnh vực. Kinh nghiệm của một chuyên gia bao gồm phát triển ứng dụng, mạng, cơ sở dữ liệu và hệ điều hành. Các chuyên gia có thể làm việc trong các vị trí của hệ thống thông tin trong khoảng 10 năm hoặc có thể lâu hơn và cũng có thể duy trì lâu dài ở cấp độ này.
6. Nhà quản lý: Công việc quản lý một cách độc lập, thể hiện giá trị của riêng từng cá nhân, Các nhà quản lý có thể hoặc không thể trở thành chuyên gia kỹ thuật theo định hướng nhưng họ có kinh nghiệm làm việc và hầu hết họ đều có trách nhiệm trong cách quản lý. Đối với các nhà quản lý kỹ thuật việc phân chia các đặc điểm công việc là các kế hoạch mục tiêu, giám sát, quản lý cá nhân, các hoạt động liên lạc,... trong hoạt động quản lý dự án.

b) Loai hình công việc

Ở đây, các loại hình công việc được bàn luận đến dựa vào cách phân loại gồm: phát triển ứng dụng, hỗ trợ ứng dụng, chuyên ngành kỹ thuật, nhân viên và những vấn đề khác.

1. Phát triển ứng dụng

Lập trình viên: Các lập trình viên chuyển đổi những đồ án chi tiết kỹ thuật sang các module mã và tự kiểm tra các đơn vị. Các lập trình viên có thể luân phiên chịu trách nhiệm giữa phát triển ứng dụng và bảo trì.

Kỹ sư phần mềm: Một kỹ sư phần mềm thực hiện những chức năng của các nhà phân tích, các nhà thiết kế và các lập trình viên cũng như đứng ra lãnh đạo dự án hoặc quản lý dự án. Các phân tích gia ở trình độ đại học có thể tham gia lập kế hoạch và nghiên cứu khả thi. Một kỹ sư quản lý phần mềm sơ cấp thường dành nhiều thời gian lập trình trong khi một kỹ sư có trình độ cao cấp lại tập trung vào việc lập kế hoạch, nghiên cứu khả thi, phân tích và thiết kế.

Kỹ sư tri thức (KE): Các kỹ sư tri thức suy luận ra những mô hình ngữ nghĩa từ các chuyên gia để từ đó xây dựng những hệ chuyên gia và trí tuệ nhân tạo. Các kỹ sư tri thức tương tự như các kỹ sư phần mềm nhưng được chuyên môn hoá các kỹ năng để áp dụng vào các vấn đề trí tuệ nhân tạo. Việc phát triển các mô hình và các chương trình của cấu trúc trí tuệ đòi hỏi khả năng quan sát, kỹ năng phỏng vấn sâu sắc, khả năng trừu tượng hoá những vấn đề không phải của chuyên môn cá nhân để tạo ra những ý thức lập luận và thông tin cần thiết và khả năng phát triển những dự đoán về thông tin và tính chính xác với các chuyên gia.

2. Hỗ trợ ứng dụng

Chuyên gia ứng dụng: Chuyên gia ứng dụng có những vùng vấn đề được chuyên môn hoá cho phép họ tham khảo ý kiến của các đội dự án về một loại ứng dụng cụ thể.

Quản trị dữ liệu: Người quản lý dữ liệu quản lý thông tin như một nguồn thống nhất. Quản trị cơ sở dữ liệu (DBA): Những người quản lý cơ sở dữ liệu quản lý môi trường dữ liệu vật lý của một tổ chức. DBA phân tích, thiết kế, xây dựng và bảo lưu cơ sở dữ liệu cũng như môi trường phần mềm cơ sở dữ liệu.

Kỹ sư trí tuệ nhân tạo: Các kỹ sư trí tuệ nhân tạo làm việc như cố vấn giúp các đội dự án xác định, thiết kế và cài đặt trí tuệ vào các ứng dụng. Kỹ sư trí tuệ nhân tạo cùng với các kỹ sư tri thức dịch và kiểm tra những vấn đề miền dữ liệu và thông tin lập luận bằng một ngôn ngữ của trí tuệ nhân tạo. Các kỹ sư trí tuệ nhân tạo đạt được trình độ chuyên môn cao hơn các kỹ sư tri thức.

Nhà tư vấn: Người tư vấn thì biết mọi vấn đề và thực hành được tất cả. Số năm kinh nghiệm càng cao thì kiến thức có được càng nhiều.

3. Chuyên ngành kỹ thuật

Nhà phân tích và kỹ sư truyền thông: Các nhà phân tích và kỹ sư truyền thông phân tích, thiết kế, đàm phán và/ hoặc cài đặt các thiết bị và phần mềm truyền thông. Họ đòi hỏi liên quan chặt chẽ tới kỹ thuật truyền thông và có thể làm việc trên mainframe hoặc các mạng truyền thông dựa vào PC. Để bắt đầu ở mức xuất phát thì nền tảng kiến thức phải có là điện tử, kỹ thuật, các ứng dụng, khoa học máy tính và truyền thông.

Chuyên gia về mạng cục bộ: Các chuyên gia mạng cục bộ đặt kế hoạch, lắp đặt, quản lý và duy trì những khả năng của mạng cục bộ. Điểm khác nhau duy nhất giữa các chuyên gia mạng cục bộ và các chuyên gia truyền thông là phạm vi. Các chuyên gia truyền thông làm việc với nhiều mạng kể cả mainframe; còn chuyên gia mạng cục bộ chỉ làm việc trên những mạng có giới hạn về mặt địa lý và được cấu thành bởi nhiều máy tính cá nhân. Những người quản lý mạng cục bộ là vị trí đầu tiên trong nhiều công ty. Một người quản lý mạng cục bộ tạo ra người sử dụng mới, thực hiện hoặc thay đổi mức hoặc mã bảo mật, cài đặt những version mới của phần mềm điều hành mạng cục bộ, cài đặt những version mới của cơ sở dữ liệu hoặc phần mềm cơ sở mạng cục bộ khác. Giám sát tài nguyên cung cấp qua mạng cục bộ, cung cấp bản sao và khả năng phục hồi cho mạng cục bộ, và quản lý cấu hình mạng cục bộ.

Lập trình viên hệ thống: Các lập trình viên hệ thống cài đặt và bảo dưỡng hệ điều hành và ứng dụng hỗ trợ phần mềm. Giám sát hàng trăm ứng dụng để xem xét những rắc rối của nó có liên quan đến vấn đề của hệ thống hay không là một nhiệm vụ quan trọng.

Chuyên gia hỗ trợ phần mềm (SSP): Hỗ trợ phần mềm ứng dụng tương tự nhưng khác với lập trình viên hệ thống. SSP cài đặt và bảo dưỡng gói phần mềm sử dụng bởi cả các nhà phát triển ứng dụng và người sử dụng. Chúng có thể là cơ sở dữ liệu, ngôn ngữ hỏi đáp, sao lưu và phục hồi, bảng tính, quản lý khoảng trống đĩa, giao diện, truyền thông.

4. Nhân viên

Chuyên gia về bảo mật: Một chuyên gia bảo mật chịu trách nhiệm bảo mật và sẵn sàng phục hồi thảm hoạ.

Kiểm soát viên (EDP): Các kiểm soát viên EDP thực hiện việc kiểm tra khả năng tin cậy của những thiết kế ứng dụng.

Đào tạo viên: Một người đào tạo kỹ thuật học công nghệ mới, các sản phẩm đại lý, những đặc điểm ngôn ngữ mới,... sau đó dạy những người khác trong tổ chức sử dụng.

Người viết các chuẩn và kỹ thuật: Những người phát triển chuẩn làm việc

với những người quản lý để định ra những mặt công việc họ muốn chuẩn hoá và để tiêu chuẩn hoá những yêu cầu thành những chính sách và thủ tục chuẩn hoá cho tổ chức. Những kỹ năng quan trọng nhất đối với người phát triển chuẩn là ngôn ngữ và chữ viết truyền thông. Phát triển tiêu chuẩn và việc viết kỹ thuật là các hoạt động có liên quan với nhau. Người viết kỹ thuật lấy thông tin và sản phẩm phần mềm, ứng dụng hoặc những sản phẩm công nghệ thông tin khác và viết tài liệu để mô tả những đặc điểm, chức năng, công dụng của chúng.

Đảm bảo chất lượng (QA): Các dạng kiểm tra khác nhau tuỳ thuộc vào sản phẩm được duyệt. Anh ta hay cô ta cần phải tham gia đến khi sản phẩm đầu tiên của nhóm phát triển xuất hiện. Sau đó khi mà tài liệu đã có, người phân tích đảm bảo chất lượng phải xem xét sự thống nhất, hoàn thiện, chính xác uyển chuyển linh động của nó. Bất cứ vấn đề nào xuất hiện trong quá trình xem xét phải được ghi lại để trình lên người quản lý dự án.

Lập kế hoạch công nghệ: Các chuyên gia giám sát sự phát triển công nghệ xác định các xu hướng, lựa chọn các công nghệ thích hợp để thử nghiệm trong tổ chức và cuối cùng chạy đua trong thực hiện các kỹ thuật mới trong tổ chức. Những nhân viên cao cấp là cầu nối giữa thế giới bên ngoài và cộng đồng các đại lý với công ty. Đội ngũ nhân viên sơ cấp có thể làm việc với nhân viên cao cấp để tìm ra những chỉ dẫn trong hợp tác và quản lý công nghệ.

5. Những vấn đề khác

Hỗ trợ sản phẩm: Nhân viên hỗ trợ sản phẩm làm việc cho nhóm người dùng cuối hoặc bán hàng để cung cấp những chuyên môn kỹ thuật liên quan đến sản phẩm hoặc những hỗ trợ trên đường dây nóng khác. Ngoài những kiến thức kỹ thuật về sản phẩm, các cá nhân trong công việc này còn phải có kỹ năng trả lời điện thoại tốt và phải nói bằng ngôn ngữ không chuyên đối với người sử dụng về các vấn đề.

Tiếp thị sản phẩm: Nhân viên hỗ trợ tiếp thị làm việc cho nhà bán hàng để cung cấp những thông tin kỹ thuật cho đại diện bán hàng trong các tình huống tiếp thị. Loại công việc này đòi hỏi khả năng giao tiếp và kỹ năng giao tiếp tốt với một vài kiến thức về tiếp thị.

CHƯƠNG 2. QUY TRÌNH XÂY DỰNG PHẦN MỀM

MỤC ĐÍCH

- Hiểu rõ quy trình xây dựng phần mềm

- Nắm được một số mô hình phát triển phần mềm

- Xác định chi tiết những công việc phải làm trong quy trình phần mềm và cách thực hiện chúng.

- Có thể ứng dụng những mô hình phát triển phần mềm đã nghiên cứu trên những hệ thống phần mềm cụ thể.

Quy trình phần mềm là một ***tập hợp các hành động*** mà mục đích của nó là ***xây dựng và phát triển phần mềm***. Những hành động thường được thực hiện trong các quy trình phần mềm bao gồm:

- Đặc tả: đặc tả những gì hệ thống phải làm và các ràng buộc trong quá trình xây dựng hệ thống.

- Phát triển: xây dựng hệ thống phần mềm.

- Kiểm thử: kiểm tra xem liệu phần mềm đã thoả mãn yêu cầu của khách hàng.

- Mở rộng: điều chỉnh và thay đổi phần mềm tương ứng với sự thay đổi yêu cầu.

Những ***loại hệ thống khác nhau*** sẽ ***cần những quy trình phát triển khác nhau***.

Vì vậy, nếu ta không sử dụng một quy trình phát triển hệ thống thích hợp thì có thể làm giảm chất lượng của hệ thống và tăng chi phí xây dựng.

Mô hình quy trình phát triển phần mềm là gì?

Mô hình quy trình phát triển phần mềm là một thể hiện đơn giản của một quy trình phần mềm, và nó được biểu diễn từ một góc độ cụ thể. Sau đây là một số mô hình quy trình chung cũng được đề xuất như:

- Mô hình thác nước (waterfall)
- Mô hình mẫu thử
- Mô hình công nghệ phần mềm dựa thành phần (Component-based software engineering).
- Mô hình xoắn ốc
- và một số mô hình khác

2.1 Mô hình phát triển phần mềm

2.1.1 Mô hình thác nước (Mô hình tuyến tính - The linear sequential model)

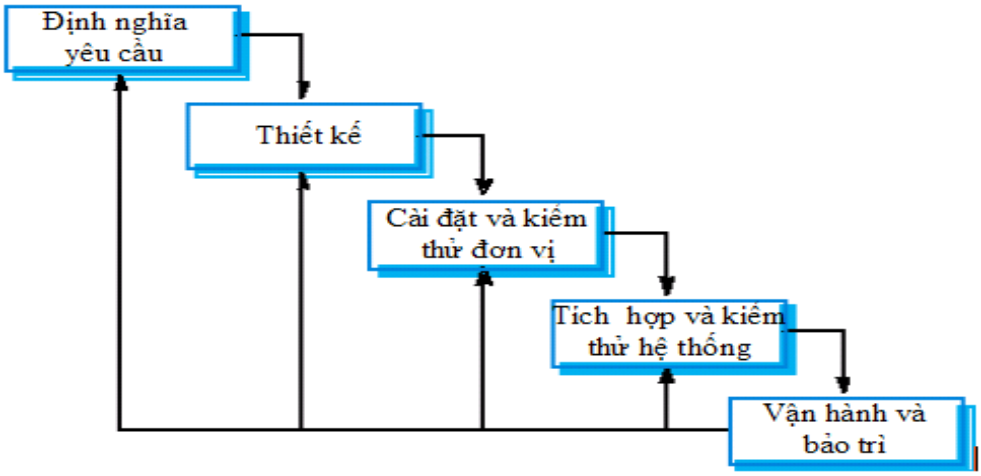
Đôi lúc còn được gọi là mô hình kinh điển (classic model). Mô hình này xem quá trình xây dựng một sản phẩm phần mềm bao gồm nhiều giai đoạn tách biệt, sau khi hoàn tất một giai đoạn thì chuyển đến giai đoạn sau.

Có hai hoạt động phổ biến được thực hiện trong mỗi giai đoạn là:

- Kiểm tra – phê chuẩn
- Quản lý cấu hình.

Tổng kết mỗi giai đoạn là sự kiểm tra, có phê chuẩn và quản lý cấu hình, đây chính là mục tiêu của sản phẩm. Việc kiểm tra đưa ra khuôn mẫu đúng đắn tương ứng giữa sản phẩm phần mềm và các đặc tính của nó. Sự phê chuẩn đưa ra chuẩn mực về sự phù hợp hay chất lượng của sản phẩm phần mềm đối với mục đích của quá trình hoạt động.

Các pha của mô hình thác nước bao gồm:



Hình 2.1: Mô hình thác nước

Trong mô hình thác nước, năm pha trên phải được thực hiện một cách tuần tự; kết thúc pha trước, rồi mới được thực hiện pha tiếp theo. Sản phẩm đầu ra của giai đoạn trước trở thành đầu vào của giai đoạn sau.

Trong một phân tích thú vị về các dự án thực tế, Brada thấy rằng bản chất tuyến tính của vòng đời cổ điển dẫn tới "các trạng thái nghẽn" mà trong đó một số thành viên của đội dự án phải đợi cho các thành viên khác của tổ hoàn thành các nhiệm vụ phụ thuộc. Trong thực tế, thời gian mất cho việc chờ đợi có thể vượt quá thời gian dành cho công việc sản xuất. Trạng thái nghẽn có khuynh hướng phổ biến vào lúc đầu và cuối của tiến trình tuần tự tuyến tính.

Nhận xét:

*** Ưu điểm:**

- Dễ quản lý. Đây chính là mô hình ưa thích của các nhà quản lý dự án. Thời gian hoàn thành dự án thường được dự báo với độ chính xác hơn so với các mô hình khác. Các tài liệu đầu ra của từng giai đoạn cũng được xây dựng đầy đủ và hệ thống hơn.

- Dễ dàng trong thẩm tra đánh giá.

- Mang tính tự nhiên khi sử dụng.

*** Nhược điểm:**

- Mối quan hệ giữa các giai đoạn không được thể hiện
- Hệ thống phải được kết thúc ở từng giai đoạn do vậy rất khó thực hiện được đầy đủ những yêu cầu của khách hàng... vì trong mô hình này 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 lúc này lại có sự thay đổi yêu cầu của người sử dụng; thì chỉ còn cách là phải thực hiện lại từ đầu.
- Khách hàng phải kiên nhẫn. Họ chỉ được tham gia vào dự án ở giai đoạn phân tích yêu cầu và test mà thôi. Ngoài ra sản phẩm sẽ chỉ được bàn giao khi tất cả các công việc liên quan đã được hoàn thành.

=> Mô hình thác nước chỉ nên được sử dụng khi đội dự án đã có kinh nghiệm, yêu cầu từ khách hàng được xác định rõ ngay từ đầu và ít có khả năng thay đổi.

2.1.2 Mô hình mẫu thử (Prototyping model) – Mô hình xây dựng tiến triển

Thông thường, khách hàng sẽ đưa ra mục tiêu của họ một cách chung chung mà họ không biết hoặc không đưa ra một cách cụ thể những cái vào, cái ra và các tiến trình xử lý chúng. Thêm vào đó, chúng ta cũng không thể không quan tâm đến thuật toán sử dụng, tính tương thích của sản phẩm phần mềm với môi trường của nó như: phần cứng, hệ điều hành...Trong trường hợp này, mô hình mẫu có thể là sự lựa chọn tốt hơn cho người lập trình.

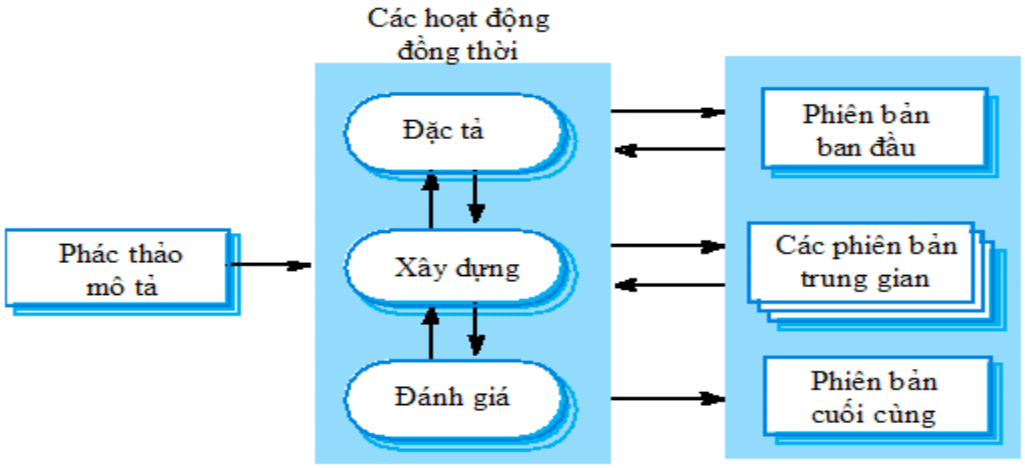
Mô hình xây dựng tiến triển dựa trên ý tưởng xây dựng một mẫu thử ban đầu và đưa cho người sử dụng xem xét; sau đó, tinh chỉnh mẫu thử qua nhiều phiên bản cho đến khi thoả mãn yêu cầu của người sử dụng thì dừng lại.

- Có hai phương pháp để thực hiện mô hình này:

  + **Phát triển thăm dò:** mục đích của nó là để làm việc với khách hàng và để đưa ra hệ thống cuối cùng từ những đặc tả sơ bộ ban đầu. Phương pháp này thường bắt đầu thực hiện với những yêu cầu được tìm hiểu rõ ràng và sau

đó, bổ sung những đặc điểm mới được đề xuất bởi khách hàng. Cuối cùng, khi các yêu cầu của người sử dụng được thoả mãn thì cũng là lúc chúng ta đã xây dựng xong hệ thống.

+ **Loại bỏ mẫu thử**: mục đích là để tìm hiểu các yêu cầu của hệ thống. Phương pháp này thường bắt đầu với những yêu cầu không rõ ràng và ít thông tin. Các mẫu thử sẽ được xây dựng và chuyển giao tới cho người sử dụng. Từ đó, ta có thể phân loại những yêu cầu nào là thực sự cần thiết và lúc này mẫu thử không còn cần thiết nữa. Như vậy, mẫu thử chỉ có tác dụng để làm sáng tỏ yêu cầu của người sử dụng.



Hình 2.2: Mô hình xây dựng tiến triển

Nhận xét:

Ưu điểm:

- Chú trọng tái sử dụng mẫu. Một phần của hệ thống có thể được phát triển ngay trong giai đoạn phân tích phát triển yêu cầu và thiết kế.
- Hoạt động nào cũng có sự tham gia của người dùng, nên sản phẩm cuối cùng mang tính khách quan, thỏa mãn yêu cầu của người sử dụng.
- Cho phép thay đổi yêu cầu và khuyến khích người sử dụng tham gia trong suốt chu kỳ dự án.
- Nhanh chóng xác định được các yêu cầu
- Tạo cơ sở ký kết hợp đồng
- Giúp đào tạo huấn luyện người sử dụng.
- Các hoạt đông xảy ra đồng thời, hoạt động này bổ sung cho hoạt động kia.

Nhược điểm của mô hình xây dựng tiến triển là: thiếu tầm nhìn của cả quy

trình; các hệ thống thường hướng cấu trúc nghèo nàn; yêu cầu các kỹ năng đặc biệt (Ví dụ: các ngôn ngữ để tạo ra mẫu thử nhanh chóng). Input/output chưa rõ ràng, Khó đánh giá tính hiệu quả của thuật toán.

Mô hình xây dựng tiến triển chỉ nên áp dụng với những hệ thống có tương tác ở mức độ nhỏ hoặc vừa; trên một phần của những hệ thống lớn (ví dụ như giao diện người sử dụng); hoặc những hệ thống có thời gian chu kỳ tồn tại ngắn.

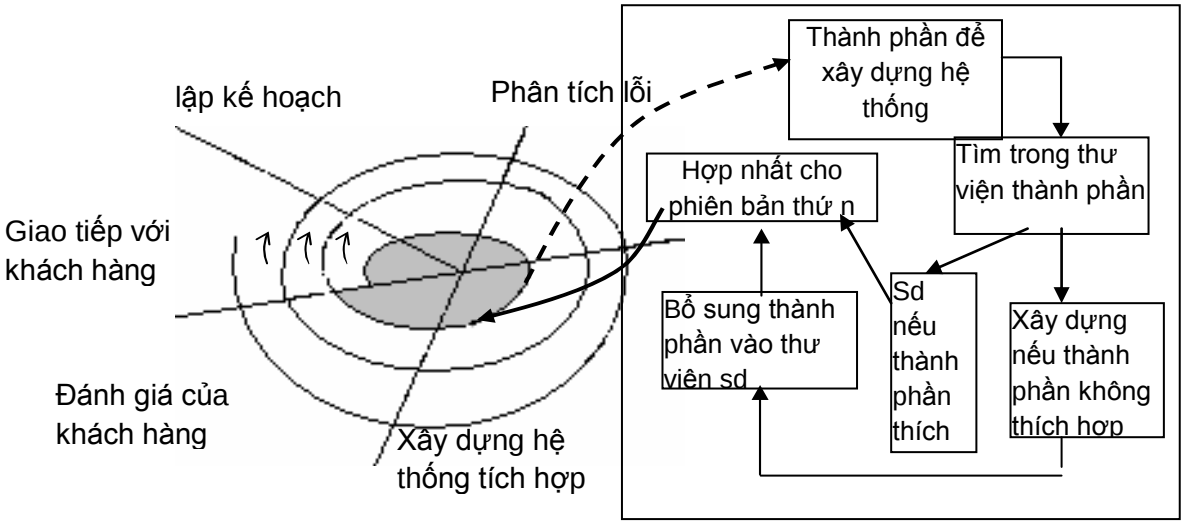
2.1.3 Mô hình phát triển dựa trên thành phần

Mô hình này dựa trên kỹ thuật tái sử dụng một cách có hệ thống; trong đó hệ thống được tích hợp từ nhiều thành phần đang đã có. Do vậy, kiến trúc phần mềm của hệ thống dựa vào kiến trúc phần mềm của các thành phần phần mềm tiêu chuẩn nên hệ thống đạt chất lượng cao hơn.

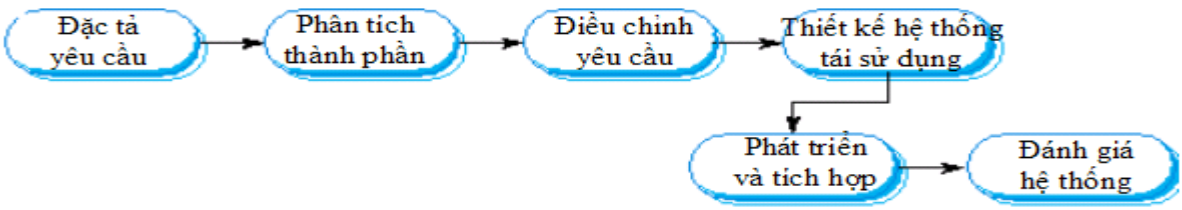
Phương pháp phát triển dựa trên thành phần gần tương tự như phương pháp phát triển hướng đối tượng. Hoạt động công nghệ bắt đầu với sự chỉ ra các lớp tham dự để phát triển hệ thống. Nếu các lớp này được tìm thấy trong thư viện và sự thích nghi là tốt, chúng sẽ được lấy ra và phát triển hệ thống. Ngược lại, chúng sẽ được phát triển để sử dụng và bổ sung vào thư viện sử dụng lại.

Thành phần phần mềm được sử dụng lại có độ chính xác cao và có thể nói là không chứa lỗi. Mặc dầu không thường xuyên được chứng minh về mặt hình thức nhưng với việc sử dụng lại, lỗi được tìm thấy và loại trừ; chất lượng của thành phần được cải thiện như là một kết quả.

Khi những thành phần sử dụng lại được ứng dụng thông qua tiến trình phần mềm, chúng ta ít tốn thời gian để tạo ra kế hoạch, mô hình, tài liệu, mã và dữ liệu mà chúng là cần thiết để tạo ra hệ thống. Thêm vào, chức năng cùng mức được phân phối cho người sử dụng với đầu vào ít công sức hơn, do vậy, hiệu suất phần mềm được cải thiện.



(hoặc:)



Hình 2.3: Công nghệ phần mềm hướng thành phần

Ưu điểm:

- Việc sử dụng các thành phần riêng biệt nên các thành phần được phát triển độc lập. Điều đó làm cho hình thành một thư viện các thành phần. Từ thư viện thành phần đó dẫn đến việc phát triển các hệ thống trở nên nhanh chóng nên chi phí cho phát triển PM giảm. Bên cạnh đó, các thành phần này được kiểm chứng tính đúng đắn, kiểm thử các lỗi logic,…. Nên hạn chế được rủi ro cho hệ thống. Đây là một thế mạnh mà các hệ thống khác khó đạt được.

- Các thành phần được tái sử dụng một cách hiệu quả. Mỗi lần phát triển một hệ thống mới không cần phải phát triển lại hoàn toàn mới các thành phần mà vẫn có thể sử dụng các thành phần đã có để phát triển tiếp. Kết thúc quá trình phát triển hệ thống mới, các thành phần mới được đưa vào thư viện thành phần. Với việc tái sử dụng thành phần, giá thành sẽ được giảm đi một cách đáng kể. không những vậy, chất lượng cũng được nâng cao, đảm bảo yêu cầu của người sử dụng.

Nhược điểm:

- Các yêu cầu thỏa hiệp là không thể tránh khỏi và điều này dẫn tới một hệ thống không đáp ứng được các nhu cầu thực của người dùng.

- Hơn nữa, một số kiểm soát sự phát triển hệ thống sẽ bị mất đi vì các phiên bản mới của các thành phần tái sử dụng không nằm dưới sự kiểm soát của tổ chức sử dụng chúng.

2.1.4 Mô hình xoắn ốc

Mô hình này được Boehm đưa ra nên đôi lúc còn được gọi là mô hình Boehm's (The Boehm's spiral model - 1988). Nó có thể xem là sự kết hợp giữa mô hình thác nước và mô hình mẫu và đồng thời thêm một thành phần mới - phân tích rủi ro. Trong mô hình xoắn ốc, quy trình phát triển phần mềm được biểu diễn như một vòng xoắn ốc.

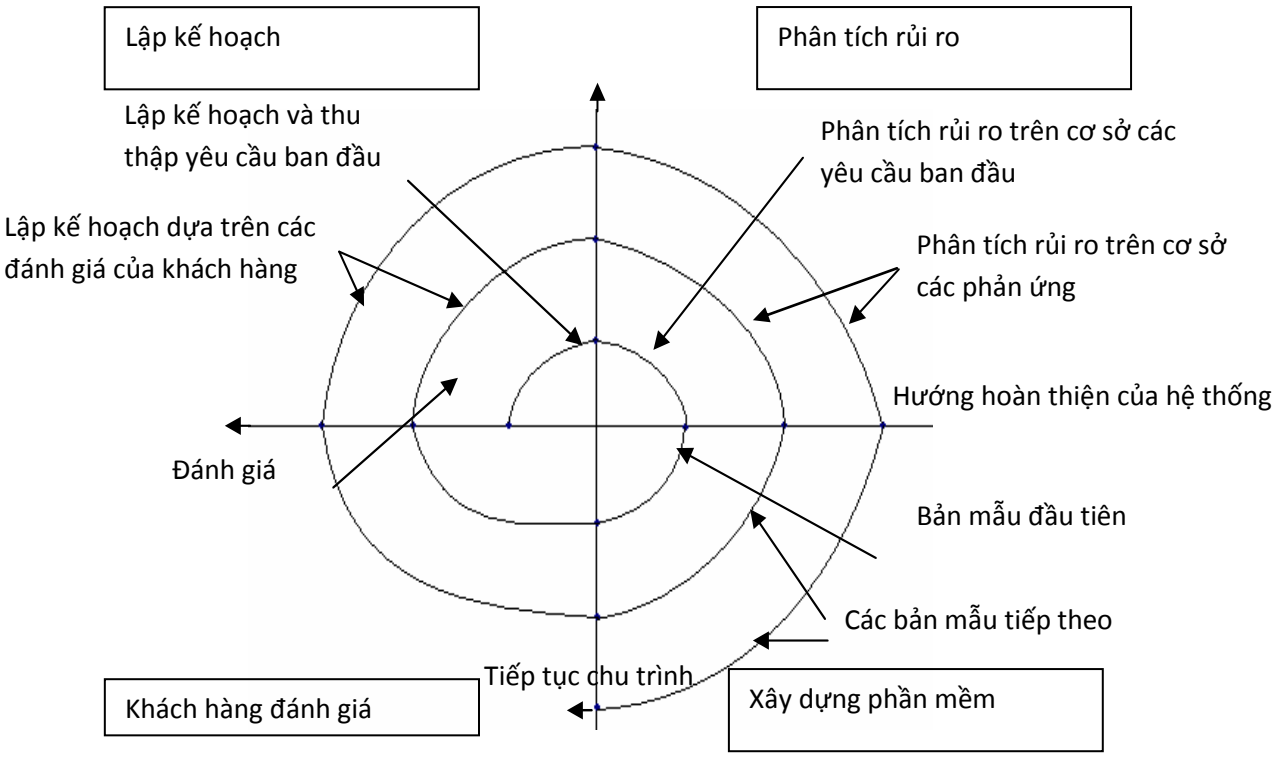
Mỗi vòng lặp trong mô hình biểu diễn một giai đoạn (pha) trong qui trình. Vòng lặp trong nhất là tính khả thi của hệ thống, vòng lặp tiếp theo là xác định các yêu cầu, vòng lặp tiếp nữa là thiết kế hệ thống….

Các rủi ro có thể được nhận rõ và được giải quyết trong suốt qui trình.

Các pha trong quy trình phát triển xoắn ốc bao gồm:

- Planning: Xác định mục tiêu, tương tác và ràng buộc.
- Risk analysis: Phân tích các lựa chọn và các chỉ định/giải quyết rủi ro.
- Engineering: Phát triển sản phẩm
- Customer evaluation: Đánh giá kết quả xây dựng.

Mô hình được tóm tắt như sau:



Nhận xét

Ưu điểm:

- Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hành nên các chức năng của hệ thống có thể nhìn thấy sớm hơn.

- Các vòng trước đóng vai trò là mẫu thử để giúp tìm hiểu thêm các yêu cầu ở những vòng tiếp theo.

- Những chức năng của hệ thống có thứ tự ưu tiên càng cao thì sẽ được kiểm thử càng kỹ.

- Mô hình xoắn ốc là một cách tiếp cận hiện thực để phát triển các phần mềm và các hệ thống lớn. Vì PM được phát triển theo sự tiến hóa, khách hàng và nhà phát triển hiểu nhau tốt hơn và rủi ro sẽ được giảm thiểu sau mỗi lần tiến hóa.

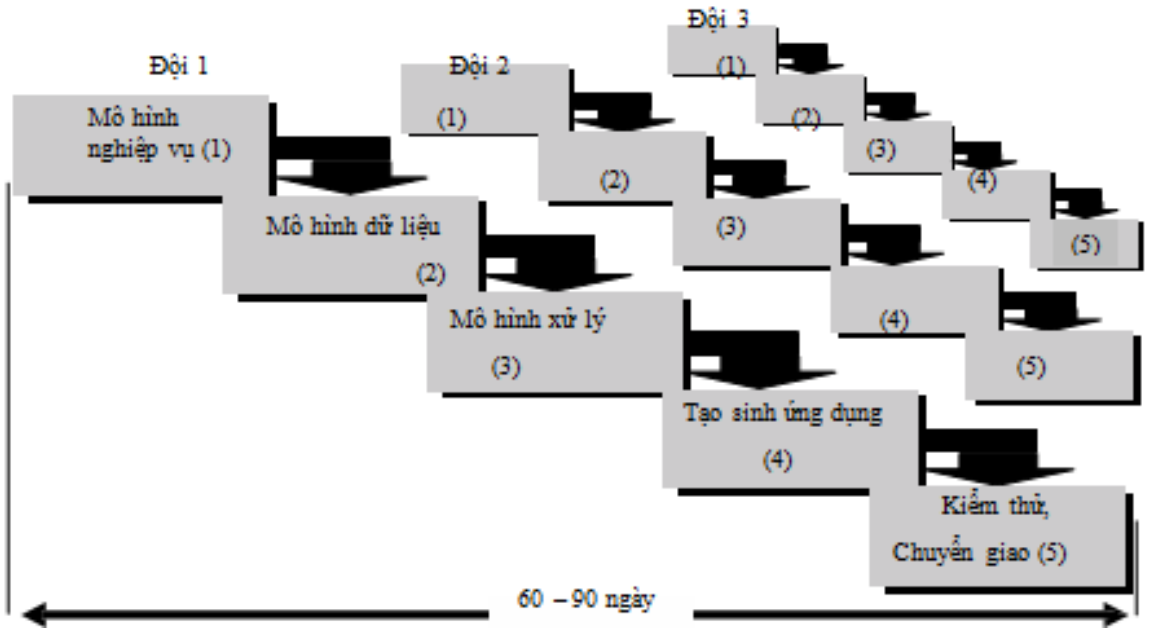
Nhược điểm :

- Mô hình này chưa thực sự quen thuộc với các nhà phát triển như trong mô hình tuyến tính.

- Hạn chế lớn nhất của mô hình này là khả năng thuyết phục khách hàng về giảm thiểu tính rủi ro trong quá trình phát triển.

2.1.5 Mô hình phát triển ứng dụng nhanh (RAD – Rapid Application Development)

Mô hình RAD được biểu diễn như hình dưới:



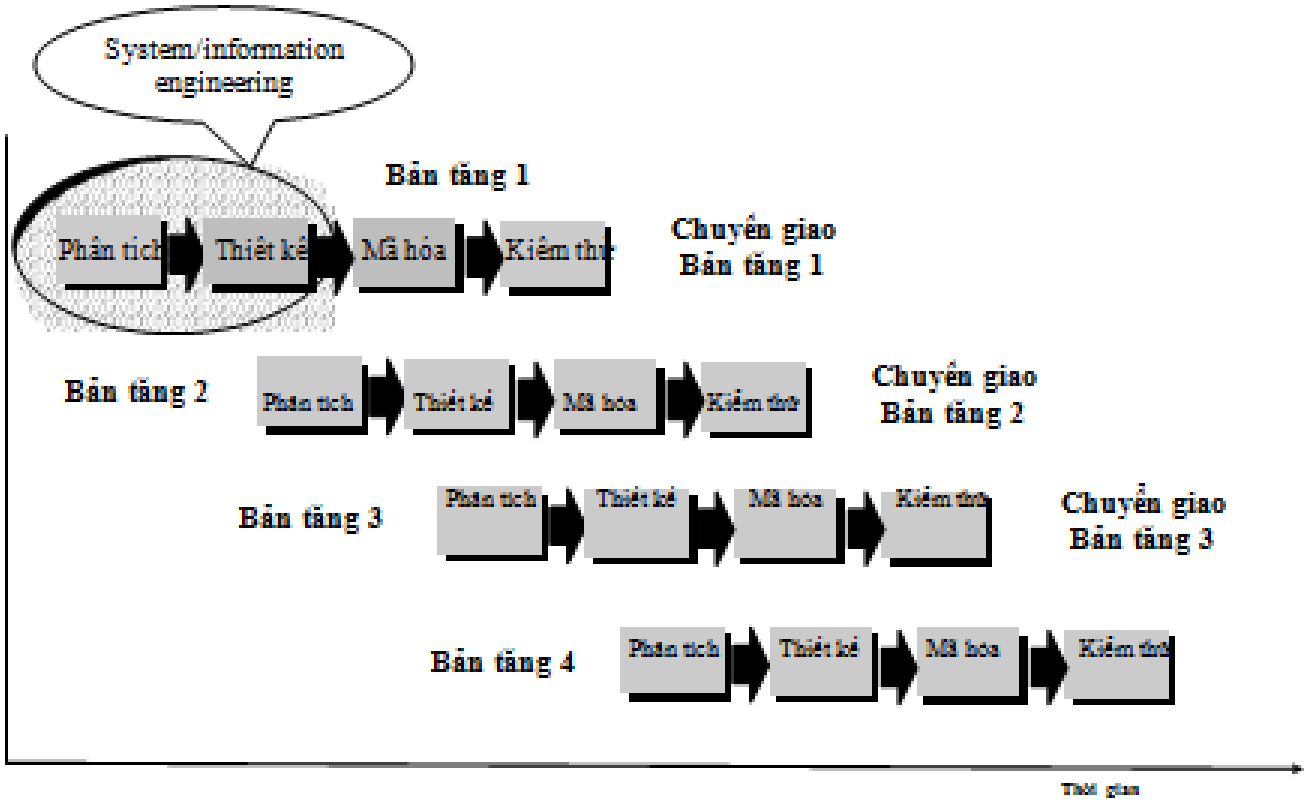
Đặc điểm của mô hình RAD:

- ✓ Hợp với những hệ thống có khả năng mô đun hóa cao: Hướng thành phần, tái sử dụng, sử dụng công cụ tự động.

- ✓ Thời gian phát triển sản phẩm nhanh: 60 – 90 ngày
- ✓ Không phù hợp với các sản phẩm: khó phân chia thành các thành phần, đòi hỏi hiệu năng cao.

2.1.6 Mô hình tăng trưởng (incremental model)

Mô hình tăng trưởng được biểu diễn như hình dưới:

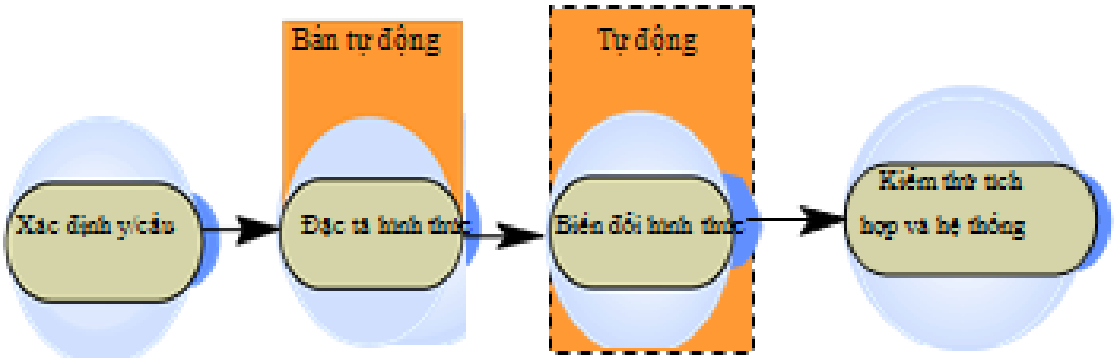


Đặc điểm của mô hình tăng trưởng

- ✓ Chuyển giao dần từng thành phần của hệ thống
  - o Sản phẩm chia thành từng lần tăng theo yêu cầu chức năng
  - o Yêu cầu người dùng ưu tiên theo thứ tự lần tăng.
- ✓ Có thể áp dụng cho các sản phẩm dùng trong thời gian ngắn
  - o Đáp ứng nhanh yêu cầu của khách hàng
  - o Chiếm lĩnh thị trường
  - o Khác với bản mẫu.
- ✓ Công ty phát triển phải có tiềm lực cao như Công nghệ, tài sản phần mềm

2.1.7 Phát triển các hệ thống hình thức hóa (Formal System Development)

Mô hình phát triển được biểu diễn như hình vẽ:

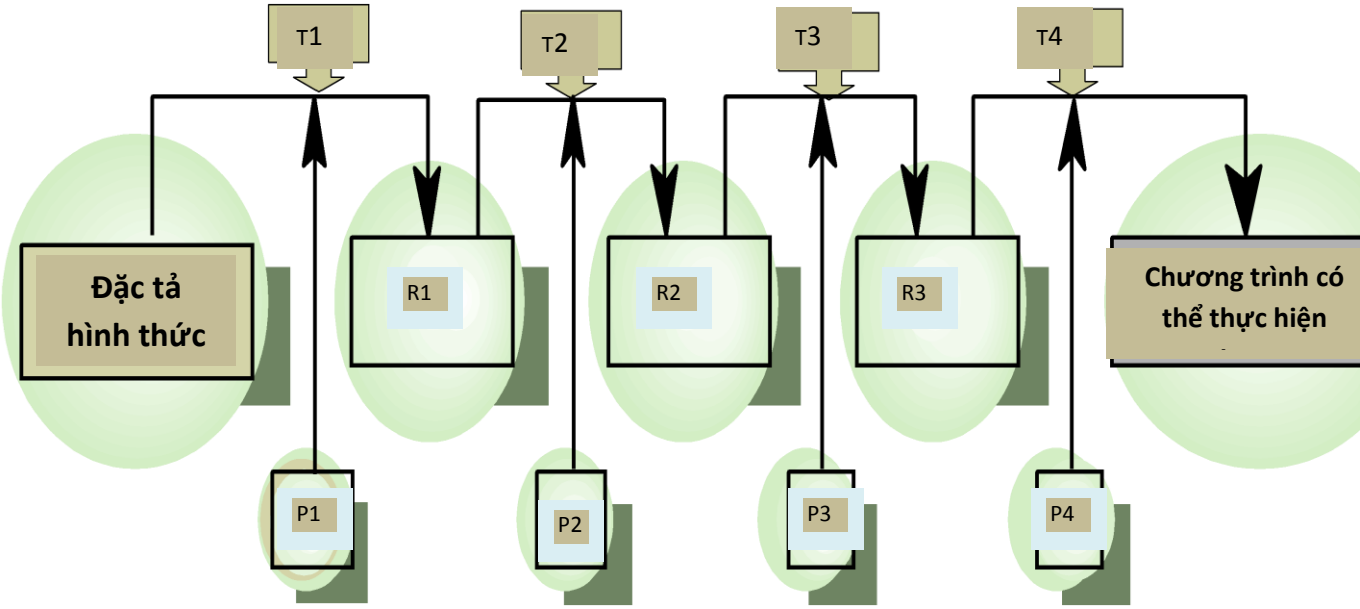


Các bước đặc trưng của tiến trình phát triển:

Đặc tả hình thức: các yêu cầu của hệ thống cần được đặc tả một cách hình thức bằng các công cụ toán học trừu tượng.

Biến đổi hình thức: quy trình biến đổi được biểu diễn như hình vẽ:

Các phép biến đổi hình thức



Các chứng minh tính đúng đắn của phép biến đổi

Hạn chế của phát triển hình thức hóa

- ✓ Cần có các kỹ năng đặc tả và sử dụng kỹ thuật tiên tiến
- ✓ Khó đặc tả được mọi khía cạnh của hệ thống, chẳng hạn như giao diện

Khả năng ứng dụng

- ✓ Những hệ thống quan trọng cần phải đảm bảo độ an toàn, bảo mật trước khi đưa vào sử dụng, xử lý nhiều, tương tác hạn chế
- ✓ Ít nhà phát triển có thể sử dụng được.

2.1.8 Phát triển hướng sử dụng lại

Phát triển hướng sử dụng lại dựa trên nền tảng của phát triển hệ thống hướng đối tượng. Ý tưởng của phát triển hệ thống hướng đối tượng như sau:

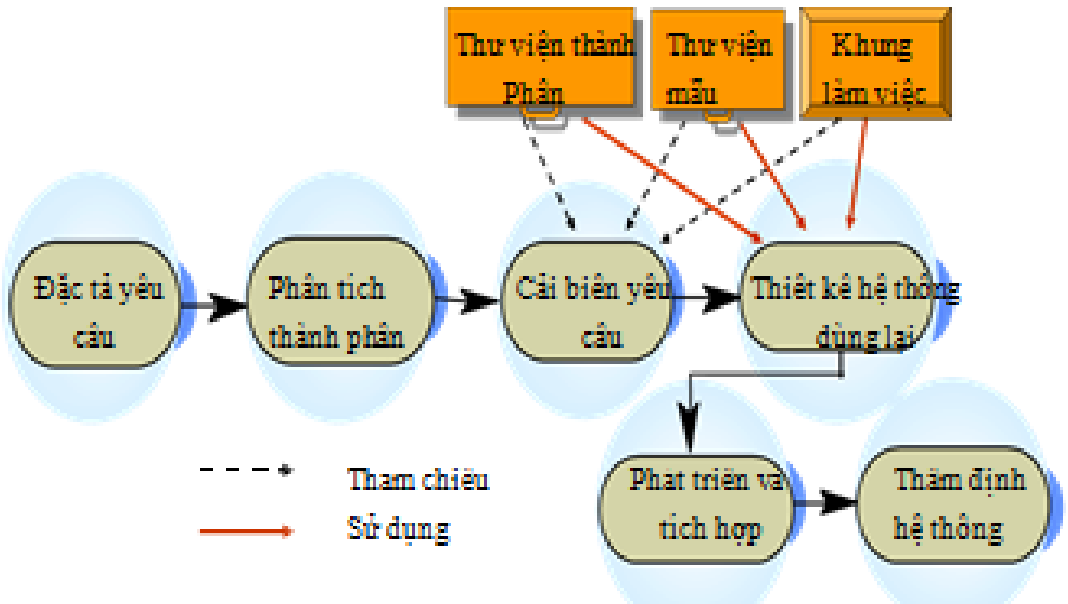
- ✓ Hệ thống được cấu thành từ các đối tượng
- ✓ Đối tượng bao gói cả dữ liệu và các xử lý trên dữ liệu đó
- ✓ Các đối tượng liên kết với nhau bằng truyền thông điệp.

Phát triển hướng đối tượng có những đặc trưng sau:

- ✓ *Hỗ trợ bao gói, che dấu thông tin*: Do bao gói cả dữ liệu và xử lý vào trong một cấu trúc là đối tượng nên đảm bảo tính độc lập tương đối giữa các chức năng, che dấu thông tin với bên ngoài.
    - o Tác động cục bộ, dễ bảo trì, dễ dùng lại
- ✓ *Tính kế thừa*:
    - o Xây dựng được các lớp cơ sở (chung),
    - o Khi cần có thể thêm các chi tiết dùng cho trường hợp cụ thể
        - ▪ Nâng cao khả năng sử dụng lại
- ✓ *Liên kết tự do, yếu*. Giúp mở rộng đơn giản, không hạn chế.

a. Tiến trình hướng sử dụng lại

Tiến trình hướng sử dụng lại được biểu diễn như hình vẽ:



Các giai đoạn của tiến trình được tiến hành tuần tự, trước tiên cần xác định và đặc tả các yêu cầu hệ thống. Phân tích thành phần nhằm phân tích hệ thống thành các

phần yêu cầu nhỏ hơn. Tiếp theo ta cần cải biên các yêu cầu theo hướng thành phần, hướng mẫu hoặc hướng khung,

=> Đây là hướng tiếp cần phát triển phần mềm quan trọng, cần kinh nghiêm, các công cụ hỗ trợ còn hạn chế.

b. Các hướng sử dụng lại

❖ *Kỹ nghệ phần mềm dựa trên cấu phần (Component Based Software Engineering - CBSE)*

Trong khoảng mười năm gần đây, cách tiếp cận phát triển phần mềm dựa trên cấu phần (CBSE) đã được nhiều người quan tâm. Cách tiếp cận này dựa trên việc tái sử dụng các thành phần một cách nhiều nhất có thể. Lắp ráp các thành phần theo đúng các yêu cầu. Khi bảo trì phần mềm, chúng ta chỉ cần thay thế các thành phần. Tiến trình thay thế này là động (tức là ta không cần dịch lại), không cần đường dẫn mà chỉ cần biết định danh của thành phần. Các thành phần là các mô đun chức năng mã đóng, chúng có kích cỡ khác nhau và mức độ trừu tượng khác nhau. Khi dùng chúng, người phát triển không cần quan tâm nó được viết bằng ngôn ngữ gì, sử dụng công nghệ gì. Đôi khi, các thành phần này là các hệ thống có bản quyền riêng của chúng (COTS hay các hệ thống thương mại off –the-shelf) mà cung cấp chức năng cụ thể. Mô hình tiến trình chung cho CBSE được chỉ ra trong hình 4.3.

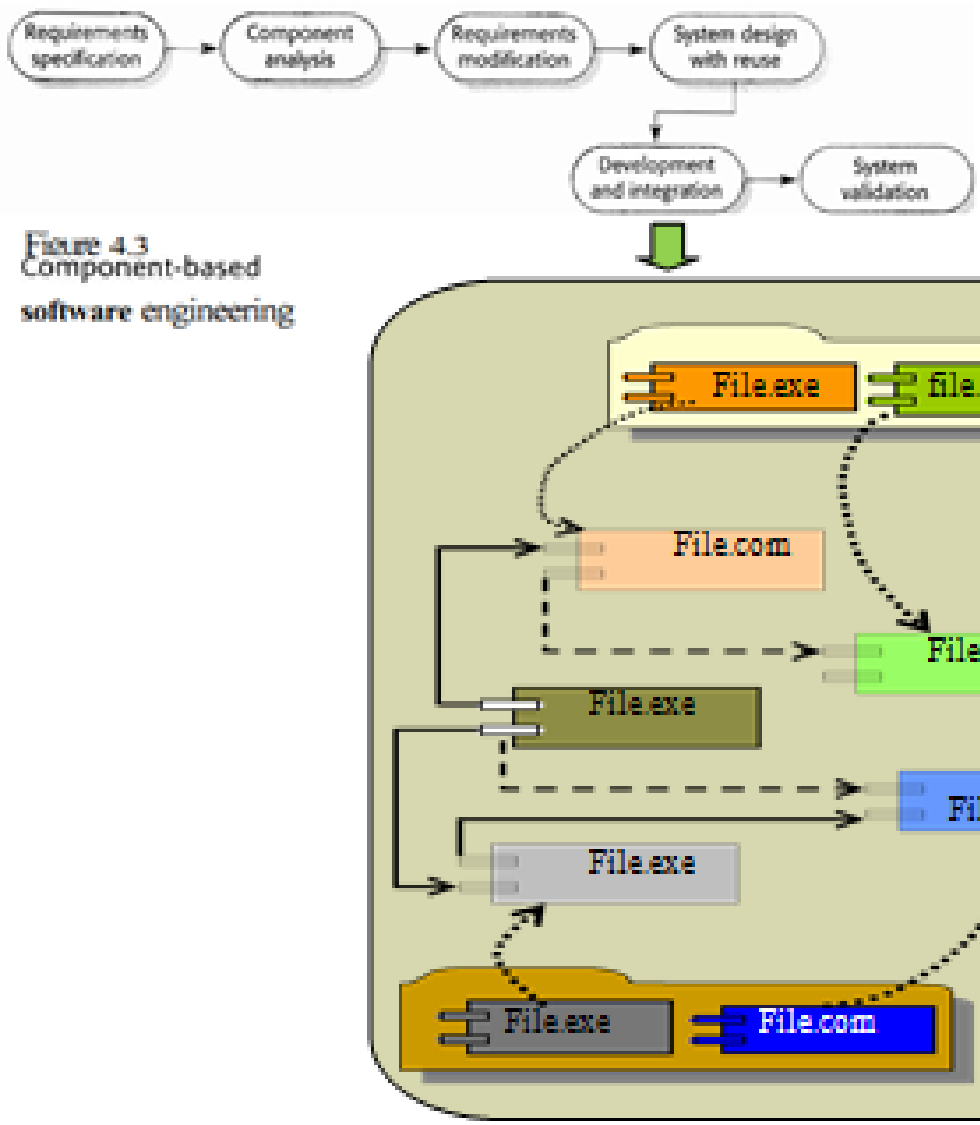


Figure 4.3
Component-based
software engineering

Giai đoạn đặc tả các yêu cầu ban đầu và giai đoạn thẩm định có thể giống với các tiến trình khác, nhưng các giai đoạn trung gian trong tiến trình hướng sử dụng lại là khác nhau. Các giai đoạn trung gian này là:

1. ***Phân tích thành phần:*** Cho trước bản đặc tả các yêu cầu, thực hiện việc tìm kiếm các thành phần mà cài đặt đặc tả này. Thông thường, không có thành phần khớp chính xác, và các thành phần được sử dụng có thể chỉ cung cấp một số chức năng đã yêu cầu.
2. ***Sửa đổi các yêu cầu:*** Trong giai đoạn này, các yêu cầu được phân tích sử dụng các thông tin về các thành phần mà đã tìm được. Sau đó chúng được sửa đổi để tương ứng với các thành phần có thể dùng. Khi việc sửa đổi là không thể, hoạt động phân tích thành phần có thể được làm lại để tìm kiếm các thành phần thay thế.
3. ***Thiết kế hệ thống bằng cách sử dụng lại:*** Trong giai đoạn này, framework của hệ thống được thiết kế hoặc một framework đang tồn tại được sử dụng lại.

Những người thiết kế nắm bắt các thành phần được sử dụng lại và tổ chức framework để phục vụ cho việc sử dụng lại. Một số phần mềm mới có thể phải được thiết kế nếu các thành phần sử dụng lại là không thể dùng.

4. ***Phát triển và tích hợp*** Phần mềm mà không thể kiếm được từ bên ngoài được phát triển, và các thành phần và các hệ thống COTS được tích hợp để tạo hệ thống mới. Tích hợp hệ thống, trong mô hình này, có thể là một phần của tiến trình phát triển hơn là một hoạt động riêng lẻ.

Kỹ nghệ phần mềm dựa trên thành phần cho thấy các lợi ích về việc giảm một lượng lớn các phần mềm được phát triển và do đó giảm chi phí và các rủi ro. Thông thường nó cũng dẫn đến phát hành phần mềm nhanh hơn. Tuy nhiên, các thương lượng yêu cầu là không thể tránh và điều này có thể dẫn đến hệ thống không thực sự thõa mãn các yêu cầu của người dùng. Hơn nữa, một số điều khiển qua tiến hóa hệ thống bị mất vì các phiên bản mới của các thành phần có thể sử dụng lại không nằm dưới sự điều khiển của tổ chức sử dụng chúng.

CBSE có nhiều điểm chung với cách tiếp cận đang mở ra để phát triển hệ thống mà dựa trên tích hợp các dịch vụ web từ một loạt các nhà cung cấp. Tham khảo thêm [1]. Mô hình này dựa trên kỹ thuật tái sử dụng một cách có hệ thống; trong đó hệ thống được tích hợp từ nhiều thành phần đang đã có. Do vậy, kiến trúc phần mềm của hệ thống dựa vào kiến trúc phần mềm của các thành phần phần mềm tiêu chuẩn nên hệ thống đạt chất lượng cao hơn.

Phương pháp phát triển dựa trên thành phần gần tương tự như phương pháp phát triển hướng đối tượng. Hoạt động công nghệ bắt đầu với sự chỉ ra các lớp tham dự để phát triển hệ thống. Nếu các lớp này được tìm thấy trong thư viện và sự thích nghi là tốt, chúng sẽ được lấy ra và phát triển hệ thống. Ngược lại, chúng sẽ được phát triển để sử dụng và bổ sung vào thư viện sử dụng lại.

Thành phần phần mềm được sử dụng lại có độ chính xác cao và có thể nói là không chứa lỗi. Mặc dầu không thường xuyên được chứng minh về mặt hình thức nhưng với việc sử dụng lại, lỗi được tìm thấy và được loại trừ; chất lượng của thành phần được cải thiện là một tất yếu. Phần mềm được phát triển theo phương pháp này là nhanh, ổn định và hiệu quả.Khi những thành phần sử dụng lại được ứng dụng thông qua tiến trình phần mềm, chúng ta ít tốn thời gian để tạo ra kế hoạch, mô hình, tài liệu, mã và dữ liệu mà chúng là cần thiết để tạo ra hệ thống. Tuy nhiên để triển khai trong thực tế, chúng ta cần các thành phần được mô tả, chúng ta cần hiểu về nó và cần có phương pháp tìm kiếm thành phần hiệu quả.

❖ *Phân tích và thiết kế hướng mẫu (Pattern Oriented Analysis & Design - POAD)*

Phương pháp phân tích và thiết kế hướng mẫu đang bắt đầu được sử dụng rộng rãi. Tuy nhiên để có thể sử dụng được nó, chúng ta cần có khả năng phân tích tốt và có hiểu biết, thành thạo về mẫu. Các bước cần thực hiện như sau:

- ✓ Phân tích yêu cầu hướng theo mẫu
- ✓ Xem xét các mẫu đã có trong thư viện
- ✓ Tìm và lựa chọn mẫu thích hợp với các yêu cầu đã được phân tích
- ✓ Chuyển thiết kế thành chương trình (hoạt động này có thể làm tự động hoặc bán tự động)

❖ *Phát triển khung làm việc (Frame Development)*

Để phát triển phần mềm theo phương phát này, chúng ta cần trải qua các hoạt động như sau:

- ✓ Xây dựng khung làm việc
  - o Xác định lớp bài toán cần giải quyết
  - o Xây dựng khung bài toán (dựa trên các mẫu)
  - o Làm sẵn các tiêu bản mẫu (dùng ngay được).
- ✓ Triển khai
  - o Phân tích bài toán theo khung
  - o Xác định tiêu bản thích hợp
  - o Lắp ghép hay tìm phần thay thế

Việc lựa chọn mô hình phát triển phần mềm phụ thuộc vào bài toán và môi trường cụ thể. Nếu những yêu cầu của hệ thống là rõ ràng, mô hình thác nước là thích hợp nhất. Nếu hệ thống là phức tạp và hướng điều khiển, nên sử dụng mô hình hướng sử dụng lại. Nếu các yêu cầu chưa rõ ràng, độ phức tạp cao, có khả năng thay đổi, không chắc chắn về tính hiệu quả, tính khả thi nên sử dụng mô hình làm bản mẫu, mô hình xoắn ốc....

2.2 Các hoạt động trong quy trình phần mềm

Trong quy trình phần mềm gồm 4 hoạt động cơ bản. Những hoạt động này bao gồm:

- **Đặc tả phần mềm:** các chức năng của hệ thống và những ràng buộc khi vận hành hệ thống cần phải được xác định một cách đầy đủ và chi tiết.

- **Thiết kế và cài đặt phần mềm**: phần mềm được xây dựng phải thoả mãn đặc tả của nó.

- **Đánh giá phần mềm**: phần mềm phải được đánh giá và thẩm định để đảm bảo rằng nó đã thoả mãn tất cả các yêu cầu.

- **Cải tiến phần mềm**: phần mềm cần phải cải tiến và điều chỉnh để phù hợp với những thay đổi về yêu cầu hệ thống.

Với mỗi mô hình khác nhau thì các hoạt động này cũng được tổ chức theo các cách khác nhau. Ví dụ, trong mô hình thác nước, chúng được tổ chức một cách tuần tự. Trong mô hình đài phun nước, các hoạt động này có thể gối lên nhau. Trong các phần tiếp sau đây, chúng ta sẽ nghiên cứu cụ thể từng hoạt động.

2.2.1 Đặc tả phần mềm

Đặc tả phần mềm (hay còn gọi là kỹ thuật xác định yêu cầu) là quy trình tìm hiểu và định nghĩa những dịch vụ nào được khách hàng yêu cầu và các ràng buộc trong quá trình vận hành và xây dựng hệ thống.

Quy trình xác định yêu cầu bao gồm bốn pha chính:

- ***Nghiên cứu tính khả thi:*** Nghiên cứu tính khả thi giúp xác định những yêu cầu của người sử dụng có thoả mãn những công nghệ hiện tại hay không. Về góc độ kinh doanh, nghiên cứu khả thi nhằm xác định hệ thống đưa ra có mang lại lợi nhuận không. Việc nghiên cứu khả thi nên được thực hiện một cách nhanh chóng và không quá tốn kém. Kết quả của việc nghiên cứu khả thi sẽ xác định có nên tiếp tục xây dựng hệ thống nữa hay không.

- ***Phân tích và rút ra các yêu cầu:*** đây là quy trình đưa ra các yêu cầu hệ thống thông qua một số phương pháp như: quan sát hệ thống hiện tại, phỏng vấn và thảo luận với người sử dụng, phân tích nhiệm vụ, phân tích tài liệu hoặc hệ thống cũ … Trong pha này, chúng ta có thể phải xây dựng một hoặc nhiều mô hình hệ thống và các mẫu thử.

- ***Đặc tả yêu cầu***: Pha này sẽ tư liệu hoá những thông tin thu thập được. Có hai loại yêu cầu cần được xác định:

*** Yêu cầu của người sử dụng**: là những yêu cầu bằng ngôn ngữ tự nhiên. Kiểu yêu cầu này được viết bởi người sử dụng.

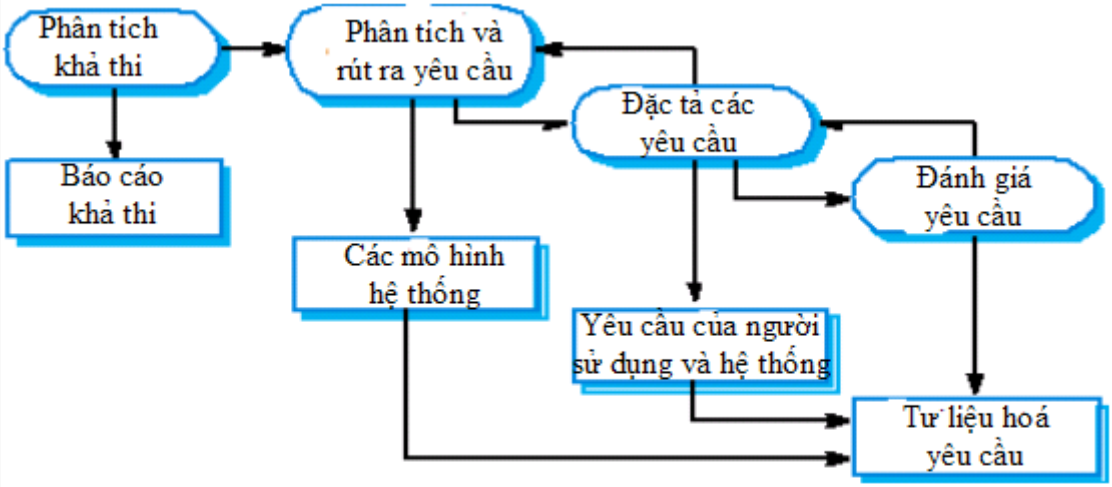
* **Yêu cầu hệ thống:** là những tài liệu có cấu trúc, đươc mô hình hoá, mô tả chi tiết về các chức năng, dịch vụ và các ràng buộc vận hành của hệ thống. Yêu cầu hệ thống sẽ định nghĩa những gì cần phải xây dựng, cho nên nó có thể trở thành bản hợp đồng giữa khách hàng và nhà thầu. Các yêu cầu hệ thống được chia làm 2 loại:

+ Các yêu cầu hệ thống chức năng: Là các dịch vụ mà hệ thống phải cung cấp

+ Các yêu cầu phi chức năng: Là các ràng buộc mà hệ thống phải tuân theo

- ***Đánh giá yêu cầu***: pha này sẽ kiểm tra lại các yêu cầu xem chúng có đúng thực tế hay không, có thống nhất không, có đầy đủ không. Nếu phát hiện ra lỗi thì ta

phải chỉnh sửa các lỗi này.



Hình 2.6: Quy trình xác định yêu cầu

2.2.2 Thiết kế phần mềm và cài đặt

a) Thiết kế phần mềm:

Là quá trình thiết kế cấu trúc phần mềm dựa trên những tài liệu đặc tả. Hoạt động thiết kế bao gồm những công việc chính sau:

- ***Thiết kế kiến trúc:*** Thiết kế các hệ thống con cấu thành lên hệ thống cần xây dựng và mối quan hệ giữa chúng được xác định và tư liệu hoá.

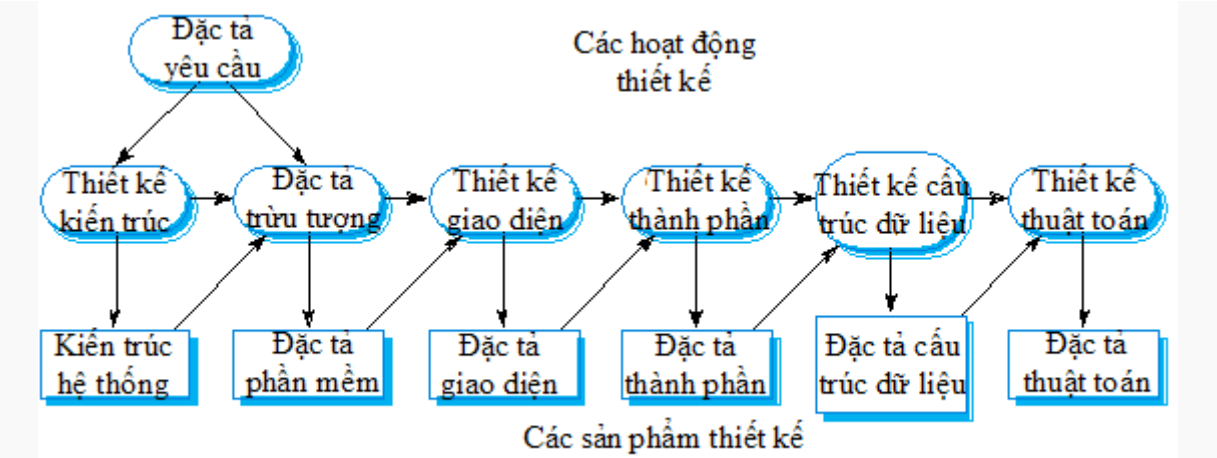
- ***Đặc tả trừu tượng:*** với mỗi hệ thống con, phải có một bản đặc tả về các dịch vụ của nó và những ràng buộc khi nó vận hành.

- ***Thiết kế giao diện***: với mỗi hệ thống con, các giao diện của nó với những hệ thống con khác phải được thiết kế và tư liệu hoá.

- ***Thiết kế thành phần:*** các dịch vụ cung cấp cho các thành phần khác và các giao diện tương tác với chúng phải được thiết kế.

- ***Thiết kế cấu trúc dữ liệu (thiết kế dữ liệu)***: cấu trúc dữ liệu được sử dụng để cài đặt hệ thống phải được thiết kế một cách chi tiết và cụ thể.

- ***Thiết kế thuật toán:*** Các thuật toán được sử dụng để cung cấp các dịch vụ phải được thiết kế chi tiết và chính xác.



Hình 2.7: Mô hình chung của quy trình thiết kế

b) Cài đặt phần mềm:

Là quy trình chuyển đổi từ tài liệu đặc tả hệ thống thành một hệ thống thực, có thể vận hành được và phải loại bỏ các lỗi của chương trình.

Lập trình là một hành động cá nhân, không có quy trình lập trình chung. Người lập trình phải thực hiện một số kiểm thử để phát hiện ra lỗi trong chương trình và loại bỏ nó trong quy trình gỡ lỗi.

Nhận xét:

Trong ba giai đoạn: thiết kế, cài đặt và bảo trì thì thiết kế là giai đoạn quan trọng nhất, chịu trách nhiệm đến 80% đối với sự thành công của một sản phẩm. Cài đặt là việc thực thi những gì đã thiết kế. Nếu trong quá trình cài đặt có xuất hiện vấn đề thì phải quay lại sửa bản thiết kế. Quá trình thiết kế tốt là cơ sở để quản lý và giảm chi phí cho công việc bảo trì phần mềm sau này.

2.2.3 Đánh giá phần mềm

Đánh giá phần mềm hay còn gọi là thẩm tra và đánh giá (V&V - Verification and validation) được sử dụng để chỉ ra rằng hệ thống đã thực hiện theo đúng các đặc tả và thoả mãn mọi yêu cầu của khách hàng.

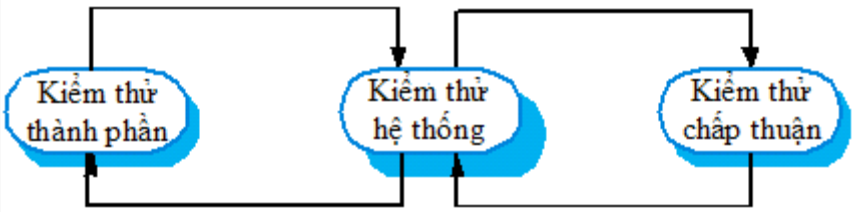
Đánh giá phần mềm bao gồm các công đoạn: kiểm tra, xem xét lại, và kiểm thử hệ thống. Kiểm thử hệ thống tức là cho hệ thống thực hiện trên những trường hợp có dữ liệu thật được lấy từ tài liệu đặc tả hệ thống. Quy trình kiểm thử gồm các pha sau:

- ***Kiểm thử thành phần (đơn vị)***: các thành phần được kiểm thử một cách độc lập, thành phần có thể là một chức năng hoặc một đối tượng hoặc một nhóm các thực thể gắn kết với nhau.

- ***Kiểm thử hệ thống***: kiểm thử toàn bộ hệ thống.

- ***Kiểm thử chấp thuận***: kiểm thử trên dữ liệu của khách hàng để kiểm tra hệ

thống có đáp ứng tất cả các yêu cầu của khách hàng hay không.

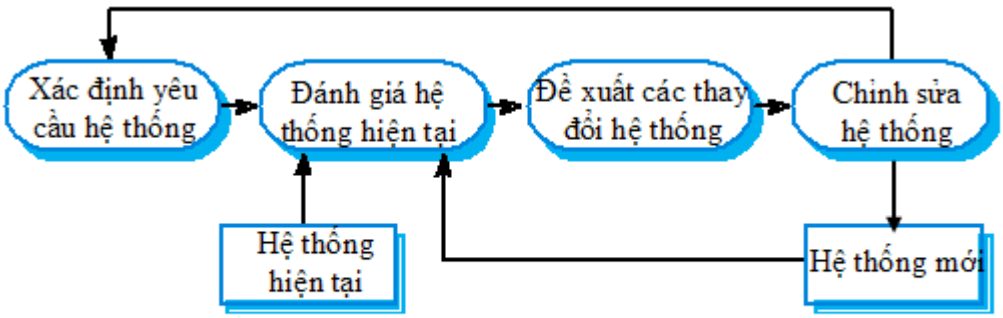


Hình 2.8: Quy trình kiểm thử

Khi chuyển giao hệ thống cho khách hàng thì quy trình kiểm thử beta sẽ được thực hiện. Khách hàng sẽ thông báo các lỗi cho đội dự án. Những lỗi này sẽ được chỉnh sửa và tiếp tục kiểm thử beta hoặc chuyển giao thực sự cho khách hàng.

2.2.4 Cải tiến phần mềm

Khi các yêu cầu hệ thống thay đổi theo sự thay đổi của các yêu cầu nghiệp vụ thì phần mềm phải cải tiến và thay đổi để hỗ trợ khách hàng. Thông thường chi phí để bảo trì và cải tiến thường đắt hơn nhiều so với chi phí xây dựng phần mềm.



Hình 2.9: Cải tiến hệ thống

2.3 Các vấn đề liên quan đến tiến trình phần mềm

- ✓ Xác định yêu cầu và thiết kế có vai trò quyết định đến chất lượng phần mềm, chiếm phần lớn công sức so với xây dựng.
- ✓ Khi chuyển tiếp giữa các pha phát triển phải thẩm định tốt để đảm bảo các lỗi không ảnh hưởng đến pha sau.
- ✓ Tài liệu tạo ra ở mỗi pha không chỉ dùng cho pha kế tiếp mà còn dùng để đảm bảo chất lượng của phần mềm và dùng trong pha bảo trì.
- ✓ Cần chuẩn hóa mẫu biểu, cách thức ghi chép, tạo tài liệu nhằm đảm bảo chất lượng phần mềm.

Quan hệ giữa tiến trình và sản phẩm: Tiến trình và sản phẩm là hai mặt của phát triển. Tiến trình tốt, đảm bảo các ràng buộc về mặt sản phẩm. Sản phẩm tốt là sự tổng hợp của nhiều yếu tố:

- ✓ Tiến trình thích hợp
- ✓ Đội ngũ chuyên môn tốt
- ✓ Công cụ trợ giúp mạnh
- ✓ Năng lực/độ thuần thục quản lý tiến trình của tổ chức cao (CMM)

CHƯƠNG 3. ĐẶC TẢ PHẦN MỀM

MỤC ĐÍCH

- *Hiểu được tầm quan trọng của việc đặc tả chính xác phần mềm*
- *Nắm được qui trình xác định yêu cầu*

Đặc tả là gì?

Theo IBM Dictionary of Computing (1994): Đặc tả là một dạng thức văn bản chi tiết cung cấp các mô tả xác định về một hệ thống nhằm để phát triển hay hợp thức hóa. Trong lĩnh vực phát triển hệ thống, đặc tả là mô tả cách thiết kế, cách bố trí thiết bị và cách xây dựng chương trình cho hệ thống.

Định nghĩa: Đặc tả một vấn đề là mô tả (một cách rất riêng nhờ các kỹ thuật thể hiện) các đặc trưng của vấn đề đó. Vấn đề có thể là đối tượng, khái niệm hoặc một thủ tục nào đó….

Như vậy, đặc tả là sự mô tả các đặc trưng nhằm diễn đạt các yêu cầu và các chức năng của một sản phẩm phần mềm cần thiết kế.

Đặc tả có các đặc trưng:

- Tính chính xác (Correctness)
- Tính trừu tượng (Abstraction): Càng ở mức cao đặc tả càng trừu tượng. Càng xuống các mức thấp, đặc tả càng tiếp cận dần tới cụ thể - tức là tới một thể hiện trên máy tính với một ngôn ngữ lập trình cụ thể.
- Tính chặt chẽ về mặt toán học (Rigorousness)

Các phương pháp đặc tả:

- **Đặc tả hình thức:** Là những đặc tả chính xác (không thể dẫn tới những cách hiểu khác nhau), được diễn đạt bằng ngôn ngữ đại số và logic toán học, rất chặt chẽ và không nhập nhằng.
- **Đặc tả phi hình thức:** Được diễn đạt bằng ngôn ngữ tự nhiện và toán học, Tuy không chặt chẽ nhưng dễ hiểu và dễ diễn đạt. Thường sử dụng khi cần phát biểu các bài toán, các yêu cầu ban đầu, hoặc để chính xác hóa những điểm chưa rõ, những khái niệm mơ hồ.
- **Đặc tả hỗn hợp:** Phối hợp giữa 2 phương pháp trên. Thường mô tả phi hình thức để giải thích rõ hơn, dễ hiểu hơn một khi mô tả hình thức quá phức tạp.

Các loại hình đặc tả (trong thực tế có nhiều loại hình đặc tả, điển hình gồm các loại):

- Đặc tả cấu trúc dữ liệu (Mô tả các thành phần của dữ liệu…)
- Đặc tả chức năng (mô tả chức năng thông qua việc mô tả tính chất của đầu vào, đầu ra…)
- Đặc tả đối tượng (bao gồm đặc tả cấu trúc dữ liệu và đặc tả chức năng…)
- Đặc tả thao tác (mô tả các thao tác cần thực hiện…)
- Đặc tả cú pháp (mô tả cách lắp ghép các ký hiệu, các từ thành chương trình)
- Đặc tả xử lý (mô tả quy trình bằng sơ đồ, ngôn ngữ có cấu trúc hoặc ngôn ngữ lập trình)
- Đặc tả thuật toán (mô tả cách giải bằng sơ đồ khối, ngôn ngữ có cấu trúc hoặc ngôn ngữ lập trình)

Đặc tả với lập trình: Trong những trường hợp có thể, người ta thường hướng đặc tả về một ngôn ngữ lập trình nào đó.

Phân tích và đặc tả yêu cầu là bước đầu tiên trong tiến trình kỹ nghệ phần mềm. Mục tiêu của hoạt động này là nhằm xác định tất cả các yêu cầu đặt ra đối với hệ thống phần mềm cần xây dựng (trả lời câu hỏi What). Đầu ra của hoạt động này là tài liệu đặc tả yêu cầu của hệ thống phần mềm. Hoạt động phân tích và đặc tả yêu cầu phần mềm còn được gọi là tiến trình kỹ nghệ yêu cầu. Tiến trình này thiết lập nên các dịch vụ mà khách hàng yêu cầu hệ thống và các ràng buộc về việc việc vận hành và phát triển hệ thống

3.1 Yêu cầu phần mềm là gì?

Yêu cầu là một phát biểu mô tả về dịch vụ mà hệ thống cung cấp cũng như sự ràng buộc lên sự phát triển và hoạt động của nó hoặc một mục tiêu mà hệ thống phần mềm phải đạt được. Một yêu cầu có thể được đặc tả bằng các công cụ khác nhau như ngôn ngữ tự nhiên, ngôn ngữ tự nhiên có cấu trúc, ngôn ngữ mô hình, ngôn ngữ hình thức...

Ví dụ: Xét phần mềm quản lý hồ sơ sinh viên của một trường đại học. Sau đây là một yêu cầu được phát biểu bằng ngôn ngữ tự nhiên về một dịch vụ mà phần mềm phải cung cấp:

REQ1: "Phần mềm phải lưu trữ được mọi thông tin cần quản lý của sinh viên từ khi nhập học cho đến khi tốt nghiệp ra trường được 10 năm".

Một yêu cầu có thể giới hạn từ một phát biểu ở mức độ trừu tượng cao về một dịch vụ hoặc một ràng buộc hệ thống đến một đặc tả chức năng toán học chi tiết. Nó là cơ sở cho việc mời thầu – do đó nó phải dễ hiểu. Nó cũng có thể là cơ sở để ký kết hợp đồng đấu thầu – do đó phải được định nghĩa chi tiết (đủ chi tiết để nghiệm thu). Các yêu cầu cũng sẽ là tài liệu đầu vào cho hoạt động thiết kế và triển khai do đó, chúng cần xác định đầy đủ, chính xác và không mẫu thuẫn với nhau. Các yêu cầu cần được phát biểu ở các mức độ trừu tượng khác nhau, từ đây ta có hai kiểu yêu cầu:

- *Các yêu cầu hệ thống (System requirements)*
- *Các yêu cầu của người dùng (User requirements)*

3.2 Yêu cầu hệ thống

3.2.1 Phân loại yêu cầu hệ thống phần mềm

Các yêu cầu của hệ thống phần mềm thường được chia thành ba loại:

+ Yêu cầu chức năng,

+ Yêu cầu phi chức năng

+ Yêu cầu miền ứng dụng.

Tuy nhiên, trong thực tế chúng ta rất khó phân biết ba loại yêu cầu này một cách rõ ràng.

Trong chương này, chúng ta sẽ sử dụng một ví dụ về hệ thống thư viện để xác định các loại yêu cầu: *Hệ thống thư viện (LIBSYS) cung cấp một giao diện đơn giản để lưu CSDL về các bài báo trên các thư viện khác nhau. Người sử dụng có thể tìm kiếm, download và in những tài liệu này.*

3.2.2 Yêu cầu chức năng

Các yêu cầu chức năng là các phát biểu về các dịch vụ mà hệ thống sẽ cung cấp, cách thức hệ thống nên tương tác với các đầu vào đặc biệt và cách thức hệ thống ứng xử với các tình huống đặc biệt. Chúng mô tả hệ thống sẽ làm những gì. Nó mô tả các chức năng hoặc các dịch vụ của hệ thống một cách chi tiết.

Ví dụ: Một số yêu cầu chức năng của LYBSYS

*REQ3: "Người sử dụng có thể **tìm kiếm** tất cả các thông tin trong CSDL hoặc trong một tập con của CSDL."*

*REQ4: "Hệ thống sẽ cung cấp các chức năng để người sử dụng **xem(đọc)** tài liệu, **download, in** tài liệu,…"*

*REQ5: "Tất cả những hoá đơn mà người sử dụng **đăng ký** để **in sao** tài liệu có một mã duy nhất."*

3.2.3 Yêu cầu phi chức năng

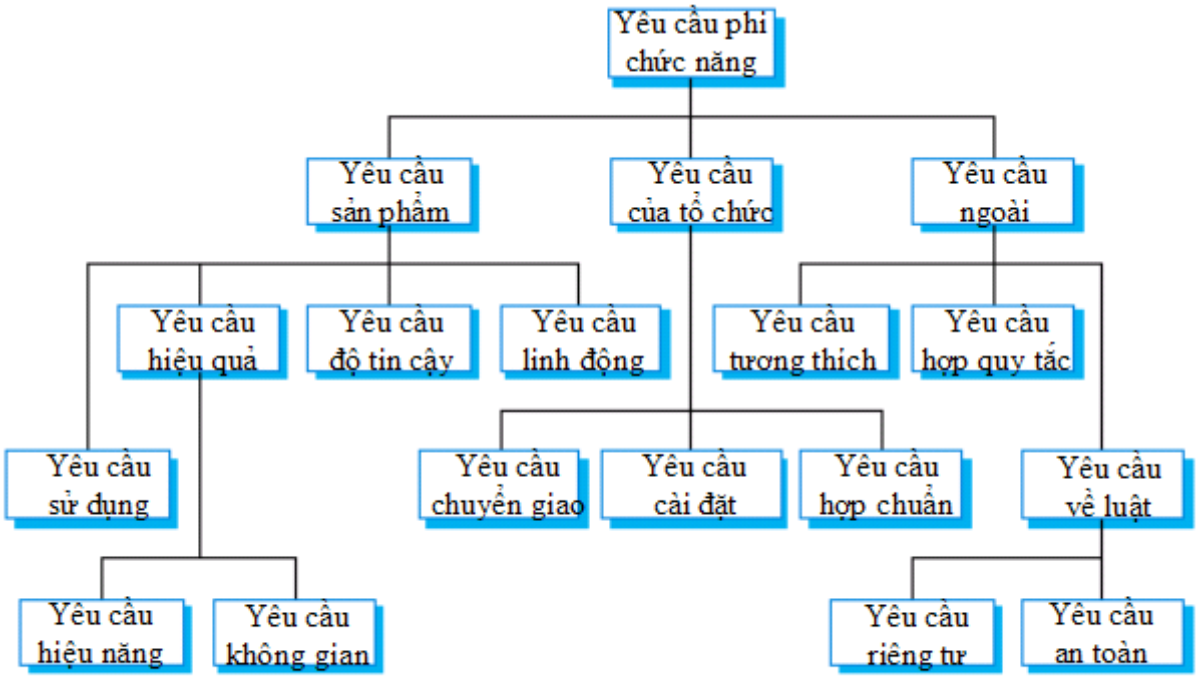
Yêu cầu phi chức năng không đề cập trực tiếp tới các chức năng cụ thể của hệ thống. Yêu cầu phi chức năng thường định nghĩa các thuộc tính như: độ tin cậy, thời gian đáp ứng, các yêu cầu về lưu trữ …và các ràng buộc của hệ thống như: khả năng của thiết bị vào/ra, giao diện … Một số yêu cầu phi chức năng còn có liên quan đến quy trình xây dựng hệ thống. Ví dụ: các chuẩn được sử dụng, các công cụ CASE, ngôn ngữ lập trình …

Các yêu cầu phi chức năng xuất hiện là do yêu cầu của người sử dụng, ràng buộc về ngân sách, các chính sách của tổ chức sử dụng hệ thống, yêu cầu tương thích giữa phần cứng và phần mềm và các tác nhân ngoài khác. Do đó, chúng ta có thể phân loại các yêu cầu phi chức năng như sau:

- **Các yêu cầu về sản phẩm như:** hiệu năng, khả năng sử dụng, độ tin cậy …

- **Các yêu cầu về tổ chức**: các yêu cầu này được lấy từ những chính sách và quy tắc của khách hàng hoặc tổ chức sử dụng hệ thống.

- **Các yêu cầu ngoài**: được xác định từ các tác nhân ngoài của hệ thống.



Hình 4.1: Phân loại các yêu cầu phi chức năng

Nhận xét: Các yêu cầu phi chức năng có thể làm hạn chế hơn những yêu cầu chức năng. Nhưng nếu nó không được thoả mãn thì hệ thống sẽ không sử dụng được hoặc không tiện dụng

Ví dụ: Xác định các yêu cầu phi chức năng của LIBSYS

- Yêu cầu về sản phẩm: LIBSYS phải được cài đặt bằng HTML mà không có frame hoặc Java applets.

- Yêu cầu về mặt tổ chức: Quy trình xây dựng hệ thống và các tài liệu chuyển giao phải thoả mãn các quy tắc đã được định nghĩa trong XYZCo-SP-STAN-95.

- Yêu cầu ngoài: Hệ thống không được để lộ các thông tin cá nhân của khách hàng.

Nói chung, chúng ta rất khó xác định chính xác và rất khó thẩm tra những ***yêu cầu phi chức năng mập mờ***. Do đó, trong tài liệu đặc tả yêu cầu, người ta thường bổ sung các mục tiêu. Mục tiêu rất hữu ích đối với người phát triển hệ thống khi nó truyền tải được những mong muốn của người sử dụng hệ thống. Còn với những ***yêu cầu phi chức năng có thể thẩm định được*** là những yêu cầu có thể kiểm thử một cách khách quan.

Ở đây chúng ta cũng cần phân biệt giữa mục tiêu và yêu cầu của hệ thống. Mục tiêu/mong muốn là cái mà chúng ta cần hướng đến. Còn yêu cầu là cái cụ thể, ta có thể kiểm tra được.

Để thẩm định các yêu cầu phi chức năng một cách khách quan, bất cứ khi nào có thể chúng ta nên phát biểu các yêu cầu phi chức năng bằng một phát biểu có định lượng (xác định đơn vị đo và điểm chuẩn).

Ví dụ:

- Yêu cầu về mặt tốc độ thực thi:

+ Tốc độ thực thi có thể tính qua số lượng giao dịch được xử lý/một giây.

+ Điểm chuẩn: ≥ 50 giao dịch/giây => Đáp ứng yêu cầu. <50 giao dịch/giây => không đạt.

- Yêu cầu về mặt kích cỡ: Tính theo KB, MB, ...hoặc số chip RAM

- Yêu cầu về tính dễ sử dụng: Có thể tính qua thời gian đào tạo và số lỗi mắc phải sau khi đào tạo người sử dụng. Yêu cầu này có thể phát biểu như sau:

REQ9: "Những người sử dụng có kinh nghiệm có thể sử dụng được tất cả các chức năng của hệ thống chỉ sau hai tiếng tập huấn. Sau khoá huấn luyện này, số lỗi chương trình gây ra bởi người sử dụng là không quá hai lỗi một ngày."

Tuy nhiên, trong nhiều trường hợp thường xảy ra xung đột giữa các yêu cầu

phi chức năng đối với những hệ thống phức tạp.

3.2.4 Yêu cầu miền ứng dụng

Yêu cầu miền ứng dụng được xác định từ miền ứng dụng của hệ thống và phản ánh các thuộc tính và ràng buộc của miền ứng dụng. Nó có thể là yêu cầu chức năng hoặc phi chức năng.

Nếu yêu cầu miền ứng dụng không được thoả mãn thì có thể hệ thống sẽ không làm việc được,hoặc không phục vụ đúng mục đích ứng dụng.

Một số vấn đề liên quan đến yêu cầu miền ứng dụng:

+ **Khả năng có thể hiểu được**: các yêu cầu được biểu diễn dưới ngôn ngữ của lĩnh vực ứng dụng.

+ Không được ẩn ý: Các chuyên gia có hiểu biết về lĩnh vực của họ nhưng họ không biết cách xây dựng những yêu cầu miền ứng dụng một cách rõ ràng, mang tính kỹ thuật.

Một số yêu cầu miền ứng dụng của LIBSYS:

REQ10: "Nên có một giao diện người dùng chuẩn cho mọi cơ sở dữ liệu dựa trên chuẩn Z39.50".

REQ11: "Vì các hạn chế bản quyền, một số tài liệu phải được xóa ngay sau khi đến. Phụ thuộc vào các yêu cầu người dùng, các tài liệu này hoặc được in trên máy in người dùng hoặc trên máy chủ hệ thống và được chuyển đến người dùng.

3.3 Yêu cầu của người sử dụng

Yêu cầu của người sử dụng là những yêu cầu được phát biểu bằng ngôn ngữ tự nhiên bởi người sử dụng, thể hiện ở mức độ trừu tượng những công việc mà người sử dụng muốn hệ thống phải đáp ứng.

Tài liệu mô tả yêu cầu của người sử dụng nên nói rõ những yêu cầu chức năng và phi chức năng để người sử dụng có thể hiểu được chúng mà không cần phải có những kiến thức về công nghệ một cách chi tiết, được định nghĩa bằng cách sử dụng ngôn ngữ tự nhiên, bảng hoặc biểu đồ đơn giản. Tuy nhiên, chúng ta sẽ gặp phải một số khó khăn khi sử dụng ngôn ngữ tự nhiên:

- *Không rõ ràng*: Tính chính xác rất khó đạt được nếu tài liệu khó đọc.

- *Yêu cầu lộn xộn*: các yêu cầu chức năng và phi chức năng không rõ ràng.

- *Lẫn lộn giữa các yêu cầu*: các yêu cầu khác nhau có thể được diễn tả cùng với nhau.

Do đó, để viết yêu cầu của người sử dụng ta nên áp dụng một số quy tắc sau:

- Đưa ra một định dạng chuẩn và áp dụng nó cho tất cả các yêu cầu.
- Bắt buộc sử dụng ngôn ngữ một cách thống nhất
- Đánh dấu những phần quan trọng trong các yêu cầu.
- Tránh sử dụng những từ ngữ mang tính chuyên môn, kỹ thuật.

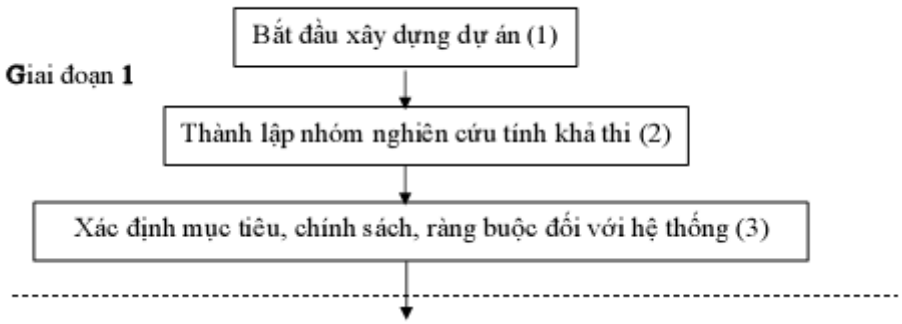
3.4 Quy trình xác định yêu cầu

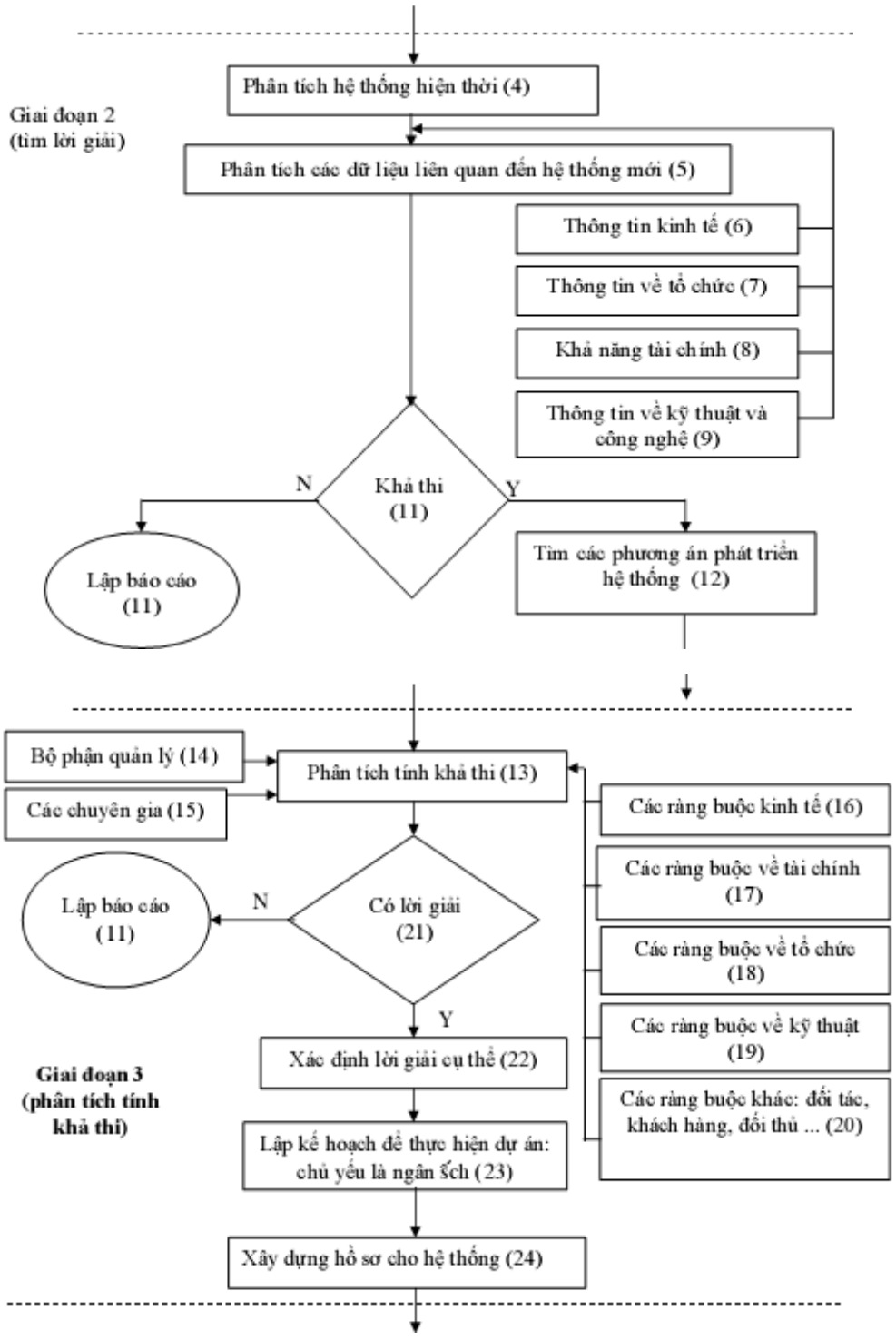
3.4.1 Nghiên cứu tính khả thi dự án

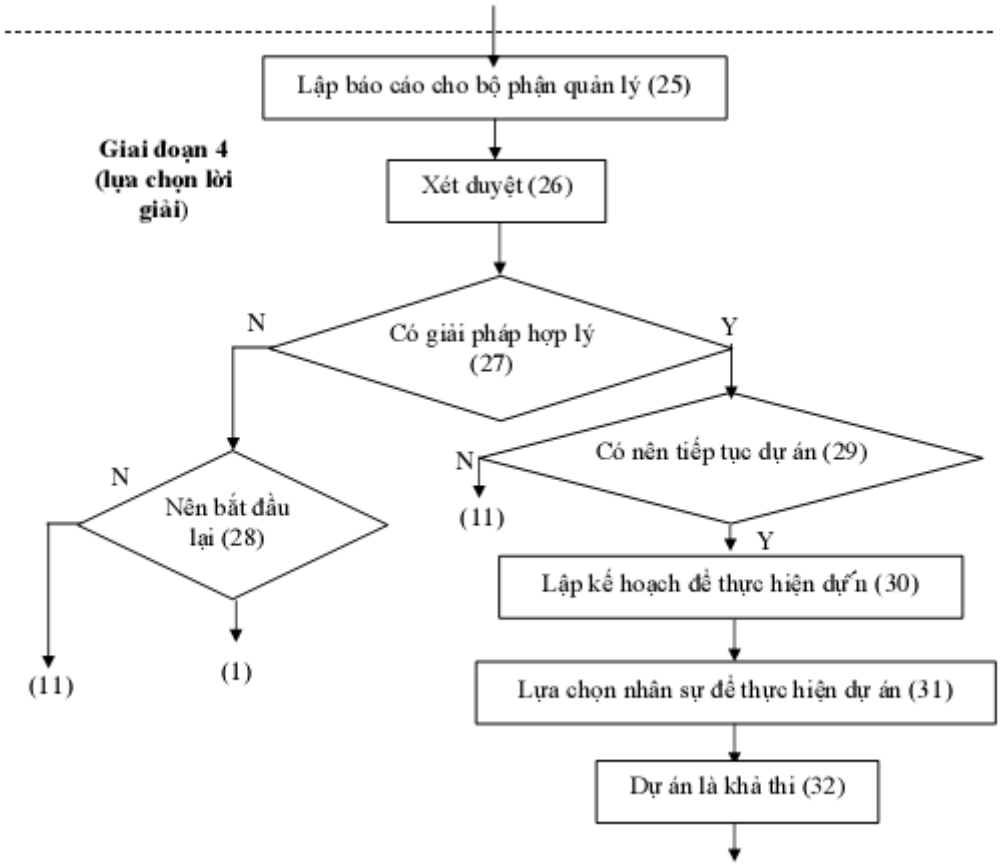
Đối với tất cả các hệ thống mới, quy trình xác định yêu cầu thường bắt đầu bằng việc phân tích khả thi. Thông tin đầu vào để phân tích khả thi là các yêu cầu nghiệp vụ, mô tả sơ bộ về hệ thống, cách thức hệ thống hỗ trợ các yêu cầu nghiệp vụ. Kết quả của việc phân tích khả thi là một báo cáo để quyết định có nên xây dựng hệ thống đề xuất hay không.

Thuật toán nghiên cứu tính khả thi của một số dự án tin học

1. Tổ chức nhóm nghiên cứu tính khả thi: giai đoạn 1
2. Tìm kiếm lời giải: giai đoạn 2
3. Phân tích tính khả thi: giai đoạn 3
4. Lựa chọn lời giải: giai đoạn 4




Giai đoạn 2
(tìm lời giải)
Phân tích hệ thống hiện thời (4)
Phân tích các dữ liệu liên quan đến hệ thống mới (5)
Thông tin kinh tế (6)
Thông tin về tổ chức (7)
Khả năng tài chính (8)
Thông tin về kỹ thuật và công nghệ (9)
Khả thi (11)
N
Y
Lập báo cáo (11)
Tìm các phương án phát triển hệ thống (12)
Bộ phận quản lý (14)
Các chuyên gia (15)
Phân tích tính khả thi (13)
Các ràng buộc kinh tế (16)
Các ràng buộc về tài chính (17)
Các ràng buộc về tổ chức (18)
Các ràng buộc về kỹ thuật (19)
Các ràng buộc khác: đối tác, khách hàng, đối thủ ... (20)
Lập báo cáo (11)
N
Có lời giải (21)
Y
Giai đoạn 3
(phân tích tính khả thi)
Xác định lời giải cụ thể (22)
Lập kế hoạch để thực hiện dự án: chủ yếu là ngân sách (23)
Xây dựng hồ sơ cho hệ thống (24)



3.4.2 Phát hiện và phân tích yêu cầu

(Tìm hiểu, xác định và phân tích yêu cầu là bước hình thành nên bài toán)

Trong pha phát hiện và phân tích yêu cầu, nhân viên kỹ thuật và khách hàng cùng hợp tác để xác định miền ứng dụng, các dịch vụ mà hệ thống cung cấp, hiệu năng của hệ thống, các ràng buộc vận hành của hệ thống…

Ở đây, chúng ta có một khái niệm mới là stakeholder. **Stakeholder** là những người tham dự vào dự án xây dựng hệ thống: người sử dụng cuối, người quản lý, kỹ sư, chuyên gia lĩnh vực, …

Ví dụ, trong hệ thống ATM gồm các Stakeholder sau: khách hàng của ngân hàng, đại diện của các ngân hàng khác, người quản lý ngân hầng, nhân viên ngân hàng, quản trị CSDL, quản lý bảo mật, phòng marketing, kỹ sư bảo trì phần cứng và phần mềm, người điều hành ngân hàng.

Tuy nhiên, việc phát hiện và tìm hiểu yêu cầu của stakeholder, chúng ta thường gặp khó khăn vì những nguyên nhân sau:

- Stakeholder không biết những gì mà họ thật sự mong muốn.

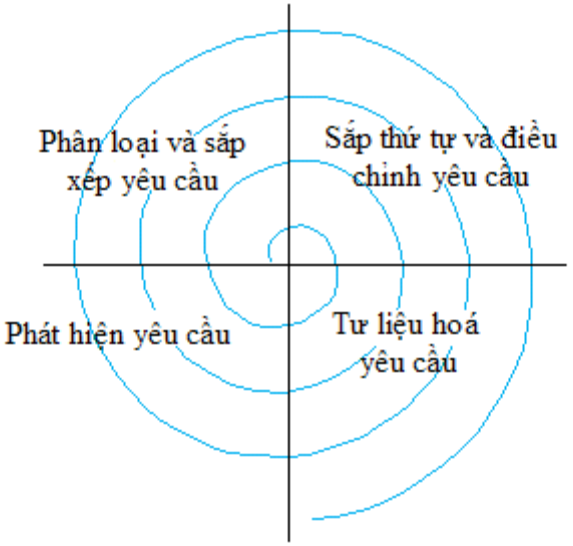
- Stakeholder mô tả các yêu cầu theo thuật ngữ của họ.

- Những stakeholder khác nhau có thể có các yêu cầu xung đột nhau

- Những yếu tố tổ chức và quyền lực có thể ảnh hưởng tới các yêu cầu hệ thống.

- Các yêu cầu có thể thay đổi trong suốt quá trình phân tích. Những stakeholder mới có thể xuất hiện và môi trường nghiệp vụ có thể thay đổi.

Do đó, người ta thường sử dụng mô hình xoắn ốc trong quy trình phát hiện và phân tích yêu cầu



Hình 5.3: Quy trình phát hiện và phân tích yêu cầu

Trong quy trình này bao gồm các hoạt động sau:

- Phát hiện yêu cầu: tiếp xúc với các stakeholder để phát hiện ra các yêu cầu của họ. Các yêu cầu miền ứng dụng cũng được phát hiện ở bước này.

- Phân loại và sắp xếp yêu cầu: nhóm các yêu cầu có liên quan lẫn nhau và tổ chức chúng thành những nhóm gắn kết với nhau.

- Sắp thứ tự ưu tiên và điều chỉnh các yêu cầu xung đột: khi có nhiều stakeholder thì các yêu cầu của họ càng có nhiều xung đột. Hoạt động này nhằm đánh thứ tự ưu tiên của các yêu cầu, phát hiện và giải quyết xung đột giữa các yêu cầu.

- Tư liệu hóa yêu cầu: yêu cầu được tư liệu hoá và là đầu vào của vòng kế tiếp trong mô hình xoắn ốc.

a) ***Phát hiện yêu cầu:***

Phát hiên yêu cầu là quy trình thu thập những thông tin về hệ thống được đề xuất và hệ thống đang tồn tại để xác định các yêu cầu hệ thống và yêu cầu của người sử dụng. Ta có thể lấy thông tin này từ các tư liệu, stakeholder, và bản đặc tả của những hệ thống tương tự. Chúng ta giao tiếp với stakeholder thông qua phỏng vấn hoặc quan sát và có thể sử dụng kịch bản và mẫu thử ….để giúp phát hiện yêu cầu.

*** Một số kỹ thuật để thu thập dữ liệu và phát hiện yêu cầu của hệ thống**

1. Khung nhìn (Viewpoint)

Khung nhìn là cách xây dựng yêu cầu để trình bày với từng stakeholder khác nhau. Ta có thể phân loại Stakeholder theo nhiều khung nhìn khác nhau. Phân tích dựa trên khung nhìn cho phép phát hiện nhiều khía cạnh khác nhau của một vấn đề và giúp phát hiện ra sự xung đột giữa các yêu cầu.

Khung nhìn được chia thành 3 loại chính và mỗi loại sẽ cung cấp các yêu cầu khác nhau.

- Khung nhìn tương tác: là những người hoặc hệ thống khác tương tác với hệ thống. Trong hệ thống ATM, khách hàng và CSDL tài khoản là những khung nhìn tương tác

- Khung nhìn gián tiếp: là những stakeholder không sử dụng hệ thống trực tiếp nhưng có ảnh hưởng tới hệ thống. Trong hệ thống ATM, nhân viên quản lý và bảo mật là những khung nhìn gián tiếp.

- Khung nhìn miền ứng dụng: là những đặc điểm và ràng buộc của miền ứng dụng, có ảnh hưởng tới các yêu cầu. Trong hệ thống ATM, các chuẩn để giao tiếp giữa nhiều ngân hàng là một ví dụ.

Ta có thể phát hiện khung nhìn dựa trên:

- Người cung cấp và người nhận các dịch vụ của hệ thống

- Các hệ thống tương tác trực tiếp với hệ thống cần xây dựng.

- Các chuẩn và các quy tắc

- Tài nguyên và các yêu cầu phi chức năng

- Marketing và các khung nhìn nghiệp vụ khác.

2. Phỏng vấn

Phỏng vấn là việc tập hợp một số lượng ít người – thường với một hoặc 2 người, cho một thời gian cố định với một mục đích cụ thể. Trong quá trình phỏng vấn, những người xác định yêu cầu sẽ đặt ra các câu hỏi cho stakeholder về hệ thống hiện tại họ đang sử dụng và hệ thống sẽ được xây dựng. Và các yêu cầu sẽ được lấy ra từ những câu trả lời của stakeholder.

Phỏng vấn thường sủ dụng hai kiểu câu hỏi:

- Câu hỏi đóng: tập các câu hỏi đã được định nghĩa trước và có nhiều đáp án để stakeholder lựa chọn trả lời. Ví dụ cho câu hỏi đóng: “Bạn có dùng các báo cáo hàng

tháng không?", các câu trả lời "có" có thể được tiếp nối bởi các câu hỏi mở: "Ông có thể giải thích…… "

- Câu hỏi mở: Là câu hỏi cho nhiều trả lời khác nhau, stakeholder phải tự giải thích và phát biểu theo quan điểm của mình về câu hỏi được người phỏng vấn đặt ra. Ví dụ :" Ông có thể nói cho tôi về……", "Ông nghĩ thế nào về…….", "Ông có thể mô tả làm thế nào để……."

Trong thực tế, chúng ta thường trộn lẫn phỏng vấn đóng và mở. Một phỏng vấn tốt có nghĩa là sẽ thu thập được tất cả các hiểu biết về công việc phải làm của stakehoder và cách họ tương tác với hệ thống như thế nào.

Để phỏng vấn thành công, người phỏng vấn nên tiến hành các bước như sau:

- Tiến hành đặt cuộc hẹn phù hợp với thời gian của người được phỏng vấn
- Chuẩn bị tốt: Tìm hiểu kỹ về người được phỏng vấn
- Đúng giờ
- Có kế hoạch cho cuộc phỏng vấn:

+ Giới thiệu bản thân và mục đích

+ Sử dụng câu hỏi mở để bắt đầu

+ Luôn chú ý vào trả lời

+ Có kế hoạch cho nội dung chính

+ Kết hợp câu hỏi đóng và mở

+ Luôn bám sát các trình bày và phát triển chi tiết

+ Luôn cung cấp các thông tin phản hồi, ví dụ: " Cho phép tôi trình bày lại điều ông vừa nói..."

+ Hạn chế ghi chép nếu thấy không tiện

+ Có kế hoạch kết thúc

+ Tóm tắt nội dung, yêu cầu hiệu chỉnh

+ Yêu cầu làm chính xác, đánh giá lại ghi chép

+ Cho biết ngày tháng họ sẽ nhận được báo cáo

+ Thống nhất lại ngày lấy lại bản hiệu chỉnh

+ Xác nhận lại lịch làm việc

Phỏng vấn hình thức hoặc phi hình thức là một trong những phần quan trọng

nhất của quy trình xác định yêu cầu. Các câu hỏi có thể được đưa ra theo kiểu cấu trúc hoặc phi cấu trúc.

Phỏng vấn hình thức là phỏng vấn trong đó người phỏng vấn đã có danh mục các mục cần duyệt qua, các câu hỏi xác định và các thông tin cần biết xác định.

Phỏng vấn phi hình thức là phỏng vấn được định hướng bởi câu trả lời. Các câu hỏi phần lớn là câu hỏi mở. Không có một kế hoạch ban đầu, do vậy người phỏng vấn căn cứ váo các câu trả lời của các câu hỏi mở để phát triển mọi câu hỏi chi tiết hơn về chủ đề. Phỏng vấn hình thức thích hợp khi bạn biết về các thông tin cần thiết trước khi phỏng vấn. Ngược lại phi hình thức thích hợp khi bạn không đoán trước được chủ đề.

Chú ý: Để buổi phỏng vấn thành công, người phỏng vấn nên:

- Cởi mở, sẵn sàng lắng nghe stakeholder và không nên có những ý tưởng đã được định hình sẵn về các yêu cầu.

- Đưa ra những câu hỏi gợi mở, không nên hỏi những câu như "Anh muốn gì?"

3. Kịch bản (ấn định công việc tạm thời)

Chúng ta thường hiểu một vấn đề thông qua các ví dụ thực tế (thực hành) dễ dàng hơn là thông qua những mô tả trừu tượng về nó (lý thuyết). Do đó, chúng ta có thể sử dụng kịch bản để phát hiện ra các yêu cầu hệ thống (tạm thời đóng vai trò là người sử dụng).

Với một công việc tạm thời, ta có được nhận thức đầy đủ hơn về các nhiệm vụ, đồng thời nó cung cấp cho ta kinh nghiệm thực tế. Đầu tiên, ta học các thuật ngữ và ngữ cảnh sử dụng nó. Thời gian tiến hành thường từ 2 tuần đến 1 tháng đủ dài để bạn có thể quen thuộc với phần lớn các công việc thông thường và các tình huống ngoại lệ nhưng không được quá dài để trở thành chuyên gia thực sự đối với công việc.

Bất lợi của công việc tạm thời là tốn thời gian và sự lựa chọn về thời gian có thể làm tối thiểu hoá vấn đề. Thêm vào đó, người kỹ sư phần mềm có thể có thiên kiến về quá trình xử lý công việc, nội dung làm ảnh hưởng tới công việc thiết kế sau này.

4.Quan sát

Quan sát là hoạt động có thể tiến hành thủ công hoặc tự động như sau:

+ Theo cách thủ công, người quan sát ngồi tại một chỗ và ghi chép các hoạt động, các bước xử lý công việc. Ghi chép hoặc băng ghi hình được dùng để phân tích cho các sự kiện, các mô tả động từ chính, hoặc các hoạt động chỉ rõ lý do, công việc hoặc các thông tin về công việc.

+Theo cách tự động, máy tính sẽ lưu giữ các chương trình thường trú, lưu lại vết của các chương trình được sử dụng, email và các hoạt động khác được xử lý bởi máy. Các file nhật ký của máy sẽ được phân tích để mô tả công việc.

Quan sát có các nhược điểm:

- Thời gian của quan sát có thể không biểu diễn cho các công việc diễn ra thông thường,
- Ý nghĩ là đang bị quan sát có thể làm thay đổi thói quen thường ngày của người bị quan sát,
- Tốn thời gian.

Tuy nhiên, quan sát có các ưu điểm:

- Kỹ sư phần mềm có thể nhận được sự hiểu biết tốt về môi trường công tác hiện tại và quá trình xử lý thông qua quan sát.
- Kỹ sư phần mềm có thể tập trung vào vấn đề, mà không bị ảnh hưởng bởi người khác.
- Các ngăn cách giữa kỹ sư phần mềm và các người được phỏng vấn sẽ được vượt qua bởi quan sát.

Để quan sát có hiệu quả, nên xác định cái gì sẽ được quan sát và xác định thời gian cần thiết cho việc quan sát. Đồng thời, hãy xin sự chấp thuận của cả người quản lý và cá nhân trước khi tiến hành quan sát.

5. Xem xét tài liệu

Khái niệm tài liệu ám chỉ tới các cẩm nang, quy định, các thao tác chuẩn mà tổ chức cung cấp như là hướng dẫn cho các nhà quản lý và nhân viên. Các tài liệu thực sự hữu ích cho các kỹ sư phần mềm để học về các lĩnh vực mà họ chưa từng có kinh nghiệm. Nó có thể hữu ích cho việc xác định các câu hỏi về quá trình thao tác và sản xuất. Tài liệu đưa ra các thông tin khách quan.

6. Xem xét phần mềm

Khi các ứng dụng cũ phải được thay thế các phần mềm mới, việc nghiên cứu các phần mềm đã tồn tại cung cấp cho chúng ta các thông tin về quá trình xử lý công việc hiện thời và các mở rộng có ràng buộc bởi thiết kế phần mềm. Khuyết điểm chính của việc nhận thông tin từ quan sát phần mềm là tài liệu có thể không chính xác hoặc kịp thời. Thêm vào đó, thời gian có thể bị lãng phí nếu ứng dụng đang bị xoá bỏ.

b) Phân tích yêu cầu

Nghiên cứu kỹ các yêu cầu của người sử dụng và của hệ thống phần mềm để

xây dựng các đặc tả về hệ thống là cần thiết, nó sẽ xác định hành vi của hệ thống. Nhiệm vụ của giai đoạn này là phải trả lời được các câu hỏi sau:

- Đầu vào của hệ thống là gì
- Những quá trình cần xử lý trong hệ thống, hay hệ thống phần mềm sẽ phải xử lý những cái gì.
- Đầu ra: kết quả xử lý của hệ thống là gì
- Những ràng buộc trong hệ thống, chủ yếu là mối quan hệ giữa đầu vào và đầu ra như thế nào.

Trả lời được câu hỏi trên, nghĩa là phải xác định được chi tiết các yêu cầu làm cơ sở để đặc tả hệ thống. Đó là kết quả của sự trao đổi, thống nhất giữa người đầu tư, người sử dụng với người xây dựng hệ thống. Mục tiêu là xây dựng các hồ sơ mô tả chi tiết các yêu cầu của bài toán nhằm nêu bật được hành vi, chức năng cần thực hiện của hệ thống dự kiến.

Như vậy, phân tích yêu cầu là quá trình suy luận các yêu cầu hệ thống thông qua quan sát hệ thống hiện tại, thảo luận với các người sử dụng, phân tích công việc. Việc này có thể liên quan với việc tạo một hay nhiều mô hình khác nhau. Nó giúp các phân tích viên hiểu biết hệ thống. Các mẫu hệ thống cũng có thể được phát triển để mô tả các yêu cầu

3.4.3 Tài liệu đặc tả yêu cầu

- Tài liệu đặc tả yêu cầu là những yêu cầu chính thức về những gì cần phải thực hiện bởi đội phát triển hệ thống.

- Tài liệu đặc tả yêu cầu nên chứa tất cả các định nghĩa về yêu cầu của người sử dụng và đặc tả yêu cầu hệ thống.

- Tài liệu đặc tả yêu cầu không phải là tài liệu thiết kế hệ thống. Nó chỉ thiết lập những gì hệ thống phải làm, chứ không phải mô tả rõ làm như thế nào.

Về nguyên tắc, các yêu cầu của phần mềm cần được mô tả thật rõ ràng, cụ thể, đầy đủ và chính xác và thống nhất.

Tính đầy đủ: Chúng nên chứa mọi mô tả về các tiện ích được yêu cầu.

Tính thống nhất: Chúng không chứa các xung đột hoặc các mâu thuẫn trong các mô tả về các tiện ích hệ thống.

Các mô tả này là cơ sở để nghiệm thu, đánh giá phần mềm trước khi chuyển giao đến người dùng. Việc mô tả sơ sài, mơ hồ các yêu cầu phần mềm sẽ dẫn đến sự hiểu nhầm giữa những người phát triển hệ thống và khách hàng. Sửa chữa những hiểu

nhầm này thực tế cho thấy phải mất rất nhiều công sức, tiền bạc và thời gian.

Để đặc tả các yêu cầu, người ta thường dùng một số công cụ đặc tả sau:

3.4.3.1 Đặc tả bằng ngôn ngữ tự nhiên

Nói chung, ngôn ngữ tự nhiên có thể được sử dụng để viết các yêu cầu hệ thống cũng như yêu cầu của người sử dụng. Tuy nhiên, yêu cầu hệ thống thường chi tiết hơn yêu cầu của người sử dụng nên đặc tả bằng ngôn ngữ tự nhiên thường gặp một số vấn đề sau:

- ✓ ***Thiếu tính trong sáng:*** Khó tạo tài liệu chính xác mà không làm cho nó khó đọc**.**
- ✓ ***Sự lộn xộn các yêu cầu:*** Các yêu cầu chức năng và phi chức năng bị trộn lẫn.
- ✓ ***Sự pha trộn các yêu cầu:*** Một số kiểu yêu cầu khác nhau được biểu diễn cùng nhau. Ví dụ: Một yêu cầu mô tả ở cả mức độ chi tiết và mức độ khái niệm

*** Một số ví dụ về các yêu cầu được phát biểu bằng ngôn ngữ tự nhiên:**

REQ13: *"LIBSYS nên cung cấp một hệ thống tài khoản tài chính mà lưu tất cả các báo cáo về các chi trả được tạo bởi người dùng hệ thống. Người quản trị hệ thống có thể cấu hình hệ thống này sao cho những người dùng thường xuyên có thể nhận được các khấu trừ."*

REQ14: *Yêu cầu đối với lưới soạn thảo biểu đồ:*

"Các tiện nghi của lưới: *Trợ giúp trong việc đặt vị trí của các thực thể trong một biểu đồ, người dùng có thể bật lưới ở chế độ centimet hoặc inch, thông qua việc chọn trên panel điều khiển. Ban đầu, lưới ở chế độ tắt. Lưới có thể được bật và tắt ở thời điểm bất kỳ trong suốt một phiên soạn thảo và có thể chuyển đổi từ inch sang centimet và ngược lại. Tùy chọn lưới sẽ cung cấp khung nhìn có thể vặn nhỏ nhưng số dòng lưới sẽ bị ẩn đi để tránh lấp kín biểu đồ nhỏ hơn bởi các dòng lưới này.* **"**

*** Các vấn đề hai yêu cầu trên**

REQ13: Yêu cầu này chứa cả hai loại thông tin chi tiết và ở mức khái niệm như sau:

- ➢ Chứa các mô tả ở mức khái niệm về hệ thống tài khoản tài chính trong LIBSYS.
- ➢ Tuy nhiên, nó cũng chứa thông tin chi tiết về việc những nhà quản trị có thể cấu hình hệ thống này – Thông tin này không cần thiết ở mức này.

=> đây chính là sự pha trộn giữa các mức độ yêu cầu

REQ14: Yêu cầu này trộn lẫn 3 kiểu yêu cầu khác nhau:

- Yêu cầu chức năng ở mức khái niệm (sự cần thiết cho một lưới soạn thảo)
- Yêu cầu phi chức năng (các đơn vị lưới).
- Yêu cầu giao diện người dùng phi chức năng (sự bật, tắt lưới).

=> đây chính là sự lộn xộn giữa các loại yêu cầu

3.4.3.2 Dùng các biểu diễn có cấu trúc

Sử dụng ngôn ngữ hướng cấu trúc sẽ yêu cầu người viết đặc tả tuân theo những mẫu được định nghĩa trước. Tất cả các yêu cầu đều được viết theo mẫu này. Ưu điểm của phương pháp này là đạt được mức độ diễn tả cao nhất của ngôn ngữ tự nhiên nhưng mức độ đồng nhất lại bị lạm dụng trong các đặc tả, các thuật ngữ cần sử dụng sẽ bị hạn chế hơn.

*** Một số ví dụ**

REQ15: Các tiện nghi của lưới

> ***"Bộ soạn thảo sẻ cung cấp tiện ích của lưới ở đó một ma trận gồm các dòng và cột cung cấp nên cho cửa sổ soạn thảo****. Lưới sẽ là lưới bị động, ở đó sự căn chỉnh các thực thể là trách nhiệm của người dùng.*
>
> - ✓ ***Lý do cơ bản****: Lưới giúp người dùng tạo các biểu đồ có trật tự/gọn gang với các thực thể có không gian tốt. Khi lưới được kích hoạt, ở đó các thực thể che đi các dòng lưới có thể là hữu ích, khi vị trí không cần chính xác. Người dùng là người tốt nhất quyết định vị trí của các thực thể*
> - ✓ ***Đặc tả****: ECLIPSE/WS/Tools/DE/FS Section 5.6*

Thông thường các yêu cầu người dùng thường được biểu diễn bởi ngôn ngữ tự nhiên hoặc các biểu diễn có cấu trúc. Tuy nhiên khi biểu diễn các yêu cầu hệ thống bởi các công cụ này thường gặp các vấn đề sau:

- **Mập mờ:** Người đọc và người viết yêu cầu phải hiểu các từ giống nhau theo các cách khác nhau.
- **Quá linh hoạt:** Cùng một yêu cầu có thể được diễn đạt theo một số cách khác nhau trong đặc tả
- **Thiếu tính mô đun hóa:** Các cấu trúc ngôn ngữ tự nhiên là không phù hợp để tổ chức các yêu cầu hê thống.

Do đó, người tả thường dùng các công cụ đặc tả các yêu cầu hệ thống thay thế ngôn ngữ tự nhiên như được mô tả dưới đây.

a. Ngôn ngữ tự nhiên có cấu trúc

Đây là cách tiếp cận phụ thuộc vào việc ta định nghĩa các mẫu biểu - form hoặc các khung sườn – template chuẩn để biểu diễn các đặc tả yêu cầu. Khi đặc tả các yêu cầu băng ngôn ngữ tự nhiên có cấu trúc, tính tự do của người viết các yêu cầu được hạn chế bởi mẫu đã định nghĩa cho các yêu cầu. Mọi yêu cầu được viết theo một cách chuẩn. Thuật ngữ được sử dụng trong mô tả có thể bị hạn chế.

** Đặc tả dựa trên mẫu biểu/Form*

Thường dùng để định nghĩa về chức năng hoặc thực thể của hệ thống. Một đặc tả gồm:

- ✓ Mô tả các đầu vào và nơi chúng đến.
- ✓ Mô tả các đầu ra và nơi chúng đi đến.
- ✓ Chỉ rõ các thực thể khác được yêu cầu.
- ✓ Các điều kiện Pre và post (if appropriate).
- ✓ Các hiệu ứng lề của chức năng (if any).

Ví dụ:

Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose Š the dose in insulin to be delivered
Destination	Main control loop
Action:	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin..
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side-effects	None

** Đặc tả Tabular/bảng*

Các đặc tả này được sử dụng để bổ xung cho ngôn ngữ tự nhiên. Chúng đặc biệt hữu ích khi ta phải định nghĩa một số tiến trình thay thế có thể xảy ra của một hành động.

Ví dụ:

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2-r1)<(r1-r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ((r2-r1) □ (r1-r0))	CompDose = round ((r2-r1)/4) If rounded result = 0 then CompDose = MinimumDose

b. Các ngôn ngữ mô tả thiết kế

Cách tiếp cận này sử dụng một ngôn ngữ giống như ngôn ngữ lập trình nhưng với các đặc trưng trừu tượng hơn để đặc tả các yêu cầu bằng cách định nghĩa mô hình vận hành của hệ thống. Cách tiếp cận này hiện thời không được sử dụng rộng rãi mặc dù nó có thể hữu ích cho việc đặc tả các giao diện.

c. Các ký pháp đồ họa

Một ngôn ngữ đồ họa được bổ sung thêm bởi các chú giải bằng văn bản được sử dụng để định nghĩa các yêu cầu chức năng cho hệ thống, ví dụ trước đây người ta hay dùng ngôn ngữ SADT, ngày nay người ta hay dụng các mô tả Use Case và các biểu đồ trình tự.

Một số dạng biểu biểu diễn:

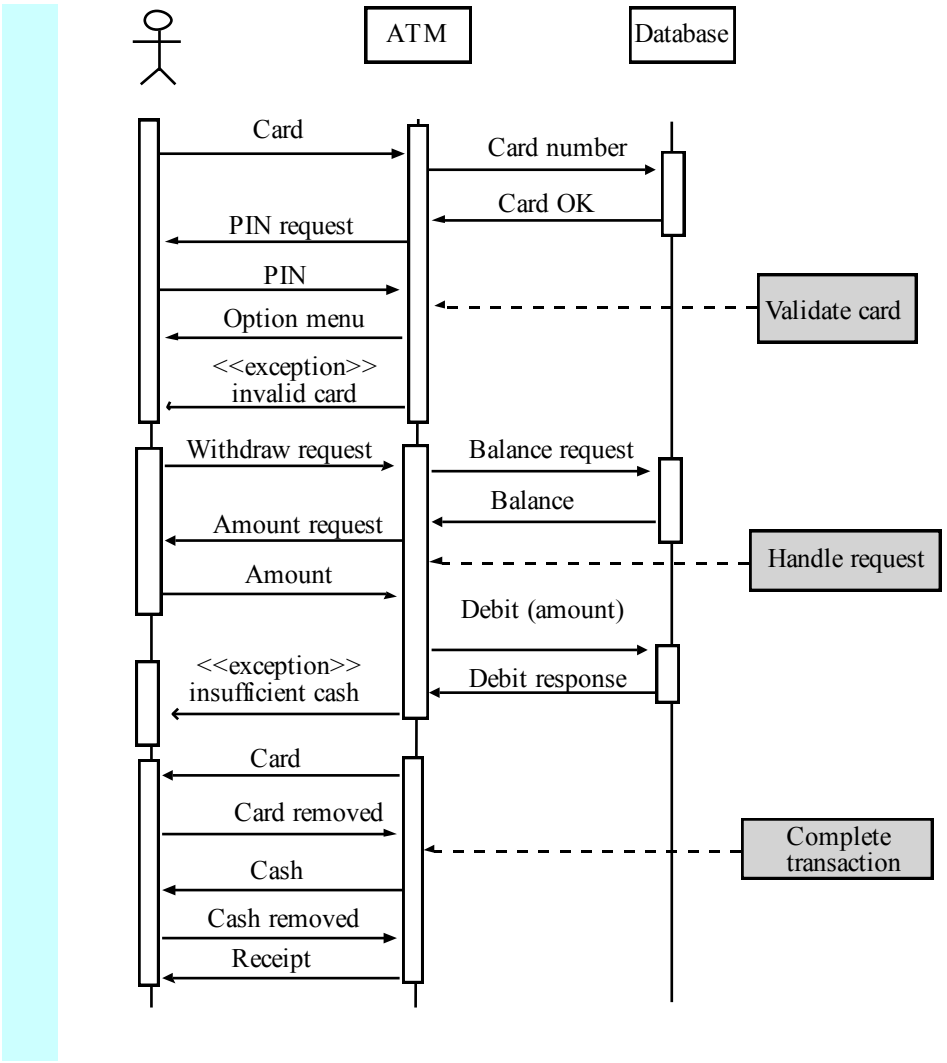
Các mô hình đồ thị: Các mô hình đồ thị hữu ích nhất khi bạn cần chỉ ra các thay đổi trạng thái như thế nào hoặc khi bạn cần mô tả một chuỗi các hoạt động. Các mô hình đồ thị được trình bày trong chương 4.

Các biểu đồ trình tự: Các biểu đồ này chỉ ra một chuỗi các sự kiện xảy ra khi người dùng tương tác với hệ thống. Khi đọc chúng ta đọc từ trên xuống dưới để biết được trình tự các hoạt động diễn ra.

Ví dụ: Xét biểu đồ trình tự của ca rút tiền qua thẻ từ ATM

- Thẩm định thẻ;
- Xử lý yêu cầu;

- Hoàn thành giao dịch.



d. Các đặc tả toán học

Đây là các ký hiệu dựa trên các khái niệm toán học như các máy trạng thái hữu hạn, hoặc các tập hợp. Các đặc tả không mập mờ này giảm các tranh luận giữa khách hàng và nhà thầu về các chức năng của hệ thống. Tuy nhiên hầu hết các khách hàng đều không hiểu các đặc tả hình thức và nó là bất đắc dĩ để chấp nhận nó như là một bản hợp đồng hệ thống.

e. **Đặc tả giao diện**

Hầu hết các hệ thống phải tương tác với các hệ thống khác và các giao diện tương tác phải được đặc tả như là một phần của các yêu cầu. Ba kiểu giao diện có thể được định nghĩa:

- ✓ Các giao diện thủ tục;
- ✓ Các cấu trúc dữ liệu được trao đổi;
- ✓ Các trình bày dữ liệu.

Các ký hiệu hình thức là một kỹ thuật hiệu quả cho việc đặc tả giao diện.

```
interface PrintServer {

// defines an abstract printer server
// requires:          interface Printer, interface PrintDoc
// provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter

            void initialize ( Printer p ) ;
            void print ( Printer p, PrintDoc d ) ;
            void displayPrintQueue ( Printer p ) ;
            void cancelPrintJob (Printer p, PrintDoc d) ;
            void switchPrinter (Printer p1, Printer p2, PrintDoc d) ;
} //PrintServer
```

Tài liệu đặc tả yêu cầu có thể xem như một phần của bản hợp đồng giữa nhà thầu và khách hàng. Sau đây là cấu trúc chung của tài liệu đặc tả yêu cầu dựa theo chuẩn IEEE 839 - 1984:

1. Giới thiệu

1.1. Mục đích của tài liệu yêu cầu

1.2. Phạm vi của sản phẩm

1.3. Các định nghĩa, từ viết tắt

1.4. Các tham chiếu

1.5. Tổng quan về tài liệu yêu cầu (mô tả cấu trúc tài liệu)

2. Mô tả chung

2.1. Tổng quan về sản phẩm

2.2. Các chức năng của sản phẩm

2.3. Đối tương người dùng

2.4. Các ràng buộc tổng thể

2.5. Giả thiết và các phụ thuộc

3. Đặc tả yêu cầu (yêu cầu chi tiết)

3.1 Yêu cầu chức năng

3.1.1 Yêu cầu chức năng 1

3.1.1.1 Giới thiệu

3.1.1.2 Dữ liệu vào

3.1.1.3 xử lý

3.1.1.4 Kết quả

3.1.2 Yêu cầu chức năng 2

...

3.1.n Yêu cầu chức năng n

3.2 Các yêu cầu phi chức năng

3.2.1 Các thuộc tính của hệ thống

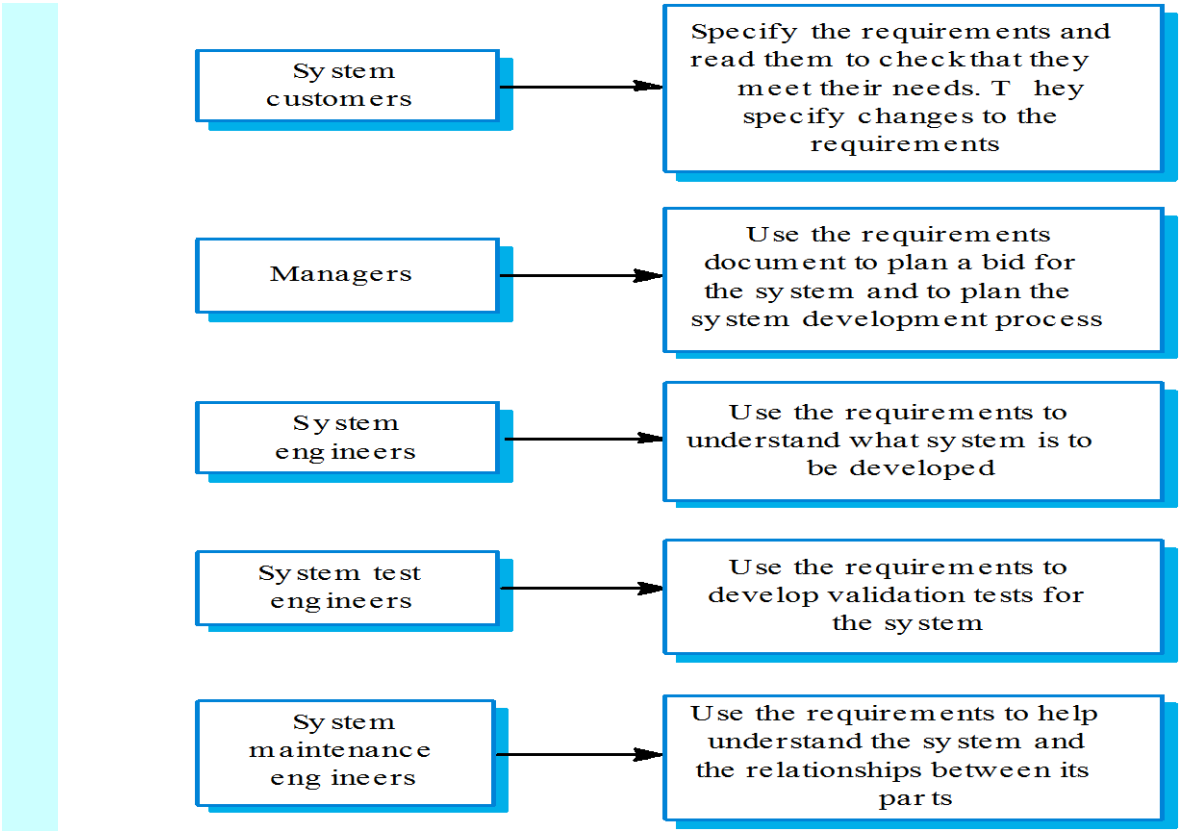
3.2.2 Các ràng buộc của hệ thống

3.2.3 Các yêu cầu khác.

4. Phụ lục

5. Chỉ mục.

Những đối tượng người đọc tài liệu đặc tả yêu cầu và mục đích của việc đọc:



3.4.4 Đánh giá yêu cầu

Đánh giá yêu cầu có liên quan đến việc giải thích các yêu cầu đã được định nghĩa trong hệ thống. Vì chi phí cho việc giải quyết các lỗi có liên quan tới yêu cầu sẽ rất cao cho nên việc đánh giá yêu cầu là vô cùng quan trọng.

Trong quá trình đánh giá yêu cầu, chúng ta phải kiểm tra các yêu cầu ở những khía cạnh sau:

- Hợp lệ: Hệ thống có cung cấp các chức năng mà hỗ trợ tốt nhất cho các yêu cầu của người sử dụng hay không?

- Nhất quán: có yêu cầu nào xung đột nhau hay không?

- Hoàn thiện: tất cả các yêu cầu của khách hàng đã được xác định đầy đủ chưa?

- Hiện thực: các yêu cầu có thể được cài đặt với một ngân sách và công nghệ cho trước?

- Xác thực: các yêu cầu có thể được kiểm tra hay không?

Các kỹ thuật đánh giá yêu cầu sau đây có thể được sử dụng đơn lẻ hoặc hỗn hợp:

- Xem xét lại các yêu cầu: phân tích các yêu cầu một cách hệ thống.

- Mẫu thử: Sử dụng các mô hình hệ thống để kiểm tra các yêu cầu

- Tạo ra các trường hợp kiểm thử

3.4.5 Lập kế hoạch quản lý yêu cầu

Quản lý yêu cầu là quy trình quản lý sự thay đổi của các yêu cầu trong quá trình phát hiện yêu cầu và phát triển hệ thống. Các yêu cầu thường không đầy đủ và không đồng nhất. Đó là do một số nguyên nhân sau:

- Những yêu cầu mới xuất hiện trong quy trình khi các yêu cầu nghiệp vụ thay đổi và khi chúng ta có hiểu biết sâu hơn về hệ thống sẽ xây dựng.

- Ở các khung nhìn khác nhau sẽ có các yêu cầu khác nhau và do đó thường xuất hiện các mâu thuẫn.

- Thứ tự ưu tiên từ các khung nhìn khác nhau cung thay đổi trong suốt quá trình phát triển hệ thống.

- Môi trường nghiệp vụ và môi trương kỹ thuật của hệ thống cũng thay đổi trong có trình xây dựng.

Các yêu cầu lâu dài là những yêu cầu ổn định kế thừa từ những hành động chính của khách hàng. Và nó có thể kế thừa từ nhiều mô hình miền ứng dụng khác.

Các yêu cầu thay đổi là những yêu cầu dễ bị thay đổi trong quá trình xây dựng hoặc

khi hệ thống được đưa vào sử dụng.

Quy trình lập kế hoạch quản lý yêu cầu:

- Xác định yêu cầu: cách xác định từng yêu cầu

- Quản lý thay đổi: xác định các hoạt động tiếp theo khi yêu cầu thay đổi.

    - Các chính sách tìm vết: lượng thông tin về mối quan hệ giữa các yêu cầu cần phải được lưu giữ.

- Hỗ trợ CASE tool: sử dụng các công cụ để hỗ trợ quản lý yêu cầu thay đổi. CASE tool thường hỗ trợ những chức năng như:

    * Lưu trữ yêu cầu: các yêu cầu được quản lý một cách bảo mật và được lưu trong kho dữ liệu.

    * Quản lý thay đổi: quy trình quản lý thay đổi là quy trình luồng công việc mà các trạng thái có thể được định nghĩa và luồng thông tin giữa các trạng thái là tự động.

    * Quản lý vết - tự động tìm kiếm mối liên kết giữa các yêu cầu.

Chúng ta nên áp dụng tất cả các khả năng thay đổi có thể cho tất cả các yêu cầu.

Các pha chính của hoạt động này bao gồm:

- Phân tích vấn đề và đặc tả thay đổi: thảo luận về các vấn đề yêu cầu và những thay đổi có thể xảy ra.

- Phân tích thay đổi và chi phí: Đánh giá ảnh hưởng của sự thay đổi trên các yêu cầu khác.

- Cài đặt thay đổi: Điều chỉnh tài liệu của các yêu cầu và những tài liệu khác để phản ánh sự thay đổi đó.

CHƯƠNG 4. CÁC MÔ HÌNH HỆ THỐNG

MỤC ĐÍCH:

- Hiểu được mô hình hoá hệ thống là gì? Và tại sao phải mô hình hoá hệ thống.

- Phân biệt được các mô hình hệ thống.

- Có khả năng lựa chọn và ứng dụng các mô hình hệ thống vào từng trường hợp cụ thể

4.1 Mô hình ngữ cảnh

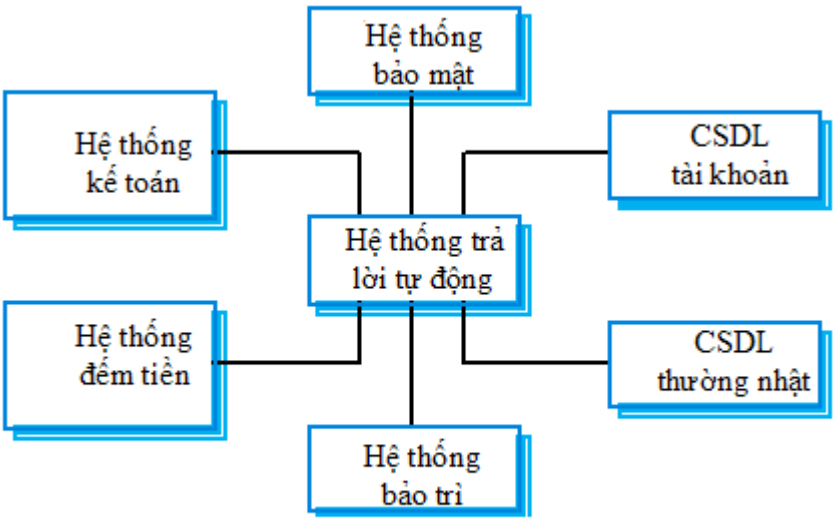
- Khi xem xét một vấn đề, bao giờ chúng ta cũng muốn có cái nhìn tổng thể về vấn đề đó.

- Mô hình ngữ cảnh cho thấy những thành phần cốt lõi của hệ thống.

Trong quá trình phát hiện và phân tích yêu cầu, chúng ta nên xác định phạm vi hệ thống, tức là phân biệt cái gì là hệ thống và cái gì là môi trường của hệ thống. Điều này sẽ giúp giảm chi phí và thời gian phân tích. Khi đã xác định phạm vi của hệ thống, hoạt động tiếp theo của quy trình phân tích là định nghĩa ngữ cảnh của hệ thống và sự phụ thuộc giữa hệ thống với môi trường của nó.

Thông thường, mô hình kiến trúc đơn giản của hệ thống sẽ được tạo ra trong bước này.

Ví dụ: mô hình ngữ cảnh của hệ thống ATM



Hình 6.1: Ngữ cảnh của hệ thống ATM

Mô hình kiến trúc mô tả môi trường của hệ thống, nhưng không chỉ ra quan hệ giữa các hệ thống khác nhau trong một môi trường. Vì vậy, người ta thường sử dụng thêm mô hình tiến trình hoặc mô hình luồng dữ liệu để bổ trợ cho nó.

Mô hình tiến trình biểu diễn tất cả các tiến trình được hệ thống hỗ trợ. Mô hình

luồng dữ liệu có thể được sử dụng để biểu diễn các tiến trình và luồng thông tin đi từ tiến trình này tới tiến trình khác.

4.2 Mô hình ứng xử

Mô hình ứng xử được sử dụng để mô tả toàn bộ ứng xử của hệ thống. Có hai kiểu mô hình ứng xử là:

- Mô hình luồng dữ liệu: biểu diễn cách xử lý dữ liệu trong hệ thống và
- Mô hình máy trạng thái: biểu diễn cách đáp ứng của hệ thống với các sự kiện xảy ra.

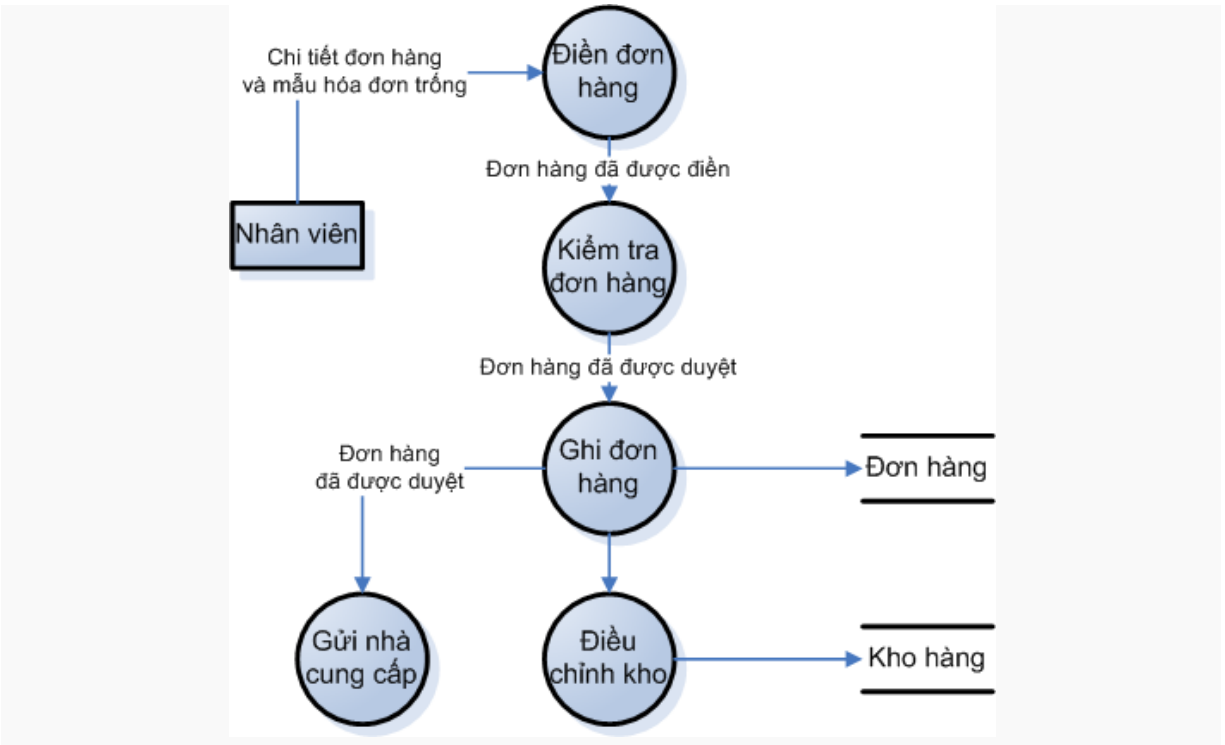
Hai mô hình này biểu diễn những góc nhìn khác nhau, nhưng cả hai đều cần thiết để mô tả ứng xử của hệ thống.

4.2.1 Mô hình luồng dữ liệu

Mô hình luồng dữ liệu được sử dụng để mô hình hoá quy trình xử lý dữ liệu của hệ thống. Mô hình này sẽ biểu diễn các bước mà luồng dữ liệu phải trải qua trong hệ thống từ điểm đầu tới điểm cuối.

Mô hình luồng dữ liệu mô hình hoá hệ thống từ góc độ một chức năng. Việc tìm vết và tư liệu hoá quan hệ giữa dữ liệu với một quy trình rất có ích đối với việc tìm hiểu toàn bộ hệ thống. Mô hình luồng dữ liệu là phần cốt lõi của rất nhiều phương pháp phân tích. Nó chứa các ký pháp rất dễ hiểu đối với khách hàng.

Ví dụ: Mô hình luồng dữ liệu của chức năng xử lý đơn hàng

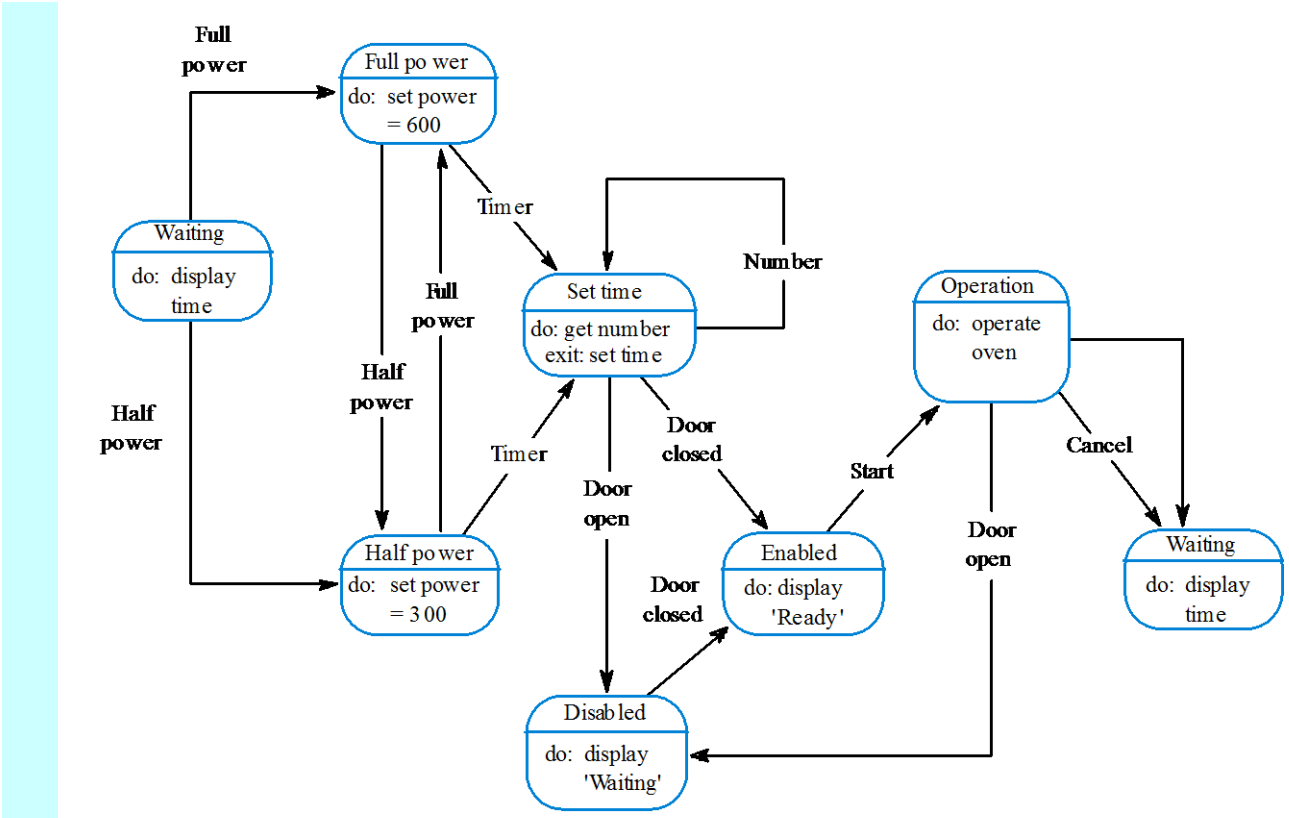


4.2.2 Mô hình máy trạng thái

Mô hình máy trạng thái mô tả đáp ứng của hệ thống với các sự kiện bên trong và bên ngoài của nó. Mô hình máy trạng thái biểu diễn các trạng thái của hệ thống và các sự kiện gây ra sự dịch chuyển trạng thái.

Mô hình máy trạng thái biểu diễn các trạng thái của hệ thống là các nút và sự kiện là các cung nối giữa các nút đó. Khi có một sự kiện xảy ra, hệ thống sẽ dịch chuyển từ trạng thái này sang trạng thái khác. Biểu đồ trạng thái là một biểu đồ trong UML và được sử dụng để biểu diễn mô hình máy trạng thái. Biểu đồ trạng thái cho phép phân tích một mô hình thành nhiều mô hình con và mô tả ngắn gọn về các hành động cần thực hiện tại mỗi trạng thái. Ta có thể vẽ các bảng để mô tả mối quan hệ giữa trạng thái và tác nhân kích hoạt.

Ví dụ: Xét mô hình máy trạng thái của chiếc lò vi sóng



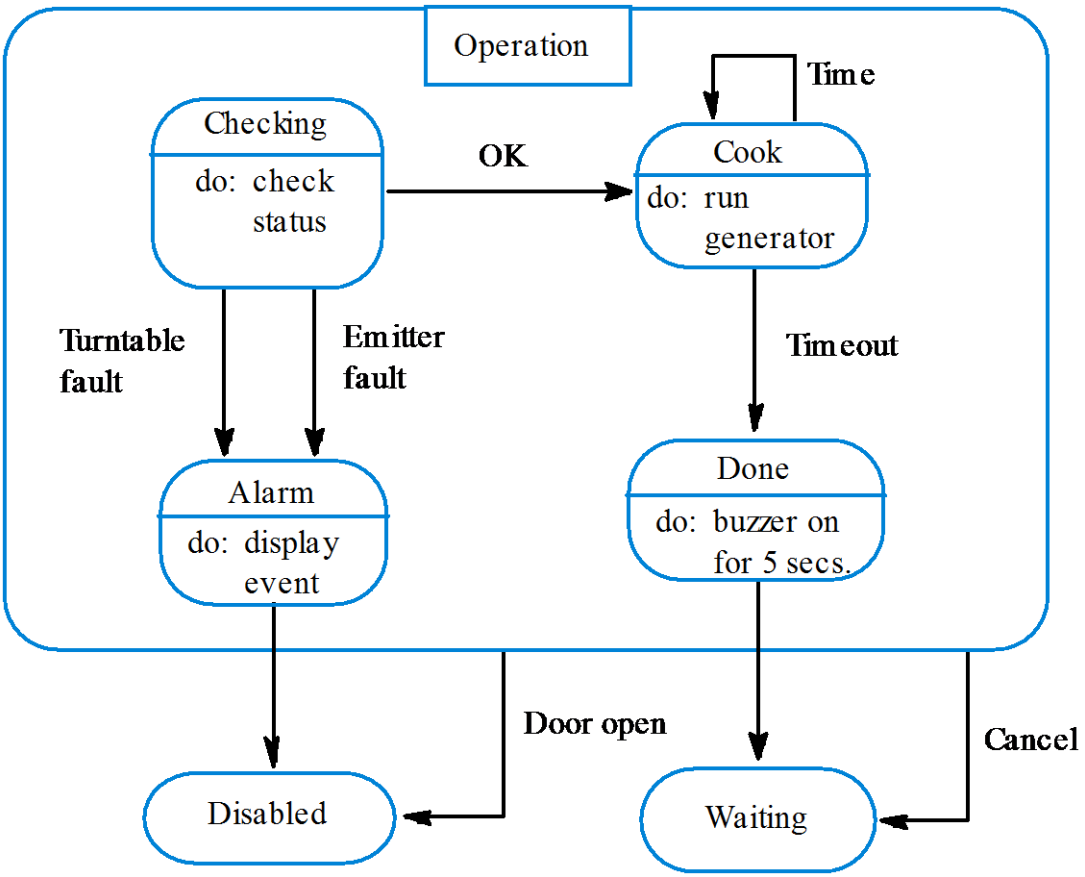
Mô tả các trạng thái của lò vi sóng:

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ÕHalf powerÕ
Full power	The oven power is set to 600 watts. The display shows ÕFull powerÕ
Set time	The cooking time is set to the userÕs input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ÕNot readyÕ
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ÕReady to cookÕ
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for 5 seconds. Oven light is on. Display shows ÕCooking completeÕ while buzzer is sounding.

Các sự kiện (kích thích) của lò vi sóng:

Stimulus	Description
Half power	The user has pressed the half power button
Full power	The user has pressed the full power button
Timer	The user has pressed one of the timer buttons
Number	The user has pressed a numeric key
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start	The user has pressed the start button
Cancel	The user has pressed the cancel button

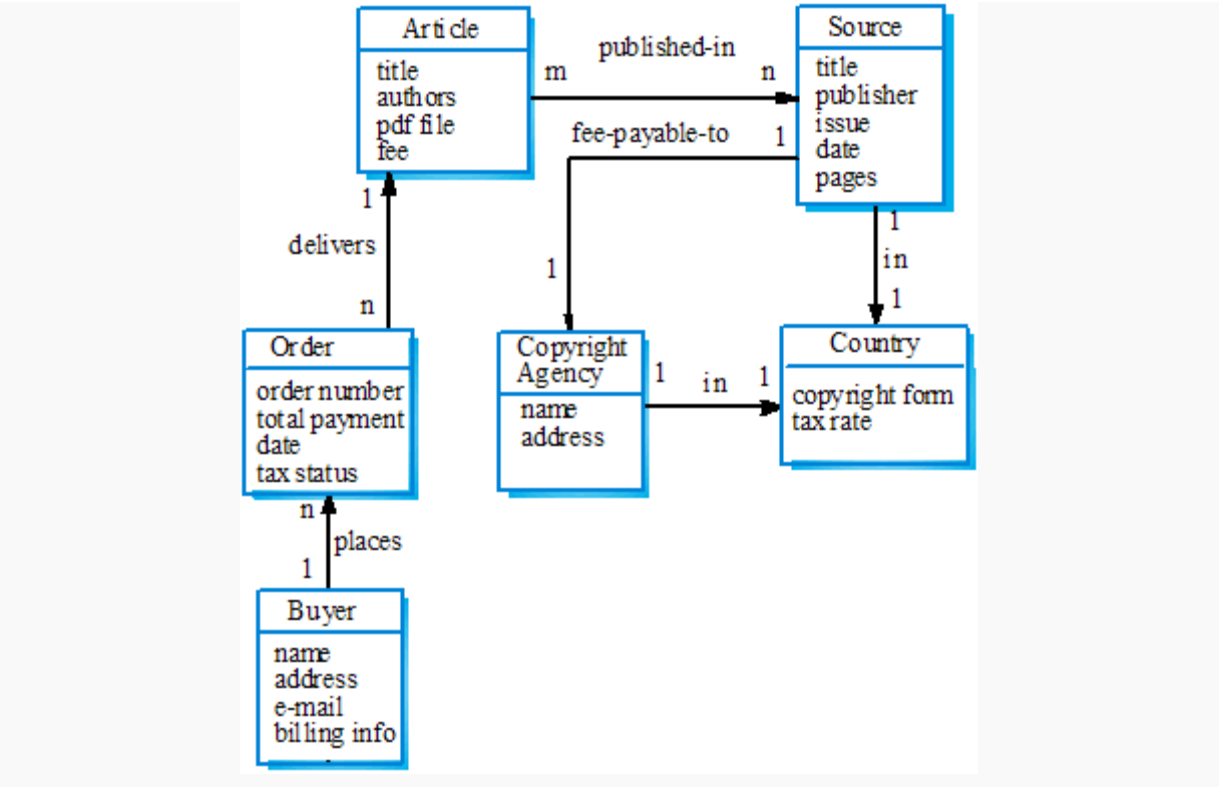
Trạng thái hoạt động/vận hành của lò vi sóng:



4.3 Mô hình dữ liệu

Mô hình dữ liệu được sử dụng để mô tả cấu trúc logic của dữ liệu được xử lý bởi hệ thống. Thông thường, chúng ta hay sử dụng mô hình thực thể - quan hệ - thuộc tính (ERA) thiết lập các thực thể của hệ thống, quan hệ giữa các thực thể và thuộc tính của các thực thể. Mô hình này được sử dụng trong thiết kế CSDL và thường được cài đặt trong các CSDL quan hệ.

Ví dụ mô hình dữ liệu của LIBSYS



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 sách tất cả các tên gọi được sử dụng trong các mô hình hệ thống. Đó có thể là các thực thể, quan hệ và các thuộc tính …. 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.

Ví dụ: từ điển dữ liệu của LIBSYS

Tên	Mô tả	Kiểu
Article	Chi tiết về bài báo có trong LIBSYS	Thực thể
Authors	Tên của tác giả viết bài báo	Thuộc tính
Buyer	Tên của người hoặc tổ chức muốn sao chép bài báo	Thực thể
Fee-payable-to	Quan hệ 1:1 giữa Article và Copyright Agency	Quan hệ
Address Buyer	Địa chỉ của người đặt mua được sử dụng để chuyển bài báo tới	Thuộc tính

4.4 Mô hình đối tượng

Sử dụng mô hình ứng xử hay mô hình dữ liệu 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 hoá trên một tập các đối tượng có thuộc tính và

phương thức chung.

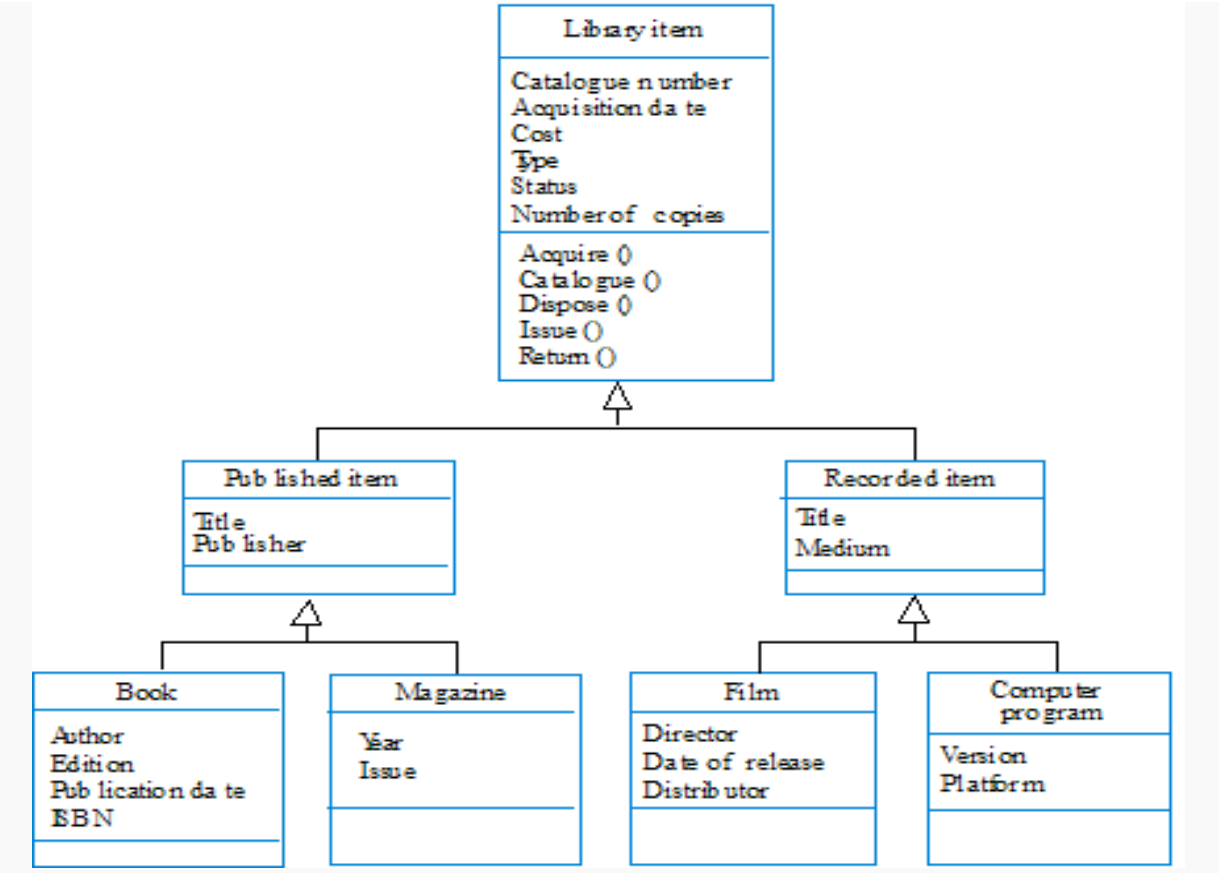
Mô hình đối tượng phản ánh các thực thể trong thế giới thực được vận dụng trong hệ thống. Nếu ta càng có nhiều thực thể trừu tượng thì việc mô hình hoá càng khó khăn. Phát hiện các lớp đối tượng là một quy trình rất khó khăn khi tìm hiểu sâu về lĩnh vực của ứng dụng. Các lớp đối tượng thường phản ảnh 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. Trong phần tiếp theo, chúng ta sẽ tìm hiểu từng loại mô hình này.

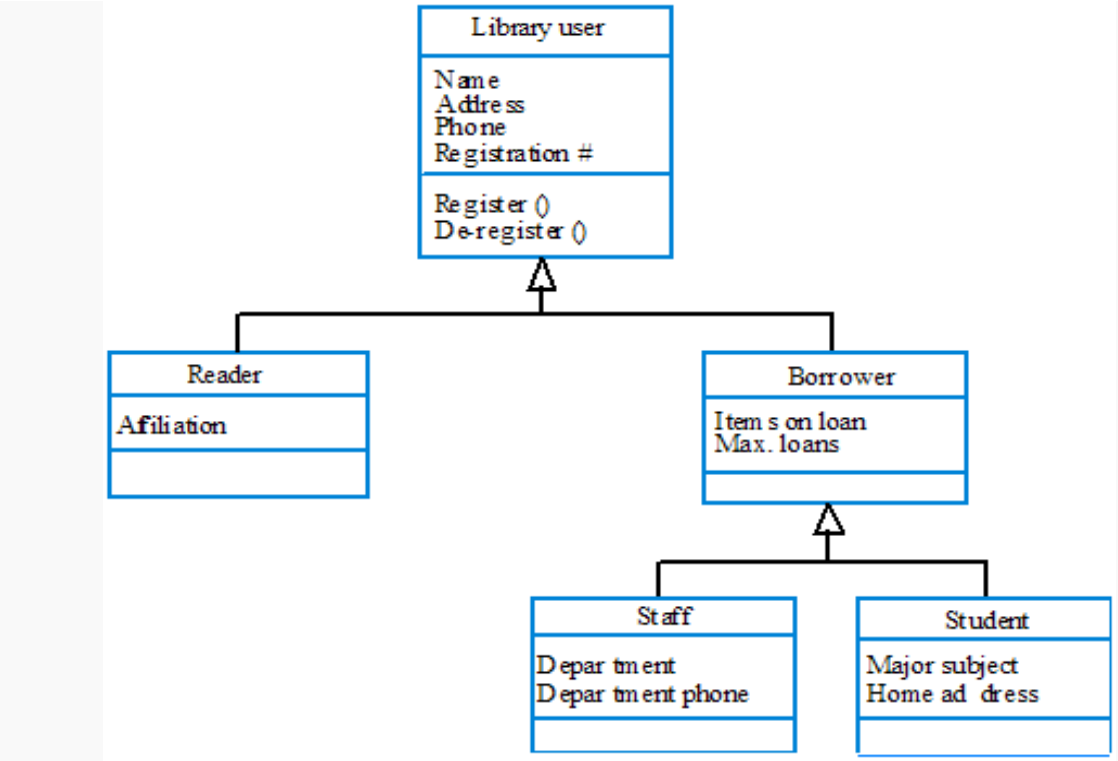
4.4.1 Mô hình thừa kế

Mô hình thừa kế tổ chức các lớp đối tượng theo một cấu trúc phân cấp. Các lớp ở đỉnh của cấu trúc phân cấp phản ánh những đặc trưng chung của tất cả các lớp. Các lớp đối tượng thừa kế những thuộc tính và phương thức của các lớp cha của nó nó có thể bổ sung những đặc điểm của riêng nó. Thiết kế lớp phân cấp là một quy trình khá phức tạp, ta nên loại bỏ sự trùng lặp giữa các nhánh khác nhau.

Ví dụ: cấu trúc phân cấp của lớp Library trong LIBSYS

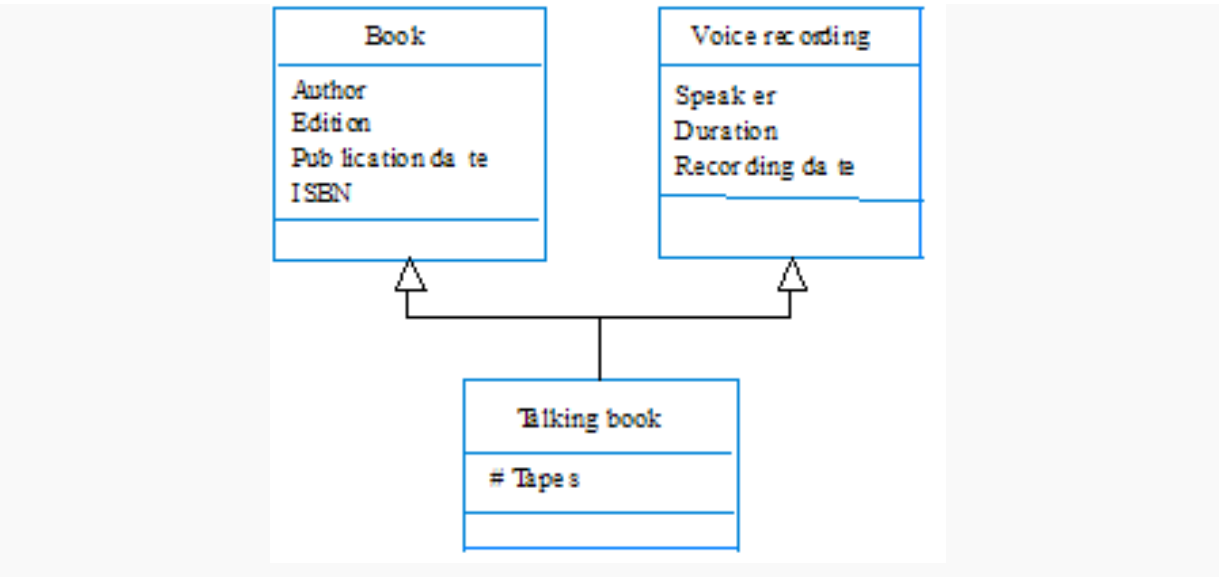


Ví dụ: cấu trúc phân cấp của lớp User trong LIBSYS



Cấu trúc đa thừa kế: lớp đối tượng có thể thừa kế từ một hoặc nhiều lớp cha. Tuy nhiên, điều này có thể dẫn tới sự xung đột về ngữ nghĩa khi các thuộc tính/phương thức trùng tên ở các lớp cha khác nhau có ngữ nghĩa khác nhau.

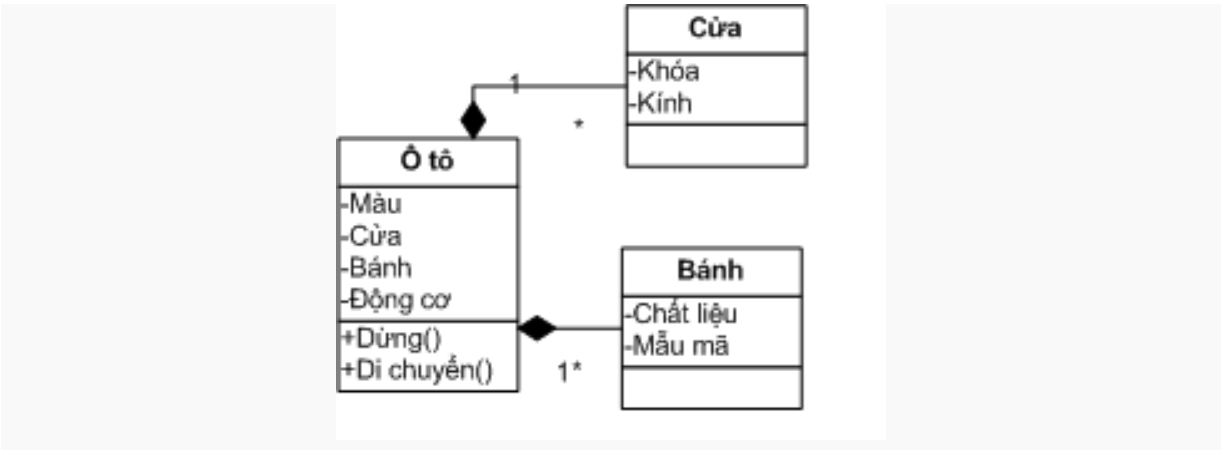
Ví dụ: lớp Talking book thừa kế từ hai lớp Book và Voice recording.



4.4.2 Mô hình kết hợp

Mô hình kết hợp biểu diễn cách cấu tạo của một lớp từ các lớp khác. Mô hình kết hợp tương tự như quan hệ hợp thành (part-of).

Ví dụ: Mô hình kết hợp: Đối tượng ô tô được tạo thành từ nhiều đối tượng khác như: cửa, bánh xe ...

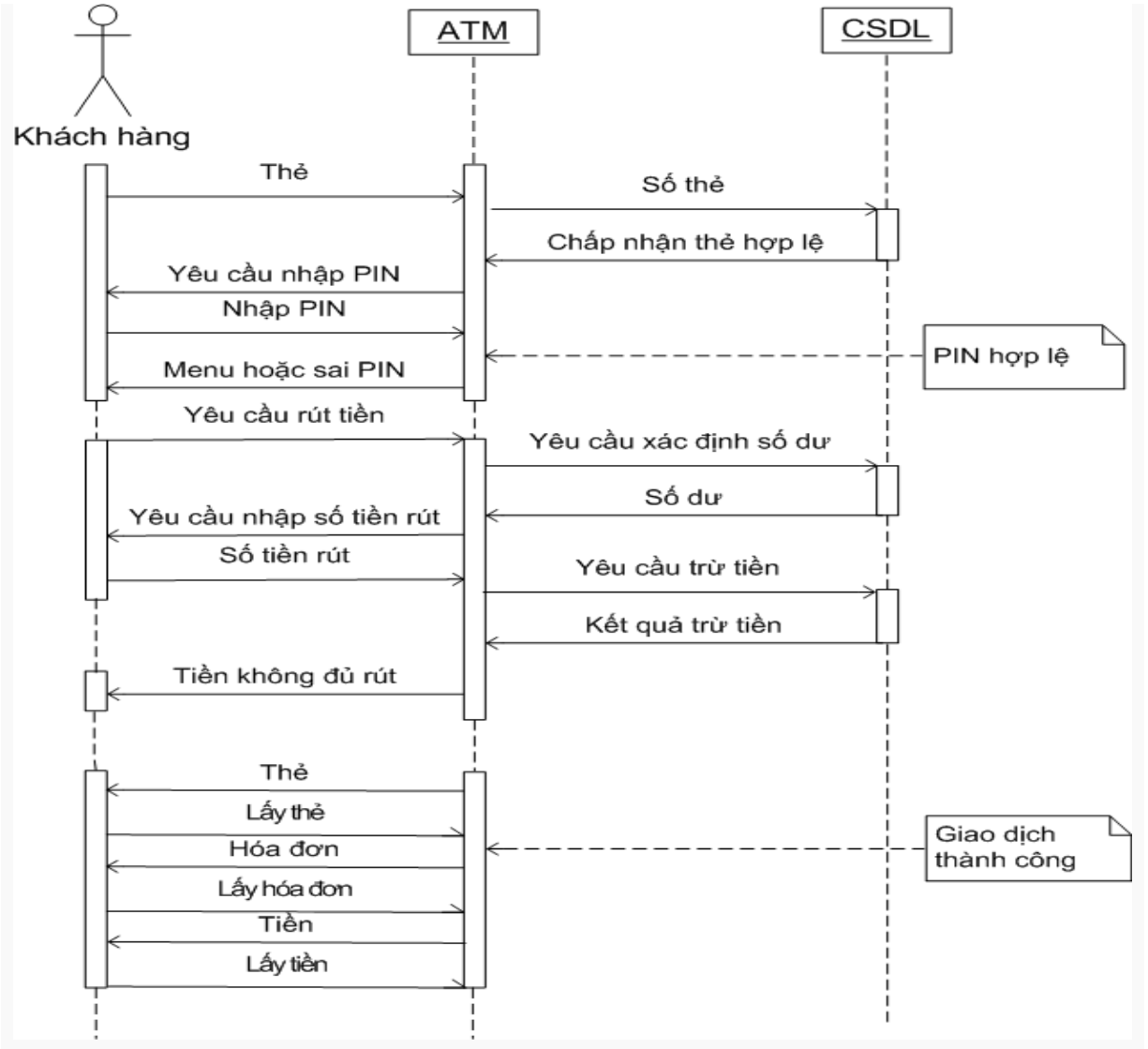


4.4.3 Mô hình ứng xử

Mô hình ứng xử mô tả tương tác giữa các đối tượng nhằm tạo ra một số ứng xử cụ thể của hệ thống mà đã được xác định như là một ca sử dụng.

Biểu đồ trình tự hoặc biểu đồ cộng tác trong UML được sử dụng để mô hình hoá tương tác giữa các đối tượng.

Ví dụ: Mô tả ca sử dụng Rút tiền của hệ thống ATM.



4.5 Phương pháp hướng cấu trúc

Ngày nay, phương pháp hướng cấu trúc rất ít khi được sử dụng do không còn phù hợp với các hệ thống lớn. Tuy nhiên, trong giáo trình này, chúng tôi vẫn trình bày phần này để học viên có cái nhìn mang tính tổng quan về vấn đề này.

Các phương pháp hướng cấu trúc đều cung cấp framework để mô hình hoá hệ thống một cách chi tiết. Chúng thường có một tập hợp các mô hình đã được định nghĩa trước, quy trình để đưa ra các mô hình đó và các quy tắc, hướng dẫn có thể áp dụng cho các mô hình.

Tuy nhiên, các phương pháp hướng cấu trúc thường có một số nhược điểm sau:

- Không mô hình hoá được các yêu cầu hệ thống phi chức năng

- Không chứa những thông tin để xác định liệu một phương thức có thích hợp với một vấn đề đưa ra hay không.

- Tạo ra quá nhiều tài liệu

- Mô hình hoá hệ thống quá chi tiết và khó hiểu đối với người sử dụng.

Đây là một vấn đề đặt ra mà chúng ta cần cải tiến đề xuất trong tương lai.

CASE Workbenches hỗ trợ cho phân tích, thiết kế có cấu trúc: CASE workbenches là tập hợp các công cụ được thiết kế để hỗ trợ các quy trình xây dựng hệ thống phần mềm như phân tích, thiết kế và kiểm thử.

CASE tools hỗ trợ mô hình hoá hệ thống là một công cụ quan trọng của phương pháp hướng cấu trúc. Sau đây là một số CASE tool thường được sử dụng:

- Soạn thảo biểu đồ

- Công cụ phân tích mô hình và kiểm tra

- Ngôn ngữ truy vấn

- Từ điển dữ liệu

- Công cụ tạo và định nghĩa báo cáo

- Công cụ định nghĩa form

- Bộ dịch

- Công cụ tạo mã lệnh tự động

CHƯƠNG 5. THIẾT KẾ VÀ CÀI ĐẶT HỆ THỐNG

MỤC ĐÍCH

- *Nắm vững các hoạt đông trong qui trình thiết kế hệ thống.*
- *Hiểu một số quy tắc trong cài đặt phần mềm*

5.1 Thiết kế hệ thống

5.1.1 Các hoạt động trong quá trình thiết kế hệ thống

Thiết kế phần mềm là hoạt động chuyển đặc tả yêu cầu thành mô hình thiết kế mà người lập trình có thể chuyển thành chương trình với một ngôn ngữ lập trình cụ thể, chương trình có thể vận hành được, đáp ứng được yêu cầu đặt ra.

Thiết kế là một quá trình sáng tạo để tìm giải pháp công nghệ (cách thức, phương án), biểu diễn cách thức, phương án đó, xét duyệt lại và chi tiết hóa dần dần. Bản thiết kế phải đủ chi tiết để người lập trình biết phải làm như thế nào để chuyển thành chương trình.

Vai trò của thiết kế

Thiết kế hệ thống là một hoạt động nhằm trả lời câu hỏi "làm thế nào để triển khai được hệ thống thỏa mãn các yêu cầu phần mềm". Nó tạo ra mô hình cài đặt của phần mềm và là công cụ giao tiếp giữa những người tham gia phát triển hệ thống, là cơ sở để đảm bảo chất lượng hệ thống:

- ✓ Bản thiết kế dễ đọc, dễ hiểu, dễ sửa đổi hơn mã chương trình
- ✓ Có nhiều mức chi tiết, cung cấp cái nhìn tổng thể
- ✓ Làm cơ sở để trao đổi, cải tiến

Bản thiết kế cũng cung cấp đầy đủ thông tin cho việc bảo trì sau này:

- ✓ Giảm công sức mã hóa khi sửa đổi
- ✓ Tiện bảo trì, phát triển, mở rộng.

Thiết kế hệ thống đóng vai trò rất quan trọng, đặc biệt với những hệ thống phần mềm lớn, phức tạp, thời gian sống lâu.

Tiêu chí chất lượng thiết kế

Cần thiết lập các tiêu chí kỹ thuật để đánh giá một bản thiết kế tốt hay không:

- ✓ Thiết kế cần có kiến trúc tốt. Bản thiết kế phải được cấu thành từ các mẫu (pattern), các thành phần có đặc trưng tốt, dễ tiến hóa.
- ✓ Thiết kế được mô đun hóa cho mỗi thành phần chức năng

- ✓ Chứa các biểu diễn tách biệt nhau về dữ liệu, kiến trúc, giao diện, thành phần và môi trường.
- ✓ Liên kết qua giao diện giúp làm giảm độ phức tạp liên kết giữa các mô đun với nhau và giữa hệ thống với môi trường.

Độ đo chất lượng thiết kế

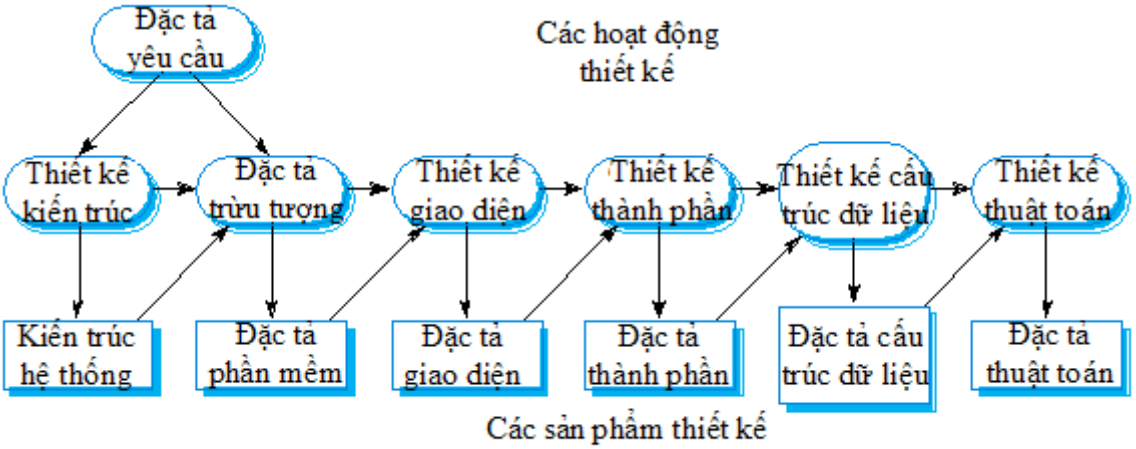
Độ đo chất lượng thiết kế phụ thuộc vào bài toán, không có phương pháp chung. Để đo chất lượng thiết kế người ta thường dùng các độ đo:

***Coupling*:** Mức độ ghép nối giữa các module

Cohension: Mức độ liên kết giữa các thành phần trong một module

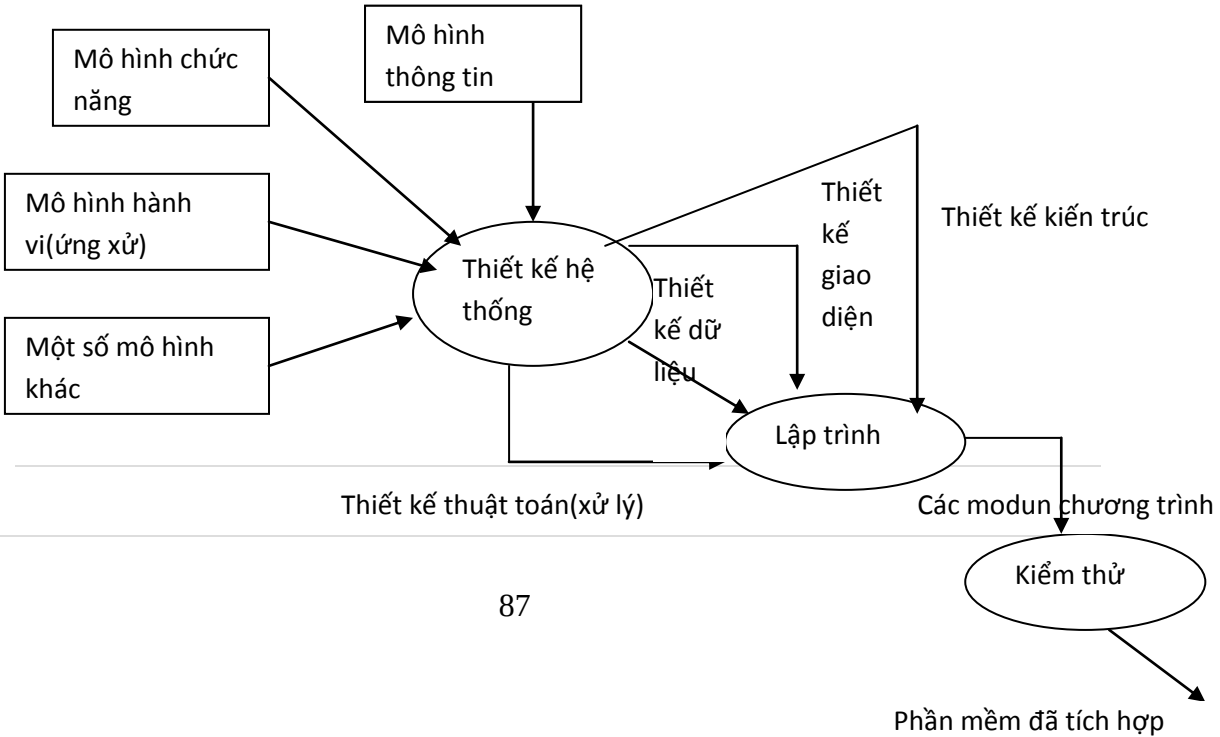
Understandability: Tính hiểu được của bản thiết kế,

Adaptability: Tính thích nghi được.



Hình 2.7: Mô hình chung của quy trình thiết kế

Mối quan hệ giữa thiết kế phần mềm và các pha còn lại trong quy trình phát triển phần mềm được thể hiện qua sơ đồ sau:



5.1.2 Thiết kế kiến trúc

5.1.2.1 Tổng quan về thiết kế kiến trúc

Thiết kế kiến trúc là gì?

- Thiết kế kiến trúc hệ thống là giai đoạn sớm nhất trong quy trình thiết kế hệ thống.

- Quy trình thiết kế kiến trúc nhằm xác định các hệ thống con (các phân hệ) cấu tạo nên hệ thống đề xuất, giao tiếp giữa các phân hệ và framework giúp điều khiển, tương tác giữa các phân hệ

- Thiết kế kiến trúc thường được thực hiện song song với một số hành động đặc tả

- Kết quả của quy trình thiết kế này là bản đặc tả về kiến trúc phần mềm.

Kiến trúc và các đặt tính hệ thống

- ✓ **Tính đáng tin cậy và khả năng thực thi:** Kiến trúc giúp khoanh vùng các thao tác then chốt và tối thiểu hóa giao tiếp đến chúng. Để nâng cao tính thực thi, nên sử dụng các thành phần cỡ lớn (large-grain) và các thành phần cỡ vừa.
- ✓ **Tính an ninh:** Sử dụng kiến trúc phân tầng với các thành phần then chốt/ quý hiếm nằm ở các tầng bên trong góp phần nâng cao tính an ninh của hệ thống.
- ✓ **Tính an toàn:** Khoanh vùng những đặc trưng an toàn then chốt trong một số lượng nhỏ các hệ thống con góp phần nâng cao tính an toàn của hệ thống.
- ✓ **Tính sẵn dùng:** Kiến trúc nên chứa các thành phần dư thừa và các cơ chế dung thứ lỗi để đảm bảo tính sẵn dùng.
- ✓ **Khả năng bảo trì**: Kiến trúc nên sử dụng các thành phần fine-grain, có khả năng thay thế để nâng cao khả năng bảo trì

Các mô hình kiến trúc

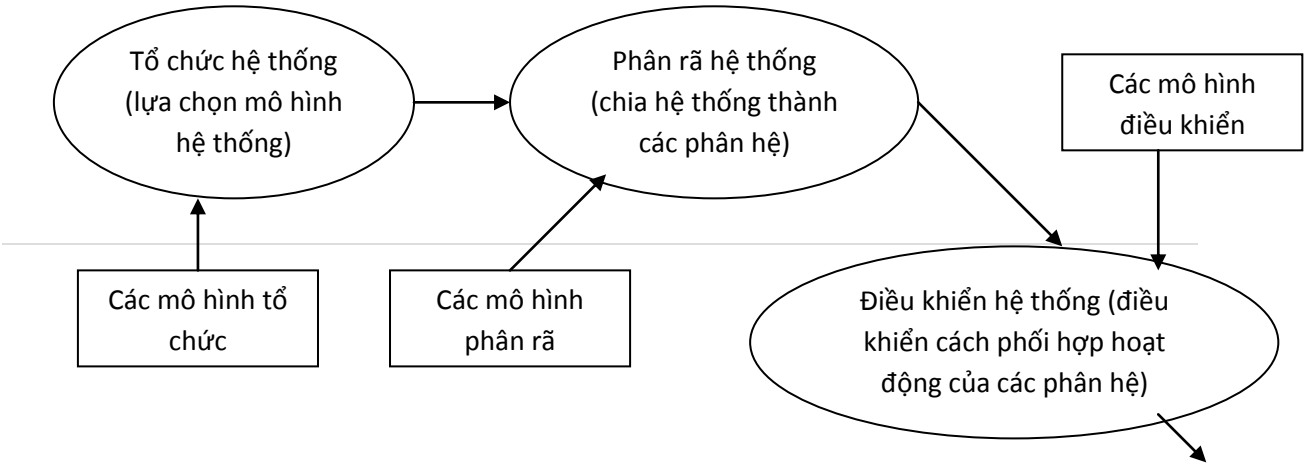
Mô hình kiến trúc của một hệ thống cụ thể có thể tuân theo kiểu kiến trúc tổng thể. Hiểu biết về các kiểu kiến trúc tổng thể này giúp ta dễ dàng hơn trong việc định nghĩa kiến trúc hệ thống. Tuy nhiên, hầu hết các hệ thống lớn là hỗn tạp, không đồng nhất và không tuân theo một kiểu kiến trúc đơn lẻ nào đó.

Mô hình kiến trúc được sử dụng để biểu diễn hoạt động thiết kế kiến trúc. Mỗi mô hình kiến trúc chỉ ra một khía cạnh về thiết kế kiến trúc:

- ✓ **Mô hình cấu trúc tĩnh**: Chỉ ra các thành phần chính của hệ thống.

- ✓ **Mô hình tiến trình động**: Chỉ ra cấu trúc tổ chức các tiến trình của hệ thống.
- ✓ **Mô hình giao diện**: Định nghĩa các giao diện hệ thống con.
- ✓ **Mô hình mối quan hệ**: Ví dụ mô hình luồng dữ liệu, chỉ ra các mối quan hệ hệ thống con.
- ✓ **Mô hình phân tán**: Chỉ ra cách thức các hệ thống con được phân tán trên các máy tính như thế nào.

Các hoạt động trong quy trình thiết kế kiến trúc



5.1.2.2 Tổ chức hệ thống

Tổ chức hệ thống là hoạt động đầu tiên phải thực hiện quá trình thiết kế kiến trúc. Mục đích nhằm xây dựng (lựa chọn) mô hình tổ chức hệ thống. Có 3 phương pháp tổ chức hệ thống thường được sử dụng:

- Kho dữ liệu dùng chung
- Server và các dịch vụ dùng chung (client-server)
- Phân lớp hoặc máy trừu tượng

a) Kho dữ liệu dùng chung

Các hệ thống con cần trao đổi thông tin với nhau. Việc trao đổi dữ liệu được thực hiện theo hai cách:

- Mỗi phân hệ duy trì một cơ sở dữ liệu riêng của mình. Dữ liệu được trao đổi giữa các phân hệ bằng cách chuyển qua các thông báo.

- Mọi dữ liệu được lưu trữ tại một cơ sở dữ liệu trung tâm, các phân hệ trong hệ thống có thể truy cập CSDL này. Mô hình này gọi là mô hình kho dữ liệu dùng chung.

=> Nếu số lượng dữ liệu dùng chung rất lớn thì mô hình kho dữ liệu dùng chung thường được sử dụng phổ biến nhất.

*** Ưu điểm của mô hình này là:**

- Đây là phương pháp hiệu quả để chia sẻ số lượng lớn dữ liệu mà không cần chuyển đổi dữ liệu từ phân hệ này sang phân hệ khác

- Các hệ thống con không cần quan tâm tới những hoạt động liên quan đến dữ liệu như: sao lưu, bảo mật, điều khiển truy cập và khôi phục … vì đã có bộ quản lý trung tâm thực hiện nhiệm vụ này

- Phân hệ tạo dữ liệu không cần lên quan đến việc các phân hệ khác sử dụng dữ liệu ntn?

*** Nhược điểm**

- Tất cả các hệ thống con phải chấp nhận mô hình kho dữ liệu, mà thực tế các phân hệ khác nhau có thể có các yêu cầu khác nhau về mức độ bảo mật, khôi phục, sao lưu, ...dữ liệu.

- Việc phân tán dữ liệu tới các hệ thống con sẽ gặp khó khăn.

b) Mô hình khách – phục vụ (client – server)

Mô hình khách - phục vụ là một mô hình hệ thống phân tán, biểu diễn việc phân tán các dữ liệu và xử lý trên nhiều máy tính khác nhau.

Các thành phần chính của mô hình là:

- Một tập các server độc lập phục vụ cho các phân hệ bằng cách cung cấp dịch vụ như: in ấn, quản lý dữ liệu, ...
- Một tập các khách hàng(client – phân hệ) truy nhập đến server để yêu cầu cung cấp dịch vụ.
- Hệ thống mạng cho phép client truy cập tới dịch vụ mà server cung cấp.

Client phải biết tên của server và các dịch vụ mà server cung cấp. Nhưng server thì không cần xác định rõ client và hiện tại có bao nhiêu client. Client tạo ra một yêu cầu tới server và chờ server trả lời.

Ưu điểm của mô hình client - server là:

- Phân tán dữ liệu rõ ràng

- Sử dụng các hệ thống được kết nối mạng một cách hiệu quả và chi phí dành cho phần cứng có thể rẻ hơn.

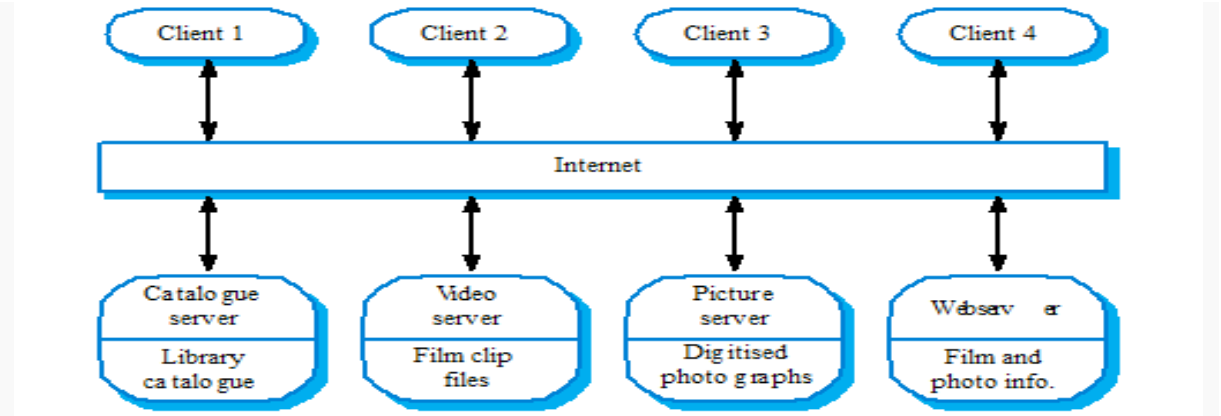
- Dễ dàng bổ sung hoặc nâng cấp server

Nhược điểm của mô hình client - server là:

- Không phải là mô hình dữ liệu dùng chung nên các hệ thống con có thể sử dụng các tổ chức dữ liệu khác nhau. Do đó, việc trao đổi dữ liệu có thể không hiệu quả vì phải chuyển đổi

- Quản lý mỗi server không thống nhất, dư thừa (nhiều sever cung cấp cùng một dịch vụ, ...)

- Không đăng ký tên và dịch vụ tập trung. Điều này làm cho việc tìm kiếm server hoặc các dịch vụ rất khó khăn.

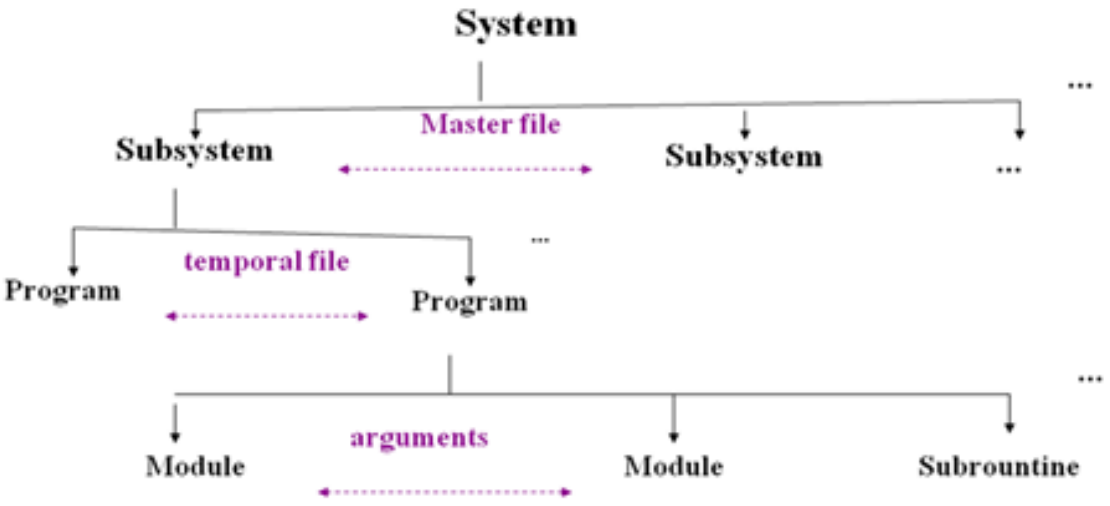


Hình 7.1: Mô hình client –server của hệ thống thư viện phim và ảnh

c) Mô hình phân lớp

Mô hình phân lớp tổ chức hệ thống thành nhiều lớp và mỗi lớp cung cấp một tập các dịch vụ. Mỗi lớp có thể được coi như một máy trừu tượng (abstract machine) mà ngôn ngữ của máy được định nghĩa bởi các dịch vụ mà lớp đó cung cấp.

5.1.2.3 Phân rã hệ thống



Sau khi cấu trúc hệ thống đã được lựa chọn, ta cần phải xác định phương pháp phân rã các hệ thống con thành các mô-đun.

Hệ thống con là một hệ thống có thể vận hành một cách độc lập, có thể sử dụng một số dịch vụ được cung cấp bởi các hệ thống con khác hoặc cung cấp dịch vụ cho các hệ thống con khác sử dụng.

Mô-đun là một thành phần của hệ thống cung cấp các dịch vụ cho các thành phần khác, nhưng nó thường không được coi như là một hệ thống riêng rẽ, độc lập.

Có hai cách để phân rã các hệ thống con thành các mô-đun:

- Phân rã hướng đối tượng: hệ thống được phân rã thành các đối tượng tương tác với nhau.

- Pipeline hướng chức năng hoặc luồng dữ liệu: hệ thống được phân rã thành các mô-đun chức năng chịu trách nhiệm chuyển đổi thông tin đầu vào thành kết quả đầu ra.

a) Phân rã hướng đối tượng

Mô hình kiến trúc hướng đối tượng cấu trúc hệ thống thành một tập hợp các đối tượng gắn kết lỏng dựa trên các giao diện (lớp) đã được định nghĩa.

Phân rã hướng đối tượng liên quan tới việc xác định lớp đối tượng, các thuộc tính và phương thức của nó. Khi cài đặt lớp, các đối tượng sẽ được tạo ra từ các lớp này và có một số mô hình điều khiển được sử dụng để kết hợp các phương thức của đối tượng.

Ưu điểm của mô hình hướng đối tượng:

- Đối tượng được gắn kết lỏng nên khi thay đổi cách cài đặt chúng có thể không ảnh hưởng tới các đối tượng khác.

- Đối tượng phản ánh thực thể trong thế giới thực.

- Các ngôn ngữ lập trình hướng đối tượng được sử dụng rộng rãi.

Hạn chế:

Khi giao diện của các thực thể trong thế giới thức hay thay đổi, phức tạp có thể gây khó khăn vì rất khó biểu diễn các thực thể như là các đối tượng.

b) Pipeline hướng chức năng

Hệ thống được phân hoá thành các module chức năng. Chúng nhận các dữ liệu chuyển hoá chúng rồi lại đưa ra các dữ liệu kết quả.

Trong mô hình luồng dữ liệu, các bộ biến đổi xử lý dữ liệu đầu vào và tạo dữ liệu đầu ra. Dữ liệu được chảy tuần tự theo luồng từ bộ biến đổi này sang bộ khác. Mỗi bước của quy trình giống như một phép biến đổi.

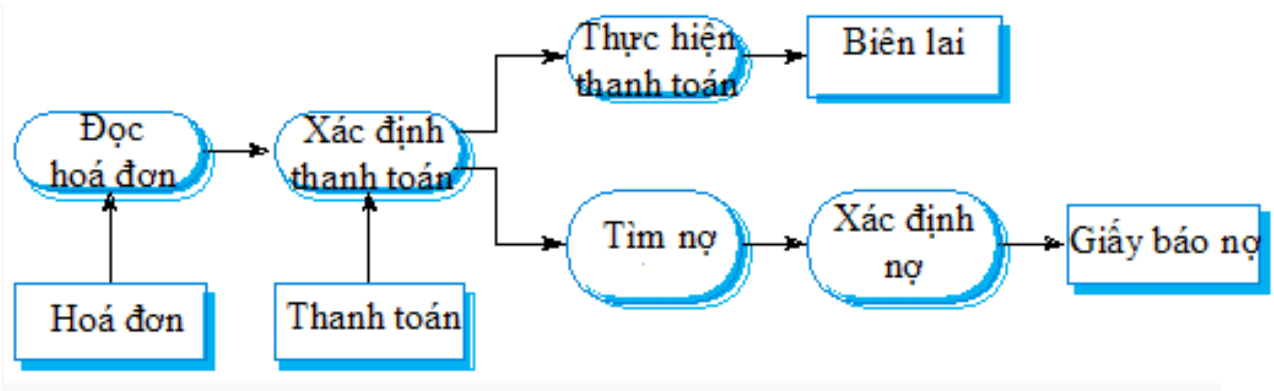
Mô hình này có ưu điểm:

- Nó hỗ trợ việc sử dụng lại các biến đổi.
- Nó phù hợp với suy nghĩ của mọi người quan niệm về dữ liệu được xử lý theo luồng có đầu vào và đầu ra.
- Thêm các xử lý khác vào hệ thống đơn giản.
- Dễ thực hiện xử lý song song hoặc tuần tự.

Nhược điểm của mô hình này là:

- Cần phải có một định dạng chung cho các dữ liệu để có thể xử lý bởi mọi bộ biến đổi.
- Các hệ thống tương tác khó được viết theo mô hình luồng dữ liệu

Ví dụ: Mô hình luồng dữ liệu của hệ thống xử lý hoá đơn



5.1.2.4 Các mô hình điều khiển

Các mô hình cấu trúc hệ thống có liên quan tới cách phân rã hệ thống thành nhiều hệ thống con. Để hệ thống làm việc tốt (đồng bộ và đúng), ta phải điều khiển được hoạt động của các hệ thống con

Có 2 loại chiến lược điều khiển:

- **Điều khiển tập trung:** một hệ thống con chịu trách nhiệm kiểm soát, khởi tạo hoặc dừng các hệ thống con khác.

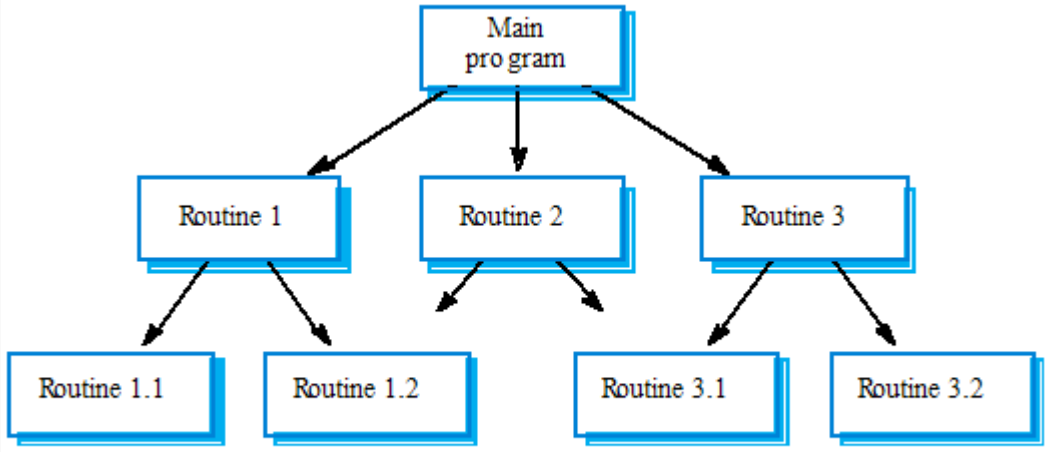
- **Điều khiển hướng sự kiện**: mỗi hệ thống đáp ứng với các sự kiện xảy ra từ các hệ thống con khác hoặc từ môi trường của hệ thống.

a) Điều khiển tập trung

Hệ thống con điều khiển chịu trách nhiệm quản lý việc thực hiện của các hệ thống con khác. Chiến lược điều khiển tập trung gồm 2 loại mô hình:

1. Mô hình gọi - trả lời (call-return)

Mô hình này phù hợp với các mô hình thủ tục top - down.



Hình 7.3: Mô hình gọi - trả lời

2. Mô hình quản lý

Thường áp dụng cho các hệ thống song song. Một cấu thành hệ thống được thiết kế như là một bộ quản trị và điều khiển việc khởi động, kết thúc và phối hợp các phân hệ khác.

b) Điều khiển hướng sự kiện

Mô hình hệ thống điều khiển bởi sự kiện có nhiều kiểu khác nhau như :

- **Mô hình phát tin**:
  - Trong mô hình này, về nguyên tắc, một sự kiện được thông báo cho các phân hệ. Các phân hệ được thiết kế để điều khiển sự kiện này sẽ tự quyết việc trả lời.
  - Mô hình này hiệu quả với các phân hệ được phân bố trên các máy tính khác nhau trên mạng.
  - Ưu điểm của nó là việc phát triển tương đối đơn giản. Một phân hệ mới xử lý một lớp sự kiện mới có thể được tích hợp khi ghi nhận các sự kiện này vào bộ điều khiển sự kiện.
  - Mỗi phân hệ có thể kích hoạt mọi phân hệ khác không cần biết tên và vị trí của các phân hệ đó.
  - Nhược điểm của mô hình này là phân hệ không biết sự kiện có được xử lý hay không và khi nào được xử lý. Rất có thể hai phân hệ khác nhau cùng sinh một sự kiện và có thể gây xung đột.
- **Mô hình điều khiển ngắt:**
  - Có một hệ thống bên ngoài được sử dụng riêng cho việc theo dõi

các ngắt bên ngoài và được chuyển tới các phân hệ tương ứng.

- Mô hình này phù hợp với các hệ thống thời gian.
- Ưu điểm của nó là cho phép đáp ứng nhanh nhất với các sự kiện.
- Nhược điểm là việc lập trình phức tạp.

5.1.2.5 Các kiến trúc tham chiếu

Có 2 loại mô hình kiến trúc cho một miền ứng dụng cụ thể:

- Mô hình chung: là sự trừu tượng hoá từ một số các hệ thống thực và bao hàm các thuộc tính quan trọng của các hệ thống này. Mô hình chung là mô hình bottom-up.

- Mô hình tham chiếu: là mô hình trừu tượng hoá và lý tưởng. Các mô hình tham chiếu thường kế thừa từ những nghiên cứu về miền ứng dụng hơn là từ các hệ thống đang tồn tại. Mô hình tham chiếu là mô hình top-down. Mô hình tham chiếu được sử dụng như một phần cơ bản để cài đặt hệ thống hoặc để so sánh với các hệ thống khác. Nó đóng vai trò là một mô hình chuẩn để đánh giá các hệ thống khác.

5.1.3 Thiết kế giao diện người dùng

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.

Trong mục thiết kế giao diện này, chúng ta sẽ nghiên cứu những vấn đề sau:

- Các yếu tố liên quan đến giao diện người dùng

- Quy trình xây dựng giao diện người dùng

5.1.3.1 Giao diện người dùng

a) 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 …

b) Các nguyên tắc thiết kế giao diên

Thiết kế giao diện phải phụ thuộc vào yêu cầu, kinh nghiệm và khả năng của người sử dụng hệ thống. Người thiết kế cũng nên quan tâm đến những giới hạn vật lý và tinh thần của con người và nên nhận ra rằng con người luôn có thể gây ra lỗi.

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 hoá 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 xoá, 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 dung khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị.

Tương tác giữa người sử dụng và hệ thống được chia thành 5 loại sau:

- Vận hành trực tiếp

- Lựa chọn menu

- Điền vào biểu mẫu (Form)

- Ngôn ngữ ra lệnh

- Ngôn ngữ tự nhiên

c) Biểu diễn thông tin

Biểu diễn thông tin có liên quan tới việc hiển thị các thông tin trong hệ thống tới người sử dụng. Thông tin có thể được biểu diễn một cách trực tiếp hoặc có thể được chuyển thành nhiều dạng hiển thị khác như: dạng đồ hoạ, âm thanh …

Thông tin cần biểu diễn được chia thành hai loại:

- Thông tin tĩnh: được khởi tạo ở đầu của mỗi phiên. Nó không thay đổi trong suốt phiên đó và có thể là ở dạng số hoặc dạng văn bản.

- Thông tin động: thay đổi trong cả phiên sử dụng và sự thay đổi này phải được người sử dụng quan sát.

Các nhân tố ảnh hưởng tới việc hiển thị thông tin:

- Người sử dụng thích hiển thị một phần thông tin hay quan hệ dữ liệu?

- Giá trị của thông tin thay đổi nhanh như thế nào? Sự thay đổi đó có cần phải thể hiện ngay lập tức hay không?

- Người sử dụng có phải thực hiện các hành động để đáp ứng với sự thay đổi không?

- Có phải là giao diện vận hành trực tiếp không?

- Thông tin ở dạng văn bản hay dạng số? Các giá trị quan hệ có quan trọng không?

- Biểu diễn digital hay analogue?

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hoá dữ liệu. Trực quan hoá 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 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.

- Cẩn thận khi sử dụng các cặp màu.

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 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, xúc 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.

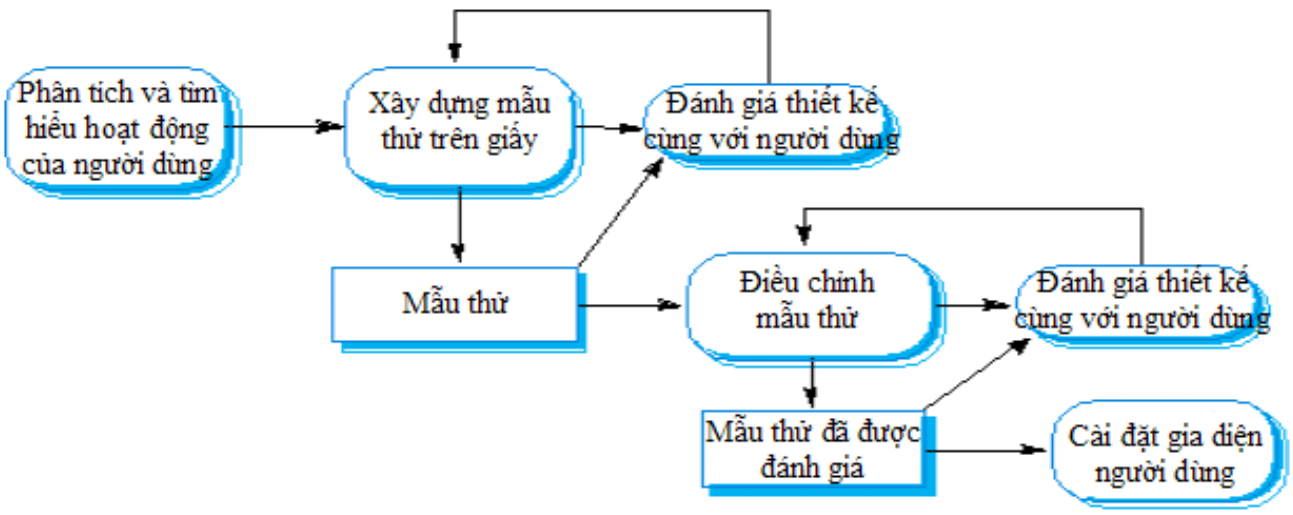
5.1.3.2 Quy trình thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một quy trình lặp lại bao gồm sự cộng tác giữa người sử dụng và người thiết kế. Trong quy trình này gồm 3 hoạt động cơ bản:

- Phân tích người sử dụng: tìm hiểu những gì người sử dụng sẽ làm với hệ thống.

- Lập mẫu thử hệ thống: xây dựng một tập các mẫu thử để thử nghiệm

- Đánh giá giao diện: thử nghiệm các mẫu thử cùng với người sử dụng.



Hình 8.3: Quy trình thiết kế gia diện người dùng

a) Phân tích người sử dụng

Nếu ta không hiểu rõ những gì người sử dụng muốn làm với hệ thống, thì ta sẽ không thể thiết kế được một giao diện hiệu quả. Phân tích người sử dụng phải được mô tả theo những thuật ngữ để người sử dụng và những người thiết kế khác có thể hiểu được.

Các ngữ cảnh mà ta mô tả thao tác ở trong đó là một trong những cách mô tả phân tích người dùng. Ta có thể lấy được rất nhiều yêu cầu của người sử dụng từ đó.

Các kỹ thuật phân tích:

- Phân tích nhiệm vụ: mô hình hoá các bước cần thực hiện để hoàn thành một nhiệm vụ.

- Phân tích nhiệm vụ phân cấp.

- Phỏng vấn và trắc nghiệm: hỏi người sử dụng về những gì mà họ làm. Khi phỏng vấn, chúng ta nên dựa trên những câu hỏi có kết thúc mở. Sau đó, người sử dụng cung cấp những thông tin mà họ nghĩ rằng nó là cần thiết; nhưng không phải tất cả các thông tin đó là có thể được sử dụng. Ngoài ra, chúng ta có thể thực hiện phỏng vấn với cả nhóm người sử dụng, điều đó cho phép người sử dụng thảo luận với nhau về những gì họ làm.

- Mô tả: quan sát người sử dụng làm việc và hỏi họ về những cách mà không được biết tới. Nên nhớ rằng có nhiều nhiệm vụ của người sử dụng thuộc về trực giác và rất khó để mô tả và giải thích chúng. Dựa trên kỹ thuật này ta có thể hiểu thêm về các ảnh hưởng xã hội và tổ chức tác động tới công việc đó.

b) Lập mẫu thử giao diện người dùng

Mẫu thử cho phép người sử dụng có được những kinh nghiệm trực tiếp với giao diện. Nếu không có những kinh nghiệm trực tiếp như vậy thì không thể đánh giá được khả năng có thể sử dụng được của giao diện.

Lập mẫu thử là một quy trình gồm 2 trạng thái:

- Lập các mẫu thử trên giấy.

- Tinh chỉnh mẫu thử và xây dựng chúng

Các kỹ thuật lập mẫu thử:

- Mẫu thử hướng nguyên mẫu: sử dụng công cụ như Macromedia Director để xây dựng một tập hợp các nguyên mẫu và màn hình. Khi người sử dụng tương tác với chúng thì màn hình sẽ thay đổi để hiển thị trạng thái kế tiếp.

- Lập trình trực quan: sử dụng các ngôn ngữ được thiết kế cho việc phát triển nhanh như Visual Basic.

- Mẫu thử dựa Internet: sử dụng web browser và script.

c) Đánh giá giao diện người dùng

Ta nên đánh giá bản thiết kế giao diện người dùng để xác định khả năng phù hợp của

nó. Tuy nhiên, việc đánh giá trên phạm vi rộng tốn nhiều chi phí và không thể thực hiện được đối với hầu hết các hệ thống.

Các kỹ thuật đánh giá đơn giản:

- Trắc nghiệm lại các phản hồi của người sử dụng

- Ghi lại quá trình sử dụng mẫu thử của hệ thống và đánh giá nó.

- Lựa chọn những thông tin về việc sử dụng dễ dàng và các lỗi của người sử dụng.

- Cung cấp mã lệnh trong phần mềm để thu thập những phản hồi của người sử dụng một cách trực tuyến.

5.1.4 Thiết kế cấu trúc dữ liệu

Hoạt động này nhằm lựa chọn và xây dựng mô hình biểu diễn thông tin cho hệ thống. Cách thức tổ chức dữ liệu của hệ thống cũng được đặc tả chi tiết dần ở các mức:

- ✓ Mô hình dữ liệu
- ✓ Cấu trúc dữ liệu

Chọn cách biểu diễn các đối tượng thiết kế có ảnh hưởng mạnh mẽ đến chất lượng phần mềm. Một số tiêu chí lựa chọn cách thức tổ chức dữ liệu của hệ thống**:**

- ✓ Thuận lợi cho các thao tác
- ✓ Tiết kiệm không gian
- ✓ Phản ánh đúng các dữ liệu thực tế

Việc chọn lựa cấu trúc dữ liệu ảnh hưởng lớn đến hiệu suất chương trình và nó tác động đến bản thân thuật toán bởi cấu trúc dữ liệu gắn bó mật thiết với thuật toán. Lựa chọn đúng đắn các cấu trúc dữ liệu giúp làm giảm không gian bộ nhớ của chương trình, giảm thời gian chạy, tăng tính khả chuyển và dễ bảo trì.

Một số nguyên tắc để làm giảm không gian lưu trữ dữ liệu:

- ✓ Đảm bảo tính đơn giản của cấu trúc lưu trữ,
- ✓ Trong một số trường hợp đừng lưu trữ, hãy tính lại khi cần thiết,
- ✓ Nén dữ liệu sau đó giải nén khi dùng,
- ✓ Sử dụng các nguyên tắc cấp phát bộ nhớ: chẳng hạn như cấp phát bộ nhớ động,...

Các mức thiết kế cấu trúc dữ liệu cho hệ thống:

Thiết kế cấu trúc logic	Thiết kế cấu trúc vật lý
✓ Các quan hệ chuẩn ✓ Các khóa ✓ Các tham chiếu ✓ Các cấu trúc thao tác dữ liệu	✓ Các file ✓ Các kiểu dữ liệu ✓ Kích cỡ

5.1.5 Thiết kế thuật toán/thủ tục

Sau khi đã xác định được các yêu cầu mà hệ thống phải đáp ứng, để xử lý các yêu cầu này ta lựa chọn cách xử lý tối ưu nhất (sử dụng thuật toán tốt nhất, làm tăng tốc độ thực thi chương trình, cho ra kết quả chính xác). Thiết kế thủ tục nhằm mô tả các bước hoạt động của từng mô đun. Các phương pháp mô tả có thể là:

- ✓ Giải mã (pseudo code)
- ✓ Sơ đồ luồng (flow chart)
- ✓ Biểu đồ (diagram)
- ✓ Biểu đồ hoạt động (activity diagram)
- ✓ ISP

Khi lựa chọn thuật toán sử dụng, chúng ta nên xác định rõ bài toán. Trước khi bắt tay vào giải bài toán, hãy tìm hiểu kỹ các yêu cầu mà bài toán đặt ra và tận dụng mọi điều đã biết từ bài toán. Các thuật toán có ảnh hưởng quan trọng đến các hệ thống phần mềm và đặc biệt chúng tăng nhanh tốc độ vận hành của chương trình.

Một số nguyên tắc làm tăng tốc độ vận hành của chương trình:

- ✓ Lưu trữ các trạng thái cần thiết để tránh tính lại,
- ✓ Tiền xử lý thông tin để đưa vào các cấu trúc dữ liệu,
- ✓ Sử dụng các thuật toán thích hợp,
- ✓ Sử dụng các kết quả được tích luỹ,...

5.2 Cài đặt hệ thống

5.2.1 Các yêu cầu đối với lập trình viên

- ✓ Có năng lực cá nhân, say mê, cần mẫn
- ✓ Hiểu biết các các phương pháp luận lập trình và các công cụ lập trình
- ✓ Nắm vững các nguyên lý lập trình

- ✓ Có kinh nghiệm lập trình

Lập trình viên tốt là người đảm bảo chương trình:

- ✓ Chạy đúng
- ✓ Dễ hiểu
- ✓ Dễ bảo trì & phát triển

5.2.2 Quá trình phát triển của kỹ thuật lập trình

Một kỹ thuật lập trình được xem là tốt nếu nó được chuyên nghiệp hóa (tuân theo các chuẩn), lập trình ổn định và hiệu quả. Sau đây là một số kỹ thuật lập trình:

5.2.2.1 Lập trình tuần tự/tuyến tính

Chương trình bao gồm một tập các câu lệnh được thực hiện một cách tuần tự. Không có các câu lệnh có cấu trúc/điều khiển trong chương trình (for, while, if, ...). Để thực hiện các hoạt động điều khiển chương trình thường lạm dụng các lệnh nhảy goto, thiếu khả năng khai báo các biến cục bộ trong chương trình. Kỹ thuật lập trình này có những hạn chế là độ ghép nối của chương trình là cao, điều này dẫn đến chương trình khó hiểu, khó sửa và dễ sinh lỗi.

Các ngôn ngữ thường dùng cho lập trình kỹ thuật này là các ngôn ngữ thế hệ 1, 2 như hợp ngữ, basic, ...

5.2.2.2 Lập trình có cấu trúc/thủ tục

Lập trình có cấu trúc là kỹ thuật lập trình được sử dụng khá phổ biến. Chương trình được chia thành nhiều module chương trình con, và cho phép sử dụng các biến cục bộ trong mỗi chương trình con. Mỗi khi chương trình cần thực hiện công việc của các mô đun này ta chỉ việc gọi đến chúng. Điều khiển chương trình sử dụng kỹ thuật này phong phú hơn so với kỹ thuật lập trình tuyến tính. Các ngôn ngữ lập trình trợ giúp kỹ thuật này cũng cung cấp khá nhiều lệnh điều khiển/có cấu trúc như các lệnh for, while, if, ...giúp ta điều khiển của hoạt động của chương trình. Kỹ thuật lập trình này hạn chế/cấm dùng lệnh goto. Kỹ thuật lập trình này phù hợp với kết quả phân tích thiết kế hướng chức năng. Chương trình sử dụng kỹ thuật lập trình này dễ hiểu hơn và an toàn hơn so với kỹ thuật lập trình trước đó.

Các ngôn ngữ lập trình thế hệ 2, 3 được dùng cho kỹ thuật lập trình này như Pascal, C, Fortran,...

5.2.2.3 Lập trình hướng hàm

Lập trình hướng hàm dựa trên nguyên tắc ghép nối dữ liệu. Việc trao đổi dữ liệu được thực hiện bằng cách truyền dữ liệu qua tham số và nhận giá trình trả về. Kỹ thuật

này loại bỏ toàn bộ các dữ liệu dùng chung. Chương trình sử dụng kỹ thuật lập trình là là một hàm. Nó được xây dựng để tính toán các biểu thức và thường ứng dụng trong các lĩnh vực tính toán và trí tuệ nhân tạo.

Ví dụ: Chương trình có cấu trúc và chương trình hướng hàm tương ứng

Chương trình có cấu trúc	Chương trình hướng hàm
Begin *NhapDL();* *XylyDL();* *XuatDL();* *End.*	Chương trình là một biểu thức/hàm *XuatDL(XulyDL(NhạpDL(...)))*

Ngôn ngữ lập trình cho kỹ thuật này là Lisp, Scheme, ...

5.2.2.4 Lập trình hướng đối tượng

Lập trình hướng đối tượng là xu hướng phát triển của phương pháp lập trình hiện đại. Phương pháp này hỗ trợ các khái niệm hướng đối tượng như kế thừa, bao gói, che dấu thông tin, ...Chương trình được cấu thành từ một tập hợp các lớp đối tượng. Các đối tượng thuộc lớp trao đổi thông tin với nhau qua việc truyền các thông điệp. Chương trình sử dụng kỹ thuật lập trình này có ưu điểm cục bộ dữ liệu và thao tác hơn, dễ tái sử dụng hơn, thuận tiện cho các ứng dụng lớn.

Các ngôn ngữ lập trình dùng cho kỹ thuật lập trình này như: Smalltalk, Java, C#

5.2.2.5 Lập trình logic

Lập trình logic là kỹ thuật lập trình tách tri thức về bài toán ra khỏi kỹ thuật lập trình. Lập trình là mô tả, xây dựng cơ sở tri thức bằng các quy tắc, mệnh đề, các mục tiêu, ... Khi người dùng đặt ra câu hỏi, chương trình chạy bằng cách tự chứng minh để tìm kiếm đường đi đến mục tiêu.

Kỹ thuật lập trình này thường được sử dụng để xây dựng các hệ chuyên gia, các ứng dụng xử lý Ngôn ngữ tự nhiên. Ngôn ngữ dùng cho kỹ thuật lập trình này như Prolog, …

5.2.2.6 Kỹ thuật lập trình thế hệ thứ 4

Lập trình bằng cách khai báo. Ví dụ lập trình CSDL. Ngôn ngữ lập trình dùng cho kỹ thuật này như SQL, ...chúng có năng lực biểu diễn cao và giúp lập trình với tốc độ cao.

5.2.3 Chọn ngôn ngữ lập trình cho ứng dụng

Khi chọn ngôn ngữ cho ứng dụng, ta cần quan tâm đến các yếu tố sau:

- ✓ *Theo yêu cầu của khách hàng*: Nếu khách tự bảo trì phầm mềm
- ✓ *Các đặc trưng của ngôn ngữ*
    - o ***Năng lực***: Khả năng hỗ trợ các kiểu biến, các cấu trúc lệnh. Những ngôn ngữ bậc cao thường có các cấu trúc và các câu lệnh phong phú. Chúng hỗ trợ nhiều kiểu dữ liệu, hỗ trợ con trỏ, hỗ trợ hướng đối tượng và có thư viện phong phú. Do đó, ta nên dùng các ngôn ngữ bậc cao thay cho bậc thấp nếu không quá cần thiết phải dùng ngôn ngữ bậc thấp
    - o ***Tính khả chuyển của ngôn ngữ***: Giúp chương trình thực hiện được trên nhiều máy tính, hệ điều hành khác nhau. Tính khả chuyển là yếu tố quan trọng của ngôn ngữ vì khi thay đổi phần cứng và thay đổi hệ điều hành chương trình vẫn có thể vận hành được. Java và các ngôn ngữ thông dịch (kịch bản, script) có tính khả chuyển. Để nâng cao tính khả chuyển của chương trình ta cần sử dụng các tính năng chuẩn của ngôn ngữ và dùng các kịch bản bất cứ khi nào có thể.
    - o Công cụ hỗ trợ của ngôn ngữ lập trình: Nên sử dụng ngôn ngữ có các công cụ hỗ trợ lập trình hiệu quả. Điều này giúp ta dễ dàng quá trình lập trình và bảo trì phần mềm sau này.
        - ▪ Trình biên dịch hiệu quả: Tốc độ biên dịch cao, khả năng tối ưu cao, có khả năng khai thác các tập lệnh và thiết bị phần cứng mới

      Có các công cụ trợ giúp hiệu quả: editor, debugger, linker, make...IDE (Integrated Dvelop Environment).
- ✓ Năng lực, kinh nghiệm của nhóm lập trình viên
    - o Chọn NN mà lập trình viên làm chủ
- ✓ Miền ứng dụng của ngôn ngữ
    - o Phần mềm hệ thống: Chương trình hiệu quả, vạn năng, dễ mở rộng. Ngôn ngữ hỗ trợ như C, C++
    - o Hệ thời gian thực: Ngôn ngữ C, C++, ADA, Assembly
    - o Hệ thống nhúng: C++, Java, ...
    - o Phần mềm nghiệp vụ: Cơ sở dữ liệu (phần mềm Oracle, DB2, SQL Server, MySQL, ...), Ngôn ngữ lập trình như FoxPro, Cobol, VB, VC++...

- Trí tuệ nhân tạo: Prolog, Lips, OPS5, ...
- Mạng: .Net, Các công nghệ Java
- Web: PHP, ASP, JSP, Perl, Java, Java Script, ...
- Phần mềm khoa học kỹ thuật: cần tính toán chính xác, thư viện toán học mạnh, dễ dàng song song hóa. Fortran là ngôn ngữ được dùng phổ biến

✓ Phương pháp luận lập trình.

=> Chưa tồn tại ngôn ngữ đa năng cho mọi ứng dụng

5.2.4 Một số nguyên tắc lập trình

✓ Đặt tên: Có ý nghĩa, gợi nhớ

✓ Trình bày: Rõ ràng, có cấu trúc, dễ hiểu

✓ Chú thích: Đầy đủ, dễ đọc

✓ Hạn chế sử dụng các cấu trúc khó kiểm soat: break, goto, continue, ...

✓ Viết lệnh rõ ràng, tránh ẩn ý

✓ Triển khai các biểu thức phức tạp

✓ Mỗi module nên giới hạn từ 25 -> 50 dòng

✓ Tránh sử dụng các số tối nghĩa

✓ Viết chương trình theo cấu trúc nhô thụt, mỗi lệnh/1 dòng

CHƯƠNG 6. KIỂM THỬ PHẦN MỀM

MỤC ĐÍCH

- Nắm được quy trình kiểm thử phần mềm

- Tìm hiểu chi tiết về kiểm thử thành phần và kiểm thử hệ thống; các phương pháp được sử dụng.

- Có khả năng thiết kế các trường hợp kiểm thử và sử dụng các công cụ giúp tự động kiểm thử

6.1 Xác minh và thẩm định phần mềm

Xác minh và thẩm định một hệ thống phần mềm là một quá trình liên tục xuyên suốt mọi giai đoạn của quá trình phần mềm. Xác minh và thẩm định là từ chung cho các quá trình kiểm thử để đảm bảo rằng phần mềm thỏa mãn các yêu cầu của chúng và các yêu cầu đó thỏa mãn các nhu cầu của người dùng phần mềm.

Xác minh và thẩm định là một quá trình kéo dài suốt vòng đời. Nó bắt đầu khi duyệt yêu cầu. Xác minh và thẩm định có hai mục tiêu:

- Phát hiện các khiếm khuyết trong hệ thống
- Đánh giá xem hệ thống liệu có dùng được hay không?

Sự khác nhau giữa xác minh và thẩm định là:

- Xác minh (*Verification)* Xem xét cái được xây dựng có đúng là sản phẩm không? Như thế, xác minh là kiểm tra chương trình có phù hợp với đặc tả hay không?
- Thẩm định (*Validation*) Xem xét cái được xây dựng có là sản phẩm đúng không? Tức là kiểm tra xem chương trình có được như mong đợi của người dùng hay không?

Như trên, kiểm thử là quá trình của tìm kiếm lỗi. Một kiểm thử tốt có khả năng cao tìm kiểm các lỗi chưa được phát hiện. Một kiểm thử thành công là kiểm thử tìm ra các lỗi mới, một kiểm tra tồi là kiểm tra mà không tìm được lỗi.

6.2 Kiểm thử phần mềm

Theo IEEE: Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống hoặc thành phần đó.

Theo Myers (the art of software testing): Kiểm thử là tiến trình thực thi chương trình với mục đích tìm thấy lỗi.

Mục đích của kiểm thử phầm mềm là tìm lỗi.

Mục địch của gỡ rối là xác định bản chất lỗi và định vị lỗi trong chương trình và tiến hành sửa lỗi.

Thế nào là lỗi phần mềm?

Có rất nhiều định nghĩa khác nhau về lỗi phần mềm, nhưng tựu chung, có thể phát biểu một cách tổng quát: *"**Lỗi phần mềm** là sự không khớp giữa chương trình và đặc tả của nó."*

Dựa vào định nghĩa, chúng ta có thể thấy lỗi phần mềm xuất hiện theo ba dạng sau:

Sai: Sản phẩm được xây dựng khác với đặc tả.

Thiếu: Một yêu cầu đã được đặc tả nhưng lại không có trong sản phẩm được xây dựng.

Thừa: Một yêu cầu được đưa vào sản phẩm mà không có trong đặc tả. Cũng có trường hợp yêu cầu này có thể là một thuộc tính sẽ được người dùng chấp nhận nhưng khác với đặc tả nên vẫn coi là có lỗi.

Một hình thức khác nữa cũng được xem là lỗi, đó là phần mềm khó hiểu, khó sử dụng, chậm hoặc dễ gây cảm nhận rằng phần mềm hoạt động không đúng.

Phân biệt một số khái niệm: error, fault, failure

Bug bao gồm: Error, fault, failure

- *Error (mistake – sai sót):* là người thực hiện đã phạm phải trong quá trình thực hiện trở thành nguyên nhân gây ra lỗi
- *Fault (lỗi):* xuất hiện trong phần mềm như là kết quả của một sai sót hay nó là nguyên nhân gây ra lỗi của chương trình
- *Failure (hỏng hóc):* nó là kết quả của một lỗi xuất hiện làm cho chương trình không hoạt động được hay hoạt động nhưng cho kết quả không như mong đợi.

Lỗi phần mềm xảy ra khi nào? Ở đâu?

Lỗi xuất hiện ở mọi nơi trong quy trình phát triển phần mềm: xen lẫn trong quá trình Requirement Definition →Logial-Physical Design →Program Development

Nguyên nhân gây ra lỗi:

+ Hiểu sai requirement

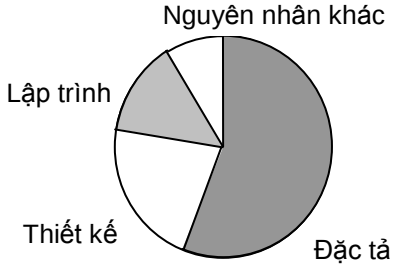
+ Thiếu cân nhắc specification

+ Design không chính xác

+ Implement không tốt

Khác với sự cảm nhận thông thường, lỗi xuất hiện nhiều nhất không phải do lập trình. Nhiều nghiên cứu đã được thực hiện trong các dự án từ rất nhỏ đến các dự án rất

lớn và kết quả luôn giống nhau. Số lỗi do đặc tả gây ra là nhiều nhất, chiếm khoảng 80%. Có một số nguyên nhân làm cho đặc tả tạo ra nhiều lỗi nhất. Trong nhiều trường hợp, đặc tả không được viết ra. Các nguyên nhân khác có thể do đặc tả không đủ cẩn thận, nó hay thay đổi, hoặc do chưa phối hợp tốt trong toàn nhóm phát triển. Sự thay đổi yêu cầu của khách hàng cũng là nguyên nhân dễ gây ra lỗi phần mềm. Khách hàng thay đổi yêu cầu không cần quan tâm đến những tác động sau khi thay đổi yêu cầu như phải thiết kế lại, lập lại kế hoạch, làm lại những việc đã hoàn thành. Nếu có nhiều sự thay đổi, rất khó nhận biết hết được phần nào của dự án phụ thuộc và phần nào không phụ thuộc vào sự thay đổi. Nếu không giữ được vết thay đổi rất dễ phát sinh ra lỗi.



Hình 1 – Các nguyên nhân gây ra lỗi phần mềm

Ví dụ: Trong đặc tả bài toán phân số:

Với việc đặc tả phi hình thức: Phân số là một cặp t/m, trong đó t là một số nguyên, m là một số tự nhiên lớn hơn 0; t được gọi là tử số, m được gọi là mẫu số của phân số.

Đặc tả hình thức là đặc tả trong đó sử dụng các ký hiệu toán học để mô tả. Một phân số có thể được đặc tả như sau:

\+ Phân số = $\{(t,m) \mid t \in Z, m \in N^+\}$ (*)

Trong đó: $N = \{0, 1, 2, 3, \ldots\}$

$N^+ = \{1, 2, 3, \ldots\}$

$Z = \{0, \pm1, \pm2, \pm3, \ldots\}$

\+ Phép chia hai phân số: (t1,m1):(t2,m2) = Reduce(t1× m2, t2 × m1), trong đó Reduce(t, m) = (t/d, m/d) với d = gcd(t, m). Hàm gcd là hàm tìm ước số chung lớn nhất của hai số tự nhiên.

Đặc tả phép chia như trên là sai với đặc tả phân số (*). Chẳng hạn, thực hiện chia hai phân số (1,3):(-2,5) = (5,-6), thì mẫu số trong trường hợp này lại là một số âm, không đúng đặc tả. Đây là một ví dụ đơn giản về việc đặc tả sai do không đủ cẩn thận. Với các bài toán lớn thì việc đặc tả sẽ rất khó và dễ nhầm lẫn, sai sót.

Nguồn gây ra lỗi lớn thứ hai là thiết kế. Đó là nền tảng mà lập trình viên dựa vào để nỗ lực thực hiện kế hoạch cho phần mềm.

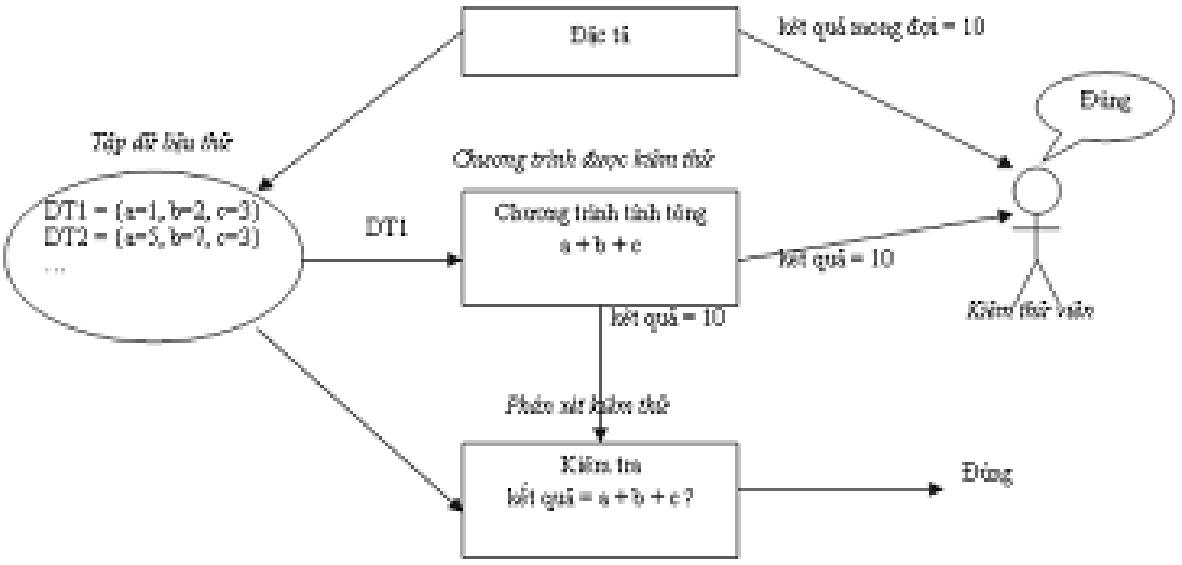
Lỗi do lập trình gây ra cũng khá dễ hiểu. Ai cũng có thể mắc lỗi khi lập trình. Thời kì đầu, phát triển phần mềm có nghĩa là lập trình, công việc lập trình thì nặng nhọc, do đó lỗi do lập trình gây ra là chủ yếu. Ngày nay, công việc lập trình chỉ là một phần việc của quá trình phát triển phần mềm, cộng với sự hỗ trợ của nhiều công cụ lập trình cao cấp, việc lập trình trở nên nhẹ nhàng hơn, mặc dù độ phức tạp phần mềm lớn hơn rất nhiều. Do đó, lỗi do lập trình gây ra cũng ít hơn. Tuy nhiên, nguyên nhân để lập trình tạo ra lỗi lại nhiều hơn. Dó là do độ phức tạp của phần mềm, do tài liệu nghèo nàn, do sức ép thời gian hoặc chỉ đơn giản là những lỗi "không nói lên được". Một điều cũng hiển nhiên là nhiều lỗi xuất hiện trên bề mặt lập trình nhưng thực ra lại do lỗi của đặc tả hoặc thiết kế.

Một nguyên nhân khác tạo ra lỗi là do bản thân các công cụ phát triển phần mềm cũng có lỗi như công cụ trực quan, thư viện lớp, bộ biên dịch,…

Một số khái niệm:

- ✓ **Dữ liệu thử (test data):** là các dữ liệu đầu vào cung cấp cho chương trình trong khi thực thi
- ✓ **Kịch bản kiểm thử (test scenario)**: là các bước thực hiện khi tiến hành kiểm thử
- ✓ **Phán xét kiểm thử (test oracle)**: là hoạt động nhằm đánh giá kết quả kiểm thử. Hoạt động này có thể tiến hành tự động bằng chương trình máy tính hoặc tiến hành thủ công bởi con người.
- ✓ **Kiểm thử viên (tester)**: là người tiến hành hoạt động kiểm thử
- ✓ **Ca kiểm thử (test case)**: Một ca kiểm thử bao gồm:
    - Tập dữ liệu thử
    - Điều kiện thực thi
    - Kết quả mong đợi

Ví dụ về ca kiểm thử:



6.3 Những nguyên tắc trong kiểm thử phần mềm

- Lấy yêu cầu của khách hàng làm tiêu chí hàng đầu. Như chúng ta đã biết, mục đích của công việc kiểm tra là phát hiện lỗi. Nhưng lỗi đó, theo quan điểm của người sử dụng, đơn giản là do không đáp ứng được yêu cầu của họ.

a) Không thể kiểm tra từ A_Z.

Nếu là một người kiểm tra (tester), chúng ta tin tưởng (ít ra là hy vọng) có thể tiếp cận, kiểm tra toàn bộ, tìm ra tất cả lỗi để hoàn thiện chương trình. Nhưng thật đáng tiếc, đó là một Nhiệm vụ bất khả thi.

Có 3 nguyên nhân chính dẫn đến điều đó:

- Khối lượng dữ liệu vào và ra quá lớn.
- Số lượng lớn các mối quan hệ giữa các bộ phận của chương trình.
- Bảng hướng dẫn kỹ thuật mang tính chủ quan và lỗi hay không còn tuỳ thuộc vào quan điểm của người sử dụng.

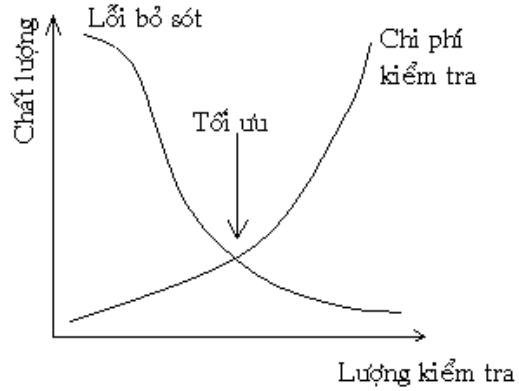
b) Không tìm ra lỗi không đồng nghĩa là chương trình không có lỗi, nhưng tìm được rất nhiều lỗi cũng không có nghĩa là đã hết lỗi.

c) Kế hoạch kiểm tra phải được lập một cách kỹ lưỡng, có thể là ngay từ khi phân tích, thiết kế. Trong quá trình thực thi, phải luôn thay đổi trọng tâm, phương pháp để sao cho tìm được lỗi càng sớm càng tốt một cách có hệ thống. Điều đó tạo tiền đề cho việc xác định, cô lập các module, cụm tích hợp bị lỗi hoặc đang bị nghi ngờ.

d) Quyết định thời điểm dừng thích hợp:

Như đã nói ở trên, việc kiểm tra tất cả từ A_Z là không khả thi. Nhưng nếu không kiểm tra tất cả thì sẽ có một số lỗi bị bỏ sót. Vấn đề được đặt ra ở đây là phải biết dừng lúc nào để hạn chế tối đa lỗi bị bỏ sót, tiết kiệm chi phí và đáp ứng đúng nhu

cầu thời gian.



Mối quan hệ giữa khối lượng kiểm tra và lỗi được phát hiện.

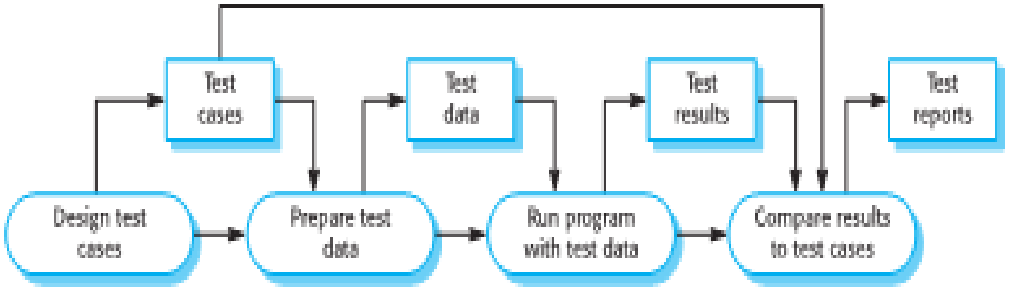
e) Để đạt hiệu quả cao, công việc kiểm tra tốt nhất nên được thực hiện bởi một nhóm thứ 3 độc lập với công việc thiết kế và cài đặt. Vì người cài đặt không phải luôn luôn là người thích hợp nhất cho việc kiểm tra.

*** Chú ý: Không phải tất cả các lỗi được phát hiện đều được sửa chữa.** Điều đó không có nghĩa là người ta chấp nhận một sản phẩm tồi mà là vì công việc sửa chữa còn phụ thuộc vào nhiều yếu tố như kinh phí, thời gian, mức độ rủi ro,…

6.4 Quy trình kiểm thử

Kiểm thử thường bao gồm các bước:

- Thiết kế các ca kiểm thử
- Bước tạo dữ liệu thử
    - Kiểm thử với tất cả các dữ liệu vào là cần thiết
        - Không thể kiểm thử "vét cạn"
    - Chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào
        - Dựa trên các tiêu chuẩn chọn dữ liệu thử
- Bước thực thi chương trình trên dữ liệu thử
    - Cung cấp dữ liệu thử
    - Thực thi
    - Ghi nhận kết quả
- Bước quan sát kết quả kiểm thử
    - Thực hiện trong khi hoặc sau khi thực thi
    - So sánh kết quả nhận được và kết quả mong đợi



Ví dụ: Cho 1 chương trình tìm số lớn nhất trong 1 dãy số a gồm n số.

* Thiết kế ca kiểm thử

- Các đầu vào (Input)

N: Integer;

A={a[i] | a[i] thuộc Z; i=1->n}

- Điều kiện thử (conditions)

- Các đầu ra (outputs): max = a[i] | a[i]>=a[j]; i=1 -> n; j= 1 ->n

* Chuẩn bị dữ liệu thử

Td1={n=-5; a={};}

Td2={n=5; a={7,8,3,6,25};

* Chạy chương trình

Max = ; (kết quả thực sự mong đợi chạy với td2);

* Phán xét: -> Chương trình chạy sai

-> Chương trình chạy đúng

6.5 Các mức kiểm thử phần mềm

Kiểm thử phần mềm có mặt tại tất cả các quá trình phát triển phần mềm và do các cá nhân khác nhau thực hiện trong quá trình thực hiện ứng dụng. Đối với các dự án lớn, người tham gia kiểm thử được chia làm 2 nhóm:

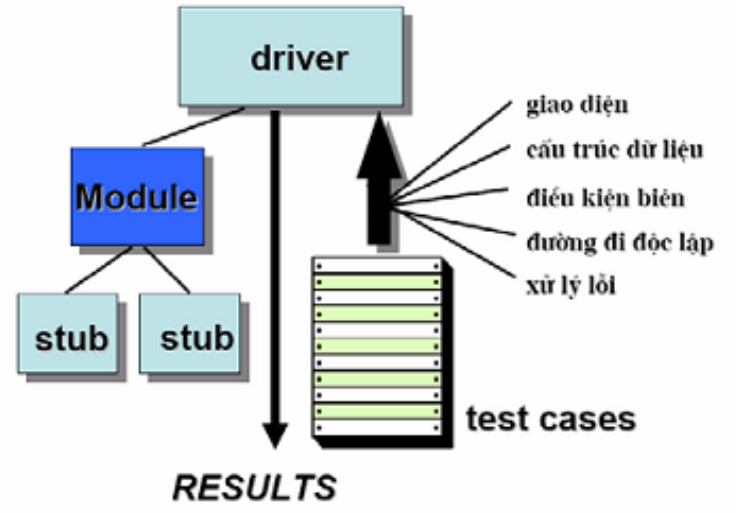
**** Kiểm thử được tiến hành bởi đội ngũ dự án được gọi là kiểm thử phát triển (Development tesingt) bao gồm:***

6.5.1 Kiểm thử đơn vị (Unit testing)

Được tiến hành cho mỗi đơn vị mã nhỏ nhất như hàm, module. Vì đơn vị được kiểm tra không phải là chương trình đầy đủ, hơn nữa đơn vị này có thể được gọi đến bởi đơn vị khác hoặc gọi đến đơn vị khác. Chúng ta xây dựng các module giả lập đơn vị gọi tên là driver và đơn vị gọi là stub.

Driver như là chương trình chính nhập các bộ số thử nghiệm và gửi chúng đến đơn vị cần kiểm tra, đồng thời nhận kết quả của đơn vị trả về.

Stub là chương trình giả lập thay thế các đơn vị được gọi đến đơn vị cần kiểm tra. Stub xử lý các thao tác đơn giản như in ấn, kiểm tra dữ liệu nhập và trả lại kết quả ra.



6.5.2 Kiểm thử tích hợp (Integration testing)

Sau khi kiểm thử đơn vị hoàn thành. Các module đảm bảo hoạt động tốt thì ta bắt đầu tích hợp các module xem chúng hoạt động ra sao.

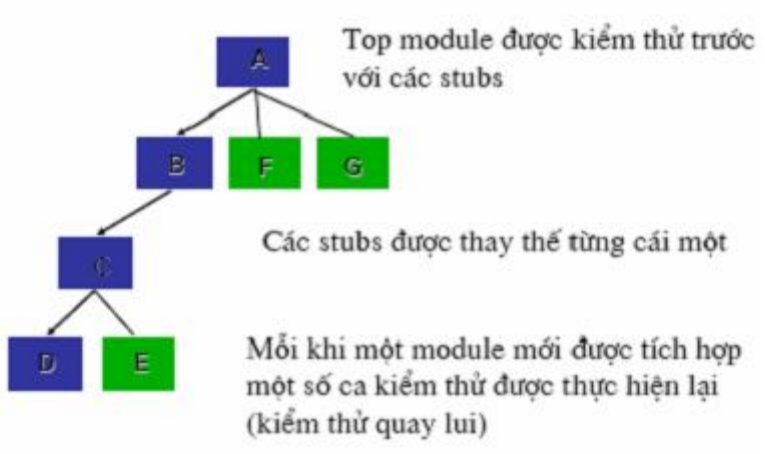
Kiểm thử tích hợp mục đích phát hiện các lỗi về mặt logic và xử lý của các khối, kiểm thử việc truyền tin giữa chúng. Kiểm thử viên sẽ tiến hành kiểm thử giao diện, sự tương tác giữa các thành phần, module, cửa sổ.

Các chiến lược kiểm thử tích hợp gồm: chiến lược tích hợp từ trên xuống (Top-Down) và tích hợp từ dưới lên từ dưới lên (Bottom- up). Ngoài ra còn sử dụng chiến lược bigbang và sandwich

a. Chiến lược từ trên xuống (top-down)

Giải thuật này gồm các bước:

- Sử dụng các module chính như là một driver, Stub dùng để thay thế cho tất cả các module là con trực tiếp của module chính
- Lần lượt thay thế các stub mỗi lần một cái bởi các module thực sự
- Tiến hành kiểm tra tính đúng đắn
- Một tập hợp bộ thử nghiệm được hoàn tất khi hết stub
- Kiểm tra lùi có thể được tiến hành để đảm bảo rằng không phát sinh lỗi mới



Ưu điểm:

- Kiểm thử từ trên xuống kết hợp với phát triển từ trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và giảm giá thành sửa đổi

- Nhanh chóng có phiên bản thực hiện với các chức năng chính

- Có thể Nghiệm thu tính dùng được của sản phẩm sớm

Nhược điểm: Nhiều module cấp thấp rất khó mô phỏng, Thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp…

b. Từ dưới lên (bottom - up)

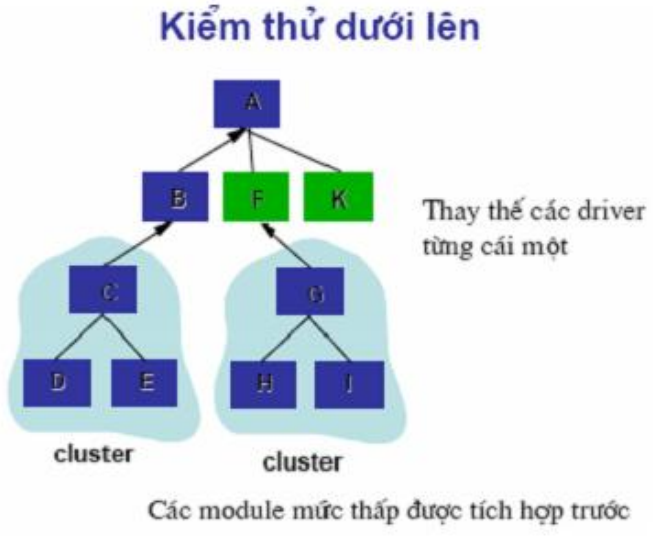
Ta kiểm tra module lá trước, do đó không cần phải viết Stub

Giải thuật như sau:

- Các module cấp thấp được nhóm thành từng nhóm(thực hiện cùng chức năng)

- Viết các driver điều khiển thao tác nhập xuất

- Bỏ driver và gắn chùm vào module cao hơn



Ưu điểm:

- Tránh xây dựng các module phức tạp

- Tránh sinh các kết quả nhân tạo
- Thuận tiện cho phát triển các module dùng lại

Nhược điểm:

- Chậm phát hiện các lỗi kiến trúc
- Chậm có phiên bản thực hiện

6.5.3 Kiểm thử chức năng (Functional testing)

Kiểm thử ở bất kỳ mức độ nào (lớp, module, giao diện hay hệ thống) để kiểm tra xem nó có đúng với các đặc tả hay không?

6.5.4 Kiểm thử hệ thống (System testing)

Kiểm thử hệ thống một cách toàn diện để đánh giá các đặc tả chức năng có được đáp ứng, các thao tác giao diện có giống thiết kế không?

Công việc kiểm thử nhằm kiểm tra công việc phục hồi sau lỗi, độ an toàn, hiệu năng, giới hạn của phần mềm.

6.5.5 Kiểm thử tích hợp hệ thống (System integration testing)

Kiểm tra hệ thống có tích hợp đúng với các phần mềm của hãng thứ ba hay các giao diện của hệ thống khác hay không?

6.5.6 Kiểm thử sự thực thi (Performance testing)

Để kiểm tra hiệu suất của chương trình có đáp ứng được như mong đợi hay không?

**** Kiểm thử được tiến hành bởi các cơ quan bên ngoài: Đây là nhóm thứ hai độc lập không thuộc nhóm thứ nhất, nhóm này có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chưa phát hiện***

Được gọi là đảm bảo chất lượng (Quality Assurance - QA) và kiểm thử chấp nhận (Acceptance testing). Người ngoài có thể là người sử dụng hay đại diện người dùng nằm ngoài sự điều khiển và quản lý của người quản lý dự án.

Kiểm thử chấp nhận được tiến hành từ phía khách hàng, còn được gọi là alpha testing. Mục đích của bước kiểm thử này nhằm thẩm định lại xem phần mềm còn có những sai sót, thiếu sót so với yêu cầu của người sử dụng không. Trong giai đoạn này dữ liệu dùng để kiểm thử do người dùng cung cấp.

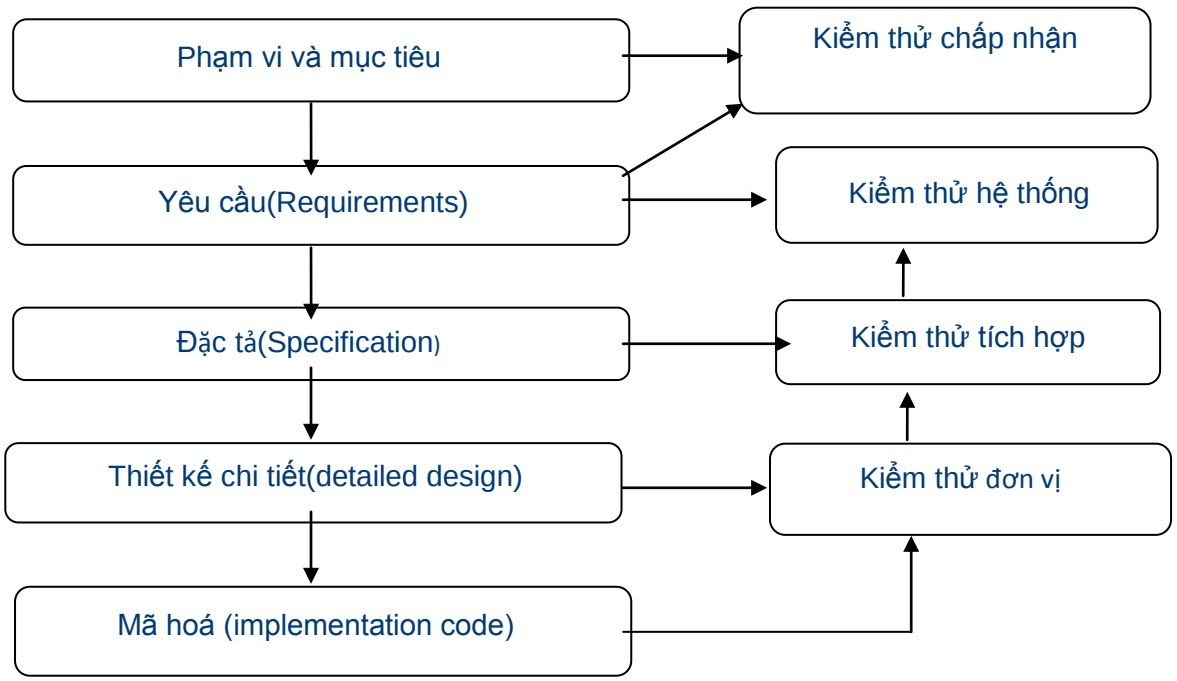
Kiểm thử Beta là giai đoạn mở rộng của kiểm thử alpha. Công việc kiểm thử được thực hiện với số lượng lớn người sử dụng. Công việc kiểm thử được tiến hành ngẫu nhiên, không có sử hướng dẫn của nhà phát triển. Lỗi tìm ra sẽ gửi lại cho nhà phát triển phần mềm.

QA testing tương tự system testing về mặt mục đích và cách tiến hành. Các báo cáo kiểm thử QA được gửi thường xuyên tới người quản lý dự án. Các QA lập kế

hoạch và kiểm tra để đảm bảo các ứng dụng thực hiện tất cả các chức năng cần thiết. Kiểm thử QA là bước cuối cùng trước khi ứng dụng được sản xuất đại trà.

Các chiến lược kiểm thử không loại trừ lẫn nhau, sử dụng riêng rẽ hay đồng thời. Với một ứng dụng phải sử dụng nhiều chiến lược để phát hiện được hết lỗi. Các chiến lược sau khi được xác định sẽ được áp dụng để tạo các trường hợp kiểm thử cụ thể

Các mức kiểm thử



Các giai đoạn phát triển ứng dụng

6.6 Kỹ thuật kiểm thử phần mềm

6.6.1 Kiểm thử hộp đen

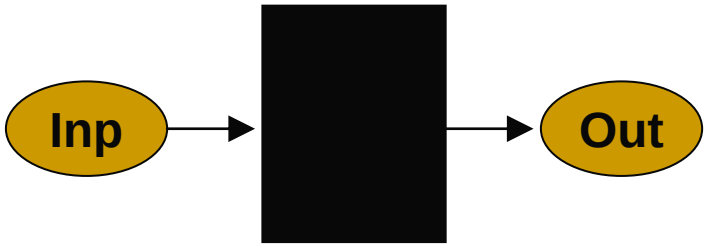
Trong các giai đoạn kiểm thử trên cần áp dụng các kỹ thuật kiểm thử hộp đen và kỹ thuật kiểm thử hộp trắng.

Kiểm thử hộp đen – Black Box: các kỹ sư kiểm thử chỉ biết rằng phần mềm đó hỗ trợ những chức năng gì thông qua bản đặc tả mà không cần biết bên trong chương trình được viết như thế nào. Ta chỉ cần nhập dữ liệu vào và sẽ nhận được kết quả mong muốn chứ không cần biết như thế nào và tại sao nó lại làm được như vậy. Với kỹ thuật

kiểm thử này chúng ta chỉ phát hiện được các lỗi chức năng, sai sót về giao diện của module, tính hiệu quả, phát hiện lỗi khởi tạo, lỗi kết thúc.

Đối với phương pháp kiểm thử hộp đen, để thiết kế các testcase dựa vào phương pháp phân hoạch miền giá trị. Lý do là không thể kiểm tra được mọi trường hợp trên thực tế. Chúng ta phân hoạch thành các miền giá trị, sau đó thiết kế bộ thử nghiệm tương ứng, đặc biệt là dữ liệu biên.

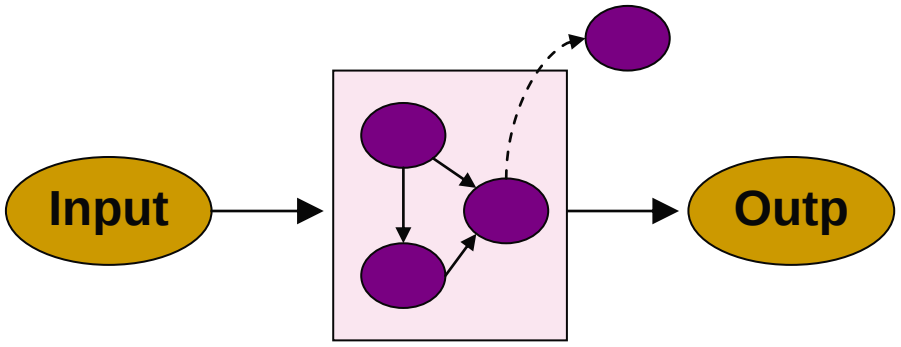
Kỹ thuật kiểm thử hộp đen chỉ dựa vào đặc tả chương trình, xây dựng dữ liệu thử trước khi mã hóa/lập trình, thường phát hiện các lỗi đặc tả yêu cầu, thiết kế; Dễ dàng thực hiện, chi phí thấp.



Kiểm thử hộp đen

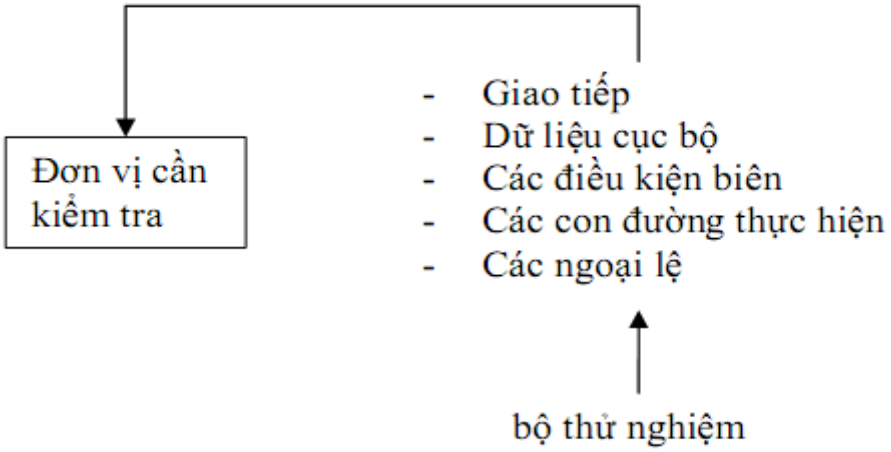
6.6.2 Kiểm thử hộp trắng

Ngược lại với kỹ thuật kiểm thử hộp đen là kỹ thuật kiểm thử hộp trắng-White-Box (clear-box) thì kỹ sư kiểm thử phải kiểm tra mã nguồn của chương trình để có thể kiểm thử hay tìm ra những đầu mối để giúp cho công việc kiểm thử. Dựa vào những gì ta thấy thì có thể xác định chính xác một số lỗi. Tuy nhiên, trong kỹ thuật này có một số rủi ro nhất định. Đó là vì can thiệp sâu vào mã nguồn nên sẽ rất có thể có ý kiến chủ quan của tester



Kỹ thuật kiểm tra hộp trắng

Theo phương pháp này ta chia testcase dựa vào cấu trúc đơn vị cần kiểm tra.



Sơ đồ các testcase sử dụng trong kỹ thuật kiểm thử hộp trắng

Trong đó:

- Kiểm tra giao tiếp của đơn vị là để đảm bảo dòng thông tin vào ra của đơn vị luôn đúng (đúng giá trị, khớp kiểm..).

- Kiểm tra dữ liệu cục bộ để đảm bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình giải thuật được thực hiện.

Ví dụ: như nhập dữ liệu sai, tên biến không đúng, kiểu dữ liệu không nhất quán, các rằng buộc hoặc ngoại lệ.

- Kiểm tra các điều kiện biên của các câu lệnh if, vòng lặp để đảm bảo đơn vị luôn chạy đúng tại các biên này.

- Kiểm tra để đảm bảo mọi con đường thực hiện phải đi qua ít nhất một lần. Con đường thực hiện của một đơn vị chương trình là một dãy có thứ tự các câu lệnh bên trong đơn vị đó sẽ được kích hoạt khi thực hiện

6.6.3 Kỹ thuật kiểm tra tĩnh và động (static và dynamic)

Kỹ thuật Static tức là kỹ thuật kiểm thử mà phần mềm đó không đang hoạt động. Ngược lại với kỹ thuật kiểm thử Static, kỹ thuật Dynamic là kỹ thuật kiểm thử mà vừa chạy phần mềm vừa kiểm thử. Ví dụ kiểm thử một chiếc xe. Kiểm thử bánh xe, màu sơn đó là kỹ thuật kiểm thử Static. Khi khởi động, nghe tiếng động cơ và lái đó là kỹ thuật kiểm thử Dynamic.

6.6.4 Kỹ thuật kiểm thử hộp xám

Nếu kết hợp hai kỹ thuật kiểm thử hộp đen, hộp trắng với hai kỹ thuật kiểm thử tĩnh và động ta có loại kiểm thử sau: Static black-box (dùng để test bản tài liệu đặc tả), dynamic black-box (dùng để test phần mềm), và static white-box (để kiểm thử code). Ngoài ra còn có thêm một kỹ thuật nữa đó là dynamic white-box (hay còn là kỹ thuật hộp xám).

Dynamic white-box là sử dụng những thông tin mà được thu lượm trong quá trình

xem xét đoạn code và cách làm việc của chúng để nghĩ ra những gì phải test, những gì không cần phải test, và làm cách nào để thực hiện việc test của mình. Một tên khác thường được sử dụng để nói đến kỹ thuật này đó là kiểm thử cấu trúc cấu trúc (Structural testing) bởi vì ta có thể nhìn thấy và sử dụng cấu trúc bên trong các đoạn code để thiết kế và chạy các test case.

Kiểm thử dynamic white-box không đơn giản là chỉ nhìn thấy những gì mà code thực hiện, mà nó còn có thể bao hàm cả việc test và điều khiển phần mềm một cách trực tiếp. Bốn lĩnh vực mà kỹ thuật dynamic white-box tập trung:

- Test bên dưới cấu trúc các hàm, thủ tục, thư viện một cách trực tiếp. Trong Microsoft Windows, các thư viện đó được gọi là Application Programming Interfaces (APIs).
- Test phần mềm ở mức độ người sử dụng như là test một phần mềm hoàn chỉnh, nhưng điều chỉnh những test case dựa vào những gì được biết về quá trình hoạt động của phần mềm.
- Đọc để lấy thông tin về các biến và trạng thái trong phần mềm để giúp ta xác định những nơi test. Và có thể ép phần mềm thực hiện những việc mà sẽ rất khó khăn nếu test nó một cách bình thường.

Dự đoán bao nhiêu code và chức năng mà được tác động vào khi chạy các test case và sau đó điều chỉnh để có thể giảm số lượng test case

6.7 Thiết kế Testcase

6.7.1 Thiết kế testcase cho kiểm thử hộp đen

6.7.1.1 Phân hoạch cân bằng

Mục đích của phân hoạch cân bằng là tối thiểu các trường hợp kiểm thử với mỗi mức kiểm tra cho trước, các mục dữ liệu vào được chia thành các nhóm dữ liệu có vai trò cân bằng nhau, mỗi nhóm đại diện cho một tập dữ liệu. Nguyên tắc là bằng cách kiểm thử triệt để một mục của mỗi tập hợp, chúng ta có thể chấp nhận tất cả các mục tương đương khác của tập hợp đó cũng sẽ được kiểm tra một cách kỹ càng do đó làm giảm tổng số các trường hợp kiểm thử phải xây dựng. Các bước:

Bước 1: Xác định (phân hoạch) các lớp tương đương: Thiết kế trường hợp kiểm thử cho phân hoạch cân bằng dựa trên một đánh giá về các lớp tương đương với một điều kiện vào. Lớp tương đương biểu thị cho một tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào. Các lớp tương đương có thể được định nghĩa theo những hướng dẫn sau:

- Nếu một điều kiện vào xác định một miền thì một lớp hợp lệ và hai lớp không hợp lệ được xác định.

- Nếu một điều kiện vào đòi hỏi một giá trị xác định thì một lớp hợp lệ và hai lớp không hợp lệ được xác định.
- Một điều kiện vào xác định một thành viên của một tập thì một lớp hợp lệ và một lớp không hợp lệ được xác định.
- Nếu một điều kiện vào là bool thì một lớp hợp lệ và một lớp không hợp lệ được xác định.

- Bước 2: Mỗi lớp tương đương xác định ít nhất một testcase.

Câu hỏi: Mỗi lớp tương đương xác định nhiều hơn 1 testcase được không?

Trả lời: Được nếu đủ thời gian và chi phí.

Ví dụ 1: Cho bài toán như sau:

User: là một ô text

Password: là một ô text

Yêu cầu: Thiết kế test case sao cho khi người dùng nhập user vào ô text thì chỉ cho nhập số ký tự [6 – 20].

Bài làm: Do yêu cầu của bài toán chỉ cho phép nhập số ký tự vào trong khi nhập của user

nằm [6 - 20] nên ta có tình huống kiểm thử sau:

- Nhập vào một trường hợp hợp lệ:nhập 7 ký tự.
- Nhập vào trường hợp không hợp lệ thứ nhất: nhập 5 ký tự.
- Nhập vào trường hợp không hợp lệ thứ hai: nhập vào 21 ký tự.
- Trường hợp đặc biệt: không nhập gì vào ô text đó (để trống).

Lập bảng các lớp tương đương:

1. Điều kiện đầu vào: Cho phép nhập số ký tự nằm [6 – 20]
2. Các lớp tương đương hợp lệ: Nhập vào 7 ký tự
3. Các lớp tương đương không hợp lệ
   - Nhập vào 5 ký tự
   - Nhập vào 21 ký tự
   - Để trống ô đó

Ví dụ 2: Yêu cầu

Chương trình đọc vào 3 giá trị nguyên từ hộp thoại vào. Ba giá trị này tương ứng với chiều dài 3 cạnh của 1 tam giác. Chương trình hiển thị 1 thông điệp cho biết tam giác đó là tam giác thường, cân, hay đều.

Ba giá trị nhập vào thỏa mãn là 3 cạnh của một tam giác khi và chỉ khi cả 3 số đều là số nguyên dương, và tổng của 2 số bất kỳ trong 3 số phải lớn hơn số thứ 3. Khi đó, một tam giác đều là tam giác có 3 cạnh bằng nhau, tam giác cân là tam giác có 2 trong 3 cạnh bằng nhau, và tam giác thường thì có 3 cạnh khác nhau.

Bài làm

1.Xác định các lớp tương đương

Các giá trị nhập vào là số	Cả 3 giá trị đều là số (1)	Tồn tại 1 giá trị không phải là số (2)
Các giá trị là nguyên	Cả 3 giá trị đều nguyên (3)	Tồn tại 1 giá trị không nguyên (4)
Các giá trị là dương	Cả 3 giá trị đều dương (5)	Tồn tại 1 giá trị <=0 (6)
Hằng số	-32768 : 32767 (7)	<-32768 (8), >32767 (9)
Tổng 2 số bất kỳ so với số thứ 3	Lớn hơn (10)	Nhỏ hơn hoặc bằng (11)

6.7.1.2 Phân tích cực biên

a) Giới thiệu về kiểm tra giá trị biên

Kiểm tra giá trị biên – đây là một phương pháp kiểm thử quan trọng, nó nằm ở gần như ở tất cả các mức kiểm thử như kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống và kiểm thử hồi quy.

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỉ lệ phần trăm cao hơn các ca kiêm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra. Phân tích giá trị biên là phương pháp thiết kế các ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở hai khía cạnh:

Kiểm tra giá trị biên không lựa chọn phần tử bất kỳ trong lớp tương đương là phần tử điển hình, mà nó yêu cầu là một hoặc nhiều phần tử được lựa chọn.

Ngoài việc chỉ tập trung vào các trạng thái đầu vào, các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả.

b) Một số quy tắc khi tiến hành kiểm thử giá trị biên

Phần tích giá trị biên yêu cầu óc sáng tạo và lượng chuyên môn hóa nhất định và nó là một quá trình mang tính kinh nghiệm rất cao. Tuy nhiên có một số quy tắc chung như sau:

1. Nếu một trang thái đầu vào định rõ giới hạn của các giá trị, hãy viết các ca kiểm thử cho các giá trị cuối của các giới hạn và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vừa ra ngoài phạm vi.
2. Nếu một trạng thái đầu vào định rõ số lượng của các giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên và một giá trị cuối của các giá trị này.

3. Sử dụng quy tắc một cho mỗi trạng thái đầu vào. Ví dụ: Với một chương trình tính toán khấu trừ lương hàng tháng với khấu trừ lương thấp nhất là 0.00$ và lương cao nhất là 500.25$, ta sẽ viết các ca kiểm thử mà khấu trừ 0.00$ và 500.25$, khấu trừ âm và khấu trừ lớn hơn 500.25$. Chú ý việc xem xét không gian kết quả là quan trọng vì không phải lúc nào các biến của miền đầu vào cũng mô tả cùng một tập sự kiện như biên của giới hạn đầu ra (ví dụ: xet chương trình con tính sin). Ngoài ra không phải lúc nào ta cũng có thể tạo ra một kết quả bên ngoài giới hạn đầu ra, nhưng tuy nhiên rất đáng để xem xét tiềm ẩn đó.
4. Sử dụng nguyên tắc hai cho các trạng thái đầu ra.
5. Nếu đầu vào hay đầu ra được xắp theo thứ tự, tập trung chú ý vào phần tử đầu tiên và phần tử cuối cùng của dãy.
6. Sử dụng sự khéo léo của bạn để tìm ra các điều kiện biên.

c) Các bước tiến hành kiểm tra giá trị biên

Kiểm tra giá trị biên thường được tiến hành sau khi sản phẩm đã hoàn thành, việc kiểm thử được tiến hành dựa vào việc kiểm thử chức năng của phần mềm xem nó có phù hợp với đặc tả ban đầu hay không. Sau đây là ba bước tiến hành kiểm tra giá trị biên:

1. Xác định các lớp tương đương.
2. Xác định các biên cho mỗi lớp tương đương.
3. Tạo test – case cho mỗi giá trị biên.

* Xác định các lớp tương đương

Các lớp tương đương được xác định bằng cách lấy mỗi trạng thái đầu vào (thường là một câu hay một cụm từ trong đặc tả) và phân chia nó thành hai hay nhiều nhóm. Từ các nhóm này ta phân chia thành các lớp hợp lệ và không hợp lệ.

- Lớp tương đương hợp lệ: Mô tả các đầu vào hợp lệ của chương trình.
- Lớp tương đương không hợp lệ: Mô tả tất cả các trạng thái có thể khác của điều kiện.

Với một đầu vào hay điều kiện bên ngoài đã cho việc xác định lớp tương đương hầu như là một quy trình mang tính kinh nghiệm. Để xác định các lớp tương đương một cách hiệu quả ta có thể áp dụng các nguyên tắc sau đây:

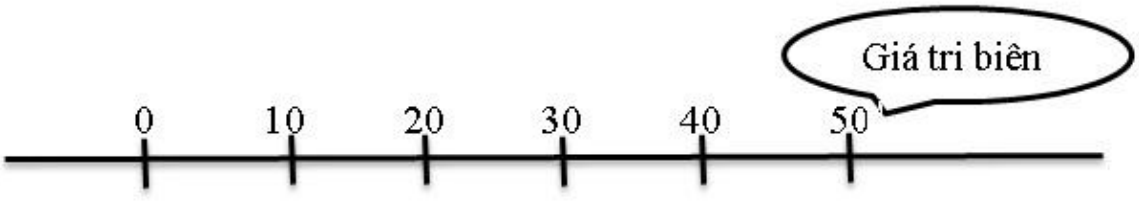
1. Nếu trạng thái đầu vào định rõ giới hạn của các giá trị hoặc xác định số giá trị, xác định một lớp tương đương hợp lệ và hai lớp tương đương không hợp lệ.
2. Nếu trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau, xác định lớp tương đương hợp lệ cho mỗi loại và một lớp tương đương không hợp lệ.
3. Nếu trạng thái đầu vào chỉ định một tình huống chắc chắn, xác định một lớp tương đương hợp lệ và một lớp tương đương không hợp lệ.

Nếu có bất kỳ một lý do để tin rằng chương trình không xử lý các phần tử trong cùng một lớp là như nhau, thì ta chia các lớp tương đương đó thành các lớp nhỏ hơn.

* Xác định biên cho mỗi lớp tương đương

Sau khi xác định được các lớp tương đương, ta tiến hành xác định biên cho các lớp tương đương. Đầu tiên ta cần xác định rõ biên là gì? Biên được hiểu đơn giản nó là các giá trị cận của tập các giá trị.

Ví dụ: Ta có một tập các lớp tương đương như sau



Nhìn vào ví dụ ta dễ dàng nhận thấy giá trị biên của một phân lớp tương đương.

* Tạo test case cho mỗi giá trị

Việc tạo test case cho mỗi giá trị biên bằng cách lựa chọn các điểm nằm ở biên, dưới biên và trên biên, lưu ý "dưới" và "trên" phụ thuộc vào những đơn vị của giá trị tài liệu. Ví dụ ta có biên là kiểu số nguyên với giá trị là 16 thì ta sẽ có điểm dưới sẽ là 15 và điểm trên sẽ là 17. Vẫn là giá trị 16 nhưng ở đây ta chọn không phải là kiểu số nguyên mà là kiểu số thực và ta chỉ lấy thêm hai số sau dấu phẩy thì ta sẽ thu được giá trị của điểm dưới sẽ là 15.99 và điểm trên sẽ là 16.01.

Điểm trên có thể nằm ở trên một lớp tương đương khác với lớp tương đương mà ta đang sét,và ta không có lý do gì để kiểm tra lại, tương tự đối với điểm dưới.

* Khi nào kết thúc quá trình kiểm tra giá trị biên

Có 3 yếu tố hết sức quan trọng trong quá trình kiểm thử đó là: Con người, thời gian và tiền bạc. Rất dễ để có thể thực hiện mọi kịch bản trong quá trình kiểm thử chấp nhận, tuy nhiên điều quan trọng là phải biết tập trung vào các chức năng, yêu cầu chính của phần mềm, tránh những công việc lan man, không cần thiết trong quá trình kiểm thử.

Việc kiểm tra giá trị biên theo đúng nguyên lý được thực hiện bởi các kiểm thử viên quá trình kiểm tra giá trị biên sẽ được ngừng khi mà các test case đã được test hết hoặc bên phát triển phần mềm có thể yêu cầu ngừng khi các chức năng chính cơ bản đã hoàn thành, tuy nhiên cũng cần tôn trọng ý kiển của kiểm thử viên vì họ mới là những người trực tiếp giám sát các ca kiểm thử.

* Tóm tắt

Kiểm tra giá trị biên là chọn lựa các ca kiểm thử thực hiện các giá trị cận. Phương pháp này cho rằng số lớn các sai xuất hiện ở biên nhiều hơn là vùng dữ liệu trung tâm. Không những chú ý đến dữ liệu trong và sát biên mà còn chú ý đến dữ liệu ngoài và sát biên.

Phương châm:

1. Nếu điều kiện và là một miền giới hạn bởi a và b thì cần thiết kế các ca kiểm thử cho cả a và b trên a và dưới b.
2. Nếu điều kiện vào đặc tả một số các giá trị thì lên thiết kế các ca kiểm thử cho cả các số trên và dưới số nhỏ nhất và lớn nhất.
3. Áp dụng các phương châm một và hai cho các điều kiện ra.
4. Nếu chương trình trung gian có các biên đã được mô tả thì chắc chắn phải thiết kế ca kiểm thử để thi hành cấu trúc dữ liệu ở biên của nó.

Các bước xác định testcase:

- Định nghĩa các lớp tương đương.
- Xác định các biên tại mỗi lớp
- Tại giá trị biên xác định 3 testcase:

+ Giá trị tại biên

+ Giá trị cận trên biên

+ Giá trị cận dưới biên

6.7.1.3 Đoán lỗi

Trên cơ sở trực giác và kinh nghiệm, các chuyên gia có thể dễ dàng kiểm tra các điều kiện lỗi bằng cách đoán cái nào dễ xảy ra nhất. Ví dụ, chia cho 0 nếu module có phép chia. Vì dựa trên trực giác nên phép thử này khó tìm hết các lỗi.

6.7.2 Tạo testcase cho phương pháp kiểm thử hộp trắng

Kiểm thử hộp trắng còn gọi là kiểm thử dựa vào đường dẫn, cấu trúc, và sự thực thi phần mềm . Kiểm thử theo cách này là loại kiểm thử sử dụng các thông tin về cấu trúc bên trong của ứng dụng. Khác với kiểm thử hộp đen, kiểm thử hộp trắng đòi hỏi người kiểm thử viên phải có kinh nghiệm lập trình.

Quá trình kiểm thử hộp trắng được thực hiện theo quy trình chung như sau:

- Phân tích phần mềm cần kiểm thử
- Định nghĩa các đường thực thi của phần mềm cần kiểm thử
- Lựa chọn giá trị đầu vào (test case) để chạy các đường được lựa chọn
- Chạy các bộ kiểm thử
- So sánh kết quả thực khi chạy các bộ test với đầu ra mong muốn. Nếu hai giá trị giống nhau thì không có lỗi. Nếu khác nhau thì chương trình cần kiểm thử có lỗi.
- Đưa ra quyết định phù hợp

Kiểm thử hộp trắng được áp dụng với các mức kiểm thử phần mềm như: mức đơn vị, mức tích hợp và mức hệ thống. Tuy nhiên kiểm thử hộp đen được áp dụng nhiều ở mức đơn vị do nhà phát triển thực hiện. Nhưng chúng ta có thể áp dụng kĩ thuật kiểm thử này để kiểm thử các đường dẫn giữa các module trong hệ thống con hoặc giữa các hệ thống con trong một hệ thống lớn hoặc trong toàn bộ hệ thống.

Hạn chế: có 4 hạn chế sau

- Số lượng các đường thực thi quá lớn
- Khó tìm ra các lỗi nhạy cảm như lỗi chia cho 0, hoặc lỗi gõ sai công thức. Ví dụ: y=2*x thì viết nhầm thành y=x2. Khi nhập x=0 thì y=0, x=2 thì y=4.
- Kiểm thử hộp trắng thừa nhận kiểm thử luồng điều khiển là đúng (hoặc gần đúng). Vì kiểm thử được thực hiện trên các đường tồn tại, còn những đường không tồn tại thì không phát hiện được qua kiểm thử hộp trắng
- Người kiểm thử phải là người có kinh nghiệm lập trình để hiểu và đánh giá được phân mềm cần kiểm thử. Trong khi đó hiện nay các công ty kiểm thử viên là người không được đào tạo một cách có bài bản.

Ưu điểm: Kiểm thử hộp trắng đảm bảo rằng tất cả các đường đi trong phần mềm đều được định nghĩa và kiểm thử.

Kiểm thử hộp trắng có hai kỹ thuật: Kiểm thử luồng điều khiển (control flow testing) và kiểm thử luồng dữ liệu (data flow testing)

6.7.2.1 Kiểm thử luồng điều khiển (control flow testing)

Kỹ thuật này tiếp cận định nghĩa về đường thực thi qua module mã chương trình. Sau đó tạo các testcase để bao phủ các đường thực thi này. Trong đó đường thực thi là trình tự các câu lệnh bắt đầu từ đầu vào cho đến đầu ra.

Một số trở ngại trong kiểm thử luồng điều khiển

- Số lượng các đường thực thi lớn, không có khả năng kiểm thử trong thời gian hợp lý. Ví dụ đoạn chương trình sau:

```
for (i=1; i<=1000; i++)
    for (j=1; j<=1000; j++)
     for (k=1; k<=1000; k++)
      doSomethingWith(i,j,k);
```

Như vậy cần 10^9 thực thi lệnh doSomethingWith(). Mỗi đường đều cần phải kiểm thử. Không thể kiểm thử được những đường bị thiếu. Ví dụ như đoạn chương trình sau:

```
if (a>0) doIsGreater();
   if (a==0) dolsEqual();
   // thiếu nhánh a<0
```

- Một module có thể đúng với hầu hết các giá trị nhưng lại sai với một số giá trị. Ví dụ như:

```
int blech (int a, int b) {
     return a/b;
   }
```

Hàm này sai nếu b=0 còn đúng với b <>0

Mặc dù kỹ thuật kiểm thử luồng điều khiển có một số mặt hạn chế, xong nó vẫn là một công cụ kiểm thử quan trọng mà người kiểm thử dùng

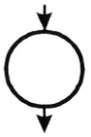
*** Các bước thực hiện**

Bước 1: Vẽ sơ đồ luồng điều khiển

Sơ đồ luồng điều khiển là nền tảng quan trọng trong kiểm thử luồng điều khiển. Sơ đồ này minh họa cho cấu trúc điều khiển của module. Các mã lệnh trong module sẽ được chuyển sang sơ đồ luồng. Đường dẫn trong sơ đồ luồng sẽ được phân tích. Các testcase sẽ được tạo ra qua bước phân tích trên. Sơ đồ luồng điều khiển bao gồm các thành phần sau:

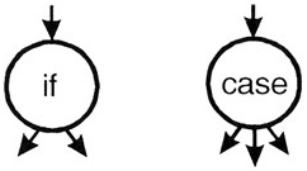
a) Khối xử lý

Khối xử lý là tập hợp các câu lệnh tuần tự được thực thi từ câu lệnh tuần tự đầu tiên đến câu lệnh tuần tự cuối cùng. Khối đầu tiên coi như là đầu vào, khối kết thúc coi như là đầu ra. Một khối được khởi tạo, các câu lệnh trong khối được thực thi tuần tự. Khối xử lý trong sơ đồ luồng được vẽ như sau:



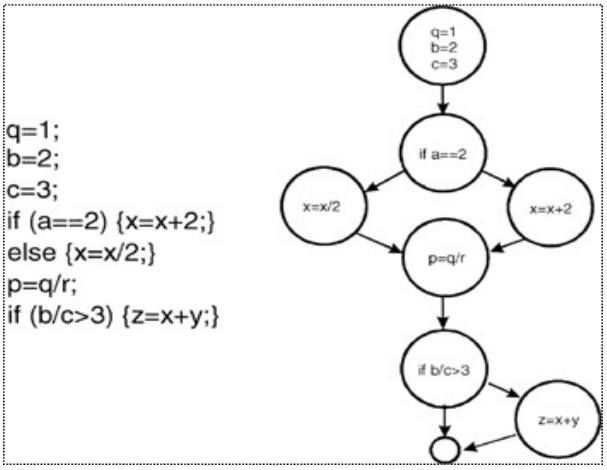
b) Điểm quyết định (decision point)

Điểm quyết định là một điểm trong module chương trình mà luồng điều khiển thay đổi. Hầu hết các điểm quyết định được tạo ra bởi câu lệnh if và câu lệnh case. Nó được biểu diễn bởi 1 đầu vào và nhiều đầu ra như sau:



c) Điểm nối: Một điểm nối là điểm mà tại đó các luồng điều khiển tham gia cùng nhau. Điểm nối được vẽ như sau:

Sau đây là ví dụ về sơ đồ luồng của đoạn mã chương trình tương ứng:



Bước 2: Tìm các mức bao phủ

Trong kiểm thử luồng điều khiển các mức độ bao phủ khác nhau đều được định nghĩa. "Bao phủ" có nghĩa là tỉ lệ mã nguồn được kiểm thử.

a) Phủ các đỉnh (câu lệnh)

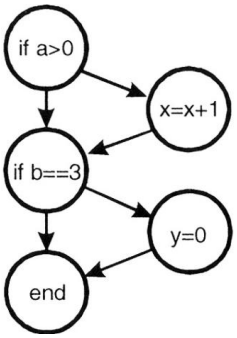
Đây là mức bao phủ thấp nhất (bao phủ câu lệnh) tức là 100% các câu lệnh trong chương trình được thực hiện ít nhất một lần

Ví dụ:

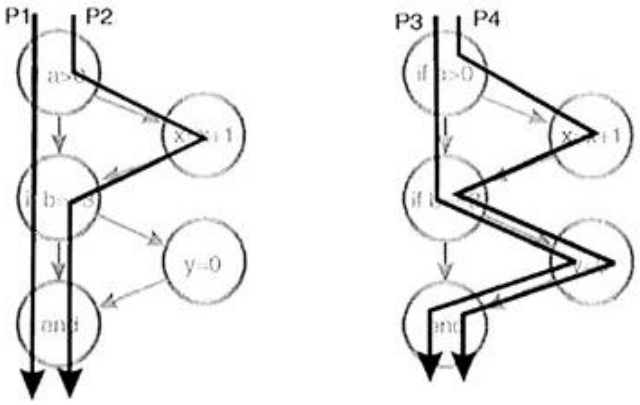
Cho đoạn mã chương trình:

```
if (a>0) {x=x+1;}
    if (b==3) {y=0;}
```

Sơ đồ luồng ứng với đoạn mã chương trình trên:



Ta sẽ tìm được 4 đường thực thi khác nhau với hai câu lệnh trên:



Qua ví dụ này ta thấy chỉ cần 1 testcase cơ bản (ví dụ cho a=6, b=3) là có thể kiểm thử mọi dòng lệnh của đoạn chương trình trên. Như vậy với mức kiểm thử câu lệnh sẽ kiểm thử thiếu rất nhiều đường thực thi khác. Chính vì vậy mức kiểm thử này là mức kiểm thử kém nhất.

Ví dụ 2:

```
function sum(x,y:integer):Integer;
begin
    if (x=0) then sum:=y;
    else sum:=x+y;
end
```

=> td1={x=6; y=10}; Td2 ={x=0;y=10}

b) Phủ mọi cung

Nguyên tắc: Phủ mọi đỉnh đồng thời mỗi cung phải được phủ ít nhất 1 lần

Vd3: Xét ĐTLĐK ở VD2

Td ={x=1;}; td2={x=0;} => phát hiện lỗi chia cho 0

VD4: Xét ĐTLĐK

c) Phủ mọi quyết định

Thiết kế dữ liệu thử sao cho thỏa mãn tiêu chuẩn phủ mọi cung; đồng thời mỗi quyết định phải được phủ ít nhất một lần (với mọi biểu thức logic con trong biểu thức logic điều kiện phải phủ với mọi giá trị đúng sai).

A	B	A and B
True	False	False
False	False	False

True	True	True
False	True	False

d) Phủ mọi lộ trình

Nên phủ 2 loại lộ trình

- lộ trình đi qua vòng lặp 1 số lần
- lộ trình đi qua vòng lặp mà không thực hiện vòng lặp nào

Vd7: Xét ĐTLĐK ở Vd6 => bộ dữ liệu thử thỏa mãn

6.7.2.2 Kỹ thuật kiểm thử luồng dữ liệu

Kiểm thử luồng điều là một công cụ kiểm thử mạnh dùng để phát hiện lỗi dùng giá trị của biến phù hợp

a) Kỹ thuật

Biến chứa giá trị dữ liệu có một vòng đời được xác định. Chúng được tạo ra, được sử dụng, và sau đó bị hủy bỏ. Trong một số ngôn ngữ lập trình (ví dụ như FORTRAN và BASIC) việc tạo ra và tiêu hủy biến là tự động. Một biến tạo ra lần đầu tiên được gán một giá trị và bị phá hủy khi thoát khỏi chương trình.

Trong các ngôn ngữ khác (như C, C++, và Java) việc tạo ra chỉ là hình thức. Các biến được khai báo bởi các câu lệnh như :

```
int  x;              / / x được tạo ra là một số nguyên
string  y;           / / y được tạo ra là một chuỗi
```

Những khai báo này thường xuất hiện bên trong một khối mã bắt đầu với một cặp ngoặc mở { và kết thúc với một ngoặc đóng }. Các biến được định nghĩa trong một khối được tạo ra khi định nghĩa của chúng được thực thi và sẽ tự động được hủy vào cuối một khối. Đây được gọi là "phạm vi" của biến. Ví dụ:

```
{    / / Bắt đầu khối ngoài
  int x;      / / x được định nghĩa là một số nguyên bên trong khối ngoài
  ...;         / / x có thể được truy cập tại đây
     {                / / Bắt đầu khối bên trong
  int y;      / / y được định nghĩa ở khối bên trong
  ...;                 / / Cả x và y có thể được truy cập tại đây
  }           / / y được tự động hủy vào cuối khối này
       ...;   / / x vẫn có thể được truy cập, nhưng y đã biến mất
```

} / / x tự động bị phá hủy

Các biến có thể được sử dụng trong tính toán (a = b +1). Chúng cũng có thể được sử dụng trong điều kiện (if (a> 42)). Trong cả hai cách sử dụng này, các biến đã được gán một giá trị trước khi nó được sử dụng.

Ba khả năng tồn tại cho sự xuất hiện đầu tiên của một biến thông qua một đường dẫn chương trình:

~d biến không tồn tại (chỉ ra bởi ~), sau đó nó được xác định (d)

~u biến không tồn tại, sau đó nó được sử dụng (u)

~k biến không tồn tại, sau đó nó bị phá hủy (k)

Ý đầu tiên là chính xác. Các biến này không tồn tại và sau đó nó được định nghĩa. Ý thứ hai là không đúng. Một biến không thể sử dụng trước khi nó được định nghĩa. Ý thứ ba là có thể không chính xác.

Phá hủy một biến trước khi nó được tạo ra là dấu hiệu của một lỗi lập trình. Bây giờ xem xét trình tự thời gian các cặp defined-được định nghĩa (d), uses-được sử dụng (u), killed-và bị phá hủy (k):

dd Định nghĩa và định nghĩa lại một lần nữa-không hợp lệ nhưng đáng nghi ngờ. Có thể là một lỗi lập trình.

du Định nghĩa và được sử dụng, hoàn toàn chính xác. Trường hợp bình thường.

dk Định nghĩa và sau đó hủy bỏ - không hợp lệ nhưng có thể là một lỗi lập trình.

ud Được sử dụng và được định nghĩa – chấp nhận.

uu Được sử dụng và được sử dụng lại – chấp nhận.

uk Được sử dụng và bị phá hủy – chấp nhận

kd Bị phá hủy và được định nghĩa – chấp nhận. Một biến bị hủy và định nghĩa lại.

ku Bị phá hủy và được sử dụng – một khiếm khuyết nghiêm trọng. Sử dụng một biến không tồn tại hoặc chưa được định nghĩa luôn là một lỗi.

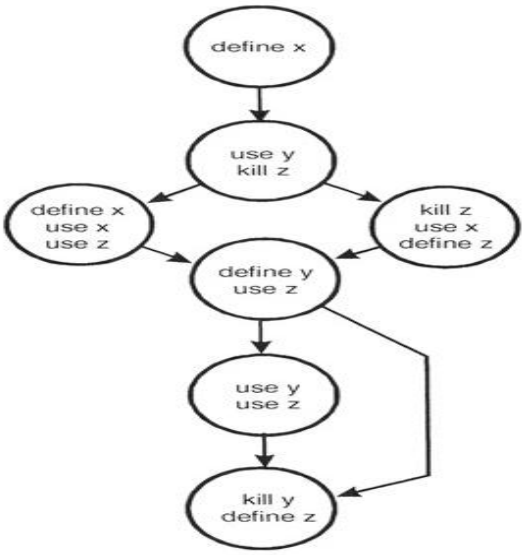
kk Bị phá hủy và bị phá hủy – có thể là một lỗi lập trình.

Một đồ thị luồng dữ liệu tương tự như một đồ thị dòng điều khiển ở chỗ nó hiển thị các luồng xử lý thông qua các module. Ngoài ra, nó chi tiết hóa các định nghĩa, sử dụng và tiêu huỷ mỗi biến của mô-đun. Chúng ta sẽ xây dựng các sơ đồ này và xác

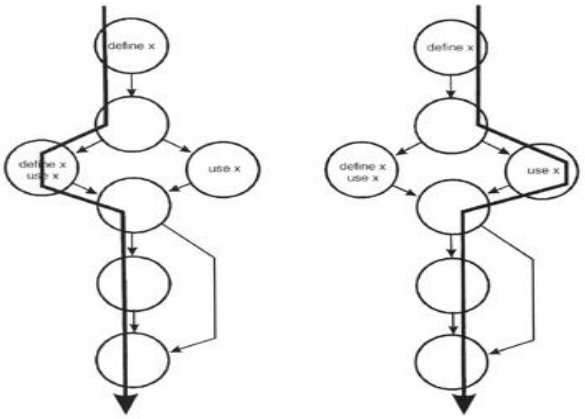
minh rằng các mẫu định nghĩa-sử dụng-phá hủy là thích hợp. Trước tiên, chúng tai sẽ thực hiện một thử nghiệm tĩnh của biểu đồ. Bằng cách "tĩnh" có nghĩa là chúng ta xem xét biểu đồ (thông qua kiểm tra hoặc thông qua nhìn-thấy). Thứ hai, chúng tôi kiểm tra năng động về module. Bằng cách "động", có nghĩa là chúng ta xây dựng và thực hiện các trường hợp thử nghiệm (test case). Hãy bắt đầu với các thử nghiệm tĩnh.

Static Data Flow Testing: Kiểm thử luồng dữ liệu tĩnh

Sơ đồ đã được chú thích với các thông tin định nghĩa-sử dụng-hủy bỏ (d-u-k) cho mỗi biến được sử dụng trong module.



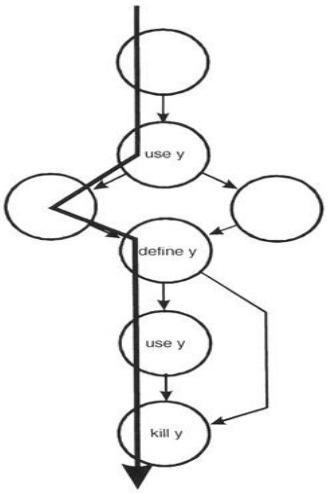
Đối với mỗi biến trong module, chúng ta sẽ kiểm tra mẫu định nghĩa-sử dụng-hủy bỏ (d-u-k) dọc theo các đường luồng điều khiển. Hãy xem xét biến x như chúng ta đi qua đường bên trái và sau đó đường bên phải:



Các mẫu d-u-k cho biến x (lấy trong cặp khi chúng ta đi dọc theo đường dẫn) là :

~định nghĩa (~d)	chính xác, trường hợp bình thường
Định nghĩa-định nghĩa (d-d)	nghi ngờ, có thể là lỗi lập trình
Định nghĩa-sử dụng (d-u)	chính xác, trường hợp bình thường

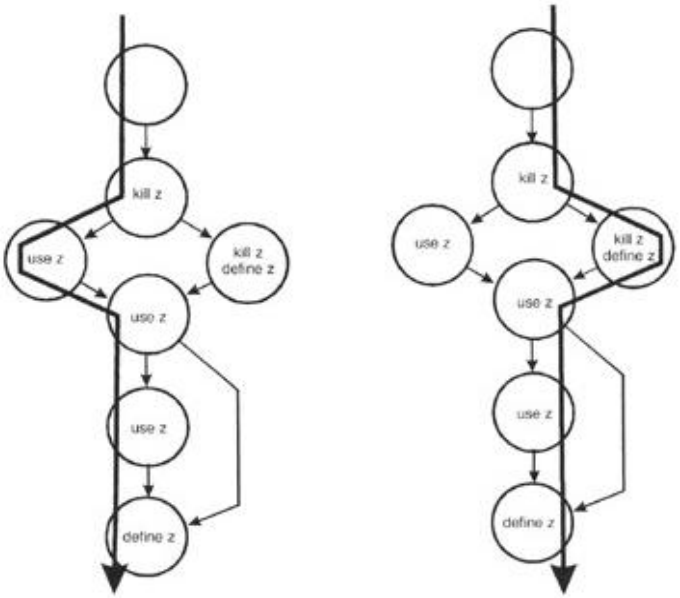
Bây giờ là cho biến y. Chú ý nhánh đầu tiên trong module không ảnh hưởng đến biến y



Mẫu d-u-k cho biến y (lấy trong cặp khi chúng ta đi dọc theo đường dẫn) là:

~u	sai lầm lớn
u-d	chấp nhận
d-u	chính xác, trường hợp bình thường
u-k	chấp nhận
d-k	có thể là lỗi lập trình

Bây giờ là cho biến z



Mẫu d-u-k là :

~k	lỗi lập trình

k-u	sai lầm lớn
u-u	chính xác, trường hợp bình thường
u-d	chấp nhận
k-k	có thể là lỗi lập trình
k-d	chấp nhận
d-u	chính xác, trường hợp bình thường

Trong khi thực hiện phân tích tĩnh trên mô hình luồng dữ liệu, có một số vấn đề :

x: define-define (d-d)

y: ~use (~u)

y: define-kill (d-k)

z: ~kill (~k)

z: kill-use (k-u)

z: kill-kill (k-k)

Thật không may, trong khi thử nghiệm tĩnh có thể phát hiện nhiều lỗi luồng dữ liệu, nó không thể tìm thấy tất cả. Xem xét các tình huống sau đây:

- Mảng là tập hợp của các phần tử dữ liệu mà có cùng tên và kiểu.

Ví dụ

```
Int     stuff[100]
```

định nghĩa một mảng có tên là stuff bao gồm 100 phần tử số nguyên. Trong C, C++, và Java, các phần tử riêng lẻ được đặt tên stuff[0], stuff [1], stuff [2], vv. Mảng được định nghĩa và hủy bỏ như một đơn vị nhưng các phần tử cụ thể của mảng được sử dụng riêng. Thông thường các lập trình viên dựa vào các stuff[j] nơi mà j thay đổi động như chương trình thực thi. Trong trường hợp tổng quát, phân tích tĩnh không thể xác định xem các quy tắc d-u-k trừ khi mỗi phần tử được xem là riêng lẻ

- Trong những luồng điều khiển phức tạp có thể là một con đường nhất định không bao giờ được thực thi. Trong trường hợp này một kết hợp d-u-k không đúng có thể tồn tại nhưng sẽ không bao giờ được thực thi và như vậy là không thực sự không đúng.

- Trong các hệ thống ngắt tiến trình, một số các hành động d-u-k có thể xảy ra ở mức độ gián đoạn trong khi các hành động d-u-k khác xảy ra ở cấp độ xử lý chính.

Ngoài ra, nếu hệ thống sử dụng nhiều cấp độ thực hiện ưu tiên, phân tích tĩnh của các tương tác có thể quá khó khăn để thực hiện bằng tay.

Vì lý do này, chúng ta bây giờ chuyển sang kiểm thử luồng dữ liệu động

Kiểm thử luồng dữ liệu động:

Bởi vì kiểm thử luồng dữ liệu là dựa trên một luồng điều khiển của một module, nó giả định rằng các luồng điều khiển cơ bản là chính xác. Các tiến trình kiểm thử luồng dữ liệu là lựa chọn đủ test cases để:

- Mỗi định nghĩa (define) bắt nguồn từ mỗi sử dụng (uses)
- Mỗi sử dụng (use) bắt nguồn từ mỗi định nghĩa (define) tương ứng

Để làm điều này, liệt kê các đường dẫn thông qua module. Điều này được thực hiện bằng cách sử dụng phương pháp tương tự như trong kiểm thử luồng điều khiển: Bắt đầu ở điểm vào của module, đi theo con đường bên trái qua các module để thoát nó. Quay trở lại đầu và thay đổi các điều kiện phân nhánh đầu tiên. Thực hiện theo đường để thoát khỏi. Quay trở lại đầu và thay đổi các điều kiện phân nhánh thứ hai, rồi thứ ba, và như vậy cho đến khi tất cả các đường dẫn được liệt kê. Sau đó, với mỗi biến, tạo ra ít nhất một trường hợp thử nghiệm (test case) để bao phủ mọi cặp define-use (d-u).

b) Ứng dụng và hạn chế

Kiểm thử luồng dữ liệu xây dựng dựa trên và mở rộng của kỹ thuật kiểm thử luồng điều khiển. Cũng như kỹ thuật kiểm thử luồng điều khiển, nó nên được sử dụng cho tất cả các module của mã mà không thể được kiểm thử đầy đủ thông qua đánh giá và kiểm tra. Hạn chế của nó là các tester phải có kỹ năng lập trình đủ để hiểu được mã, luồng điều khiển của nó, và các biến của nó. Cũng giống như kiểm thử luồng điều khiển, kiểm thử luồng dữ liệu có thể mất rất nhiều thời gian vì tất cả các mô-đun, đường dẫn, và biến mà bao gồm một hệ thống.

CHƯƠNG 7. BẢO TRÌ PHẦN MỀM

MỤC ĐÍCH

- Hiểu được vai trò của việc bảo trì phần mềm

- Nắm được các vấn đề liên quan đến bảo trì: phân loại, phương pháp, chi phí bảo trì …

- Hiểu được một số quy trình và các chiến lược cải tiến phần mềm

7.1 Bảo trì phần mềm

Bảo trì phần mềm chính là hoạt động chỉnh sửa chương trình sau khi nó đã được đưa vào sử dụng. Bảo trì thường không bao gồm những thay đổi chính liên quan tới kiến trúc của hệ thống. Những thay đổi trong hệ thống thường được cài đặt bằng cách điều chỉnh những thành phần đang tồn tại và bổ sung những thành phần mới cho hệ thống.

Bảo trì là không thể tránh khỏi vì:

- Các yêu cầu hệ thống thường thay đổi khi hệ thống đang được xây dựng vì môi trường thay đổi. Vì vậy, hệ thống được chuyển giao có thể không thoả mãn các yêu cầu của nó.

- Các hệ thống có gắn kết chặt chẽ với môi trường của nó. Khi hệ thống được cài đặt trong một môi trường nhất định nó sẽ làm thay đổi môi trường đó và vì vậy sẽ thay đổi các yêu cầu của hệ thống.

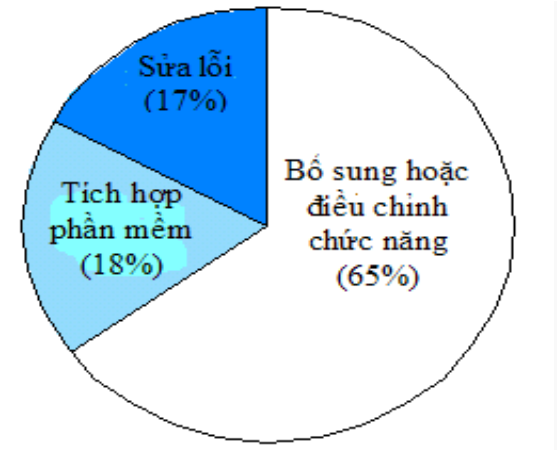
- Các hệ thống phải được bảo trì nếu chúng muốn là những phần hữu ích trong môi trường nghiệp vụ.

Phân loại các kiểu bảo trì:

- Bảo trì sửa lỗi: thay đổi hệ thống để sửa lại những khiếm khuyết nhằm thoả mãn yêu cầu hệ thống.

- Bảo trì tích hợp hệ thống vào một môi trường vận hành khác

- Bảo trì để bổ sung hoặc chỉnh sửa các yêu cầu chức năng của hệ thống: chỉnh sửa hệ thống sao cho thoả mãn các yêu cầu mới.



Hình 9.2: Phân bổ chi phí bảo trì

Các nhân tố ảnh hưởng đến chi phí bảo trì:

- Sự ổn định của đội dự án: chi phí bảo trì sẽ giảm nếu nhân viên trong đội dự án không thay đổi.

- Những trách nhiệm đã cam kết: người xây dựng hệ thống có thể không cam kết trách nhiệm bảo trì cho nên không có gì để bắt buộc họ phải thiết kế lại cho các thay đổi trong tương lai.

- Kỹ năng của nhân viên: nhân viên bảo trì thường không có kinh nghiệm và hiểu biết về miền ứng dụng của họ bị hạn chế.

- Tuổi thọ và cấu trúc chương trình: khi tuổi thọ và cấu trúc chương trình bị xuống cấp thì chúng càng trở lên khó hiểu và thay đổi nhiều.

7.1.1 Dự đoán bảo trì

Dự đoán bảo trì có liên quan tới việc đánh giá những phần nào của hệ thống có thể gây ra lỗi và cần nhiều chi phí để bảo trì. Khả năng chịu được sự thay đổi phụ thuộc vào khả năng bảo trì của các thành phần bị ảnh hưởng bởi sự thay đổi đó. Thực hiện các thay đổi có thể làm hỏng hệ thống và giảm khả năng bảo trì của nó. Chi phí bảo trì phụ thuộc vào số lượng các thay đổi và chi phí thay đổi phụ thuộc vào khả năng bảo trì.

7.1.2 Dự đoán thay đổi

Dự đoán số lượng các thay đổi có thể xảy ra và tìm hiểu mối quan hệ giữa hệ thống và môi trường của nó. Sự thay đổi yêu cầu hệ thống có liên quan chặt chẽ tới sự thay đổi của môi trường. Trong đó, các nhân tố ảnh hưởng tới mối quan hệ này bao gồm:

- Số lượng và độ phức tạp của các giao diện hệ thống

- Số lượng các yêu cầu bất ổn định có tính phân cấp

- Các quy trình nghiệp vụ của hệ thống.

Ta có thể dự đoán bảo trì thông qua việc đánh giá độ phức tạp của các thành phần hệ thống. Độ phức tạp phụ thuộc vào:

- Độ phức tạp của cấu trúc điều khiển

- Độ phức tạp của cấu trúc dữ liệu

- Kích thước của đối tượng, phương thức và mô-đun.

Ngoài ra, ta có thể sử dụng các phép đo quy trình để đánh giá khả năng bảo trì.

- Số lượng các yêu cầu cần bảo trì sửa lỗi.

- Thời gian trung bình cần thiết để phân tích ảnh hưởng

- Thời gian trung bình để cài đặt một yêu cầu thay đổi.

- Số lượng các yêu cầu cần giải quyết.

7.2 Các quy trình cải tiến phần mềm

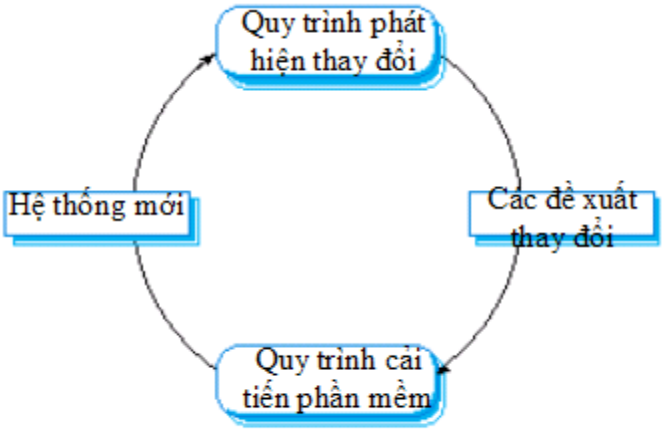
Các quy trình cải tiến phần mềm phụ thuộc vào:

- Kiểu phần mềm cần bảo trì

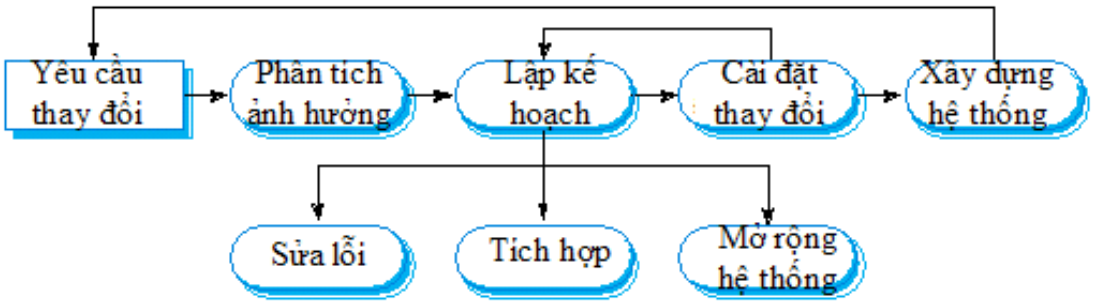
- Quy trình phát triển phần mềm đã được sử dụng

- Kỹ năng và kinh nghiệm của các stakeholder.

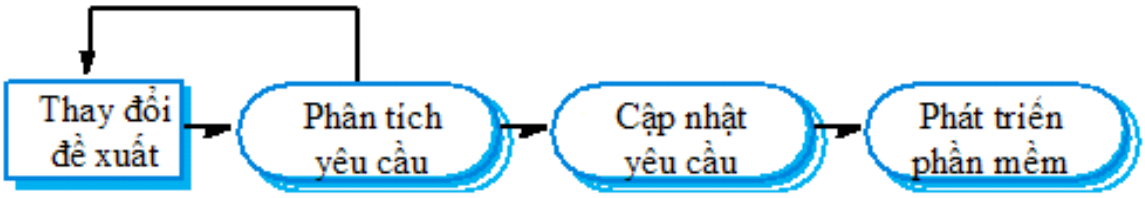
Các đề xuất thay đổi là định hướng để cải tiến hệ thống. Phát hiện thay đổi và cải tiến được thực hiện trong vòng đời hệ thống. Các hình vẽ sau đây thể hiện một cách khái quát các quy trình cải tiến hệ thống.



Hình 9.3: Phát hiện thay đổi và cải tiến



Hình 9.4: Quy trình cải tiến hệ thống



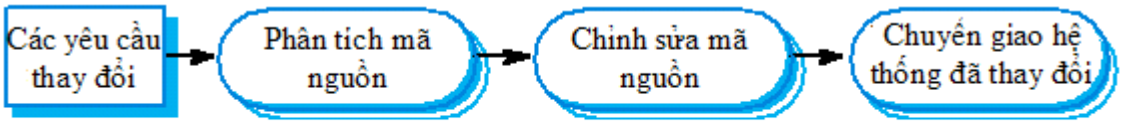
Hình 9.5: Cài đặt thay đổi

Trên đây là những quy trình cơ bản. Tuy nhiên, với các yêu cầu thay đổi khẩn cấp, ta có thể cài đặt chúng nay mà không cần phải trải qua tất cả các pha của quy trình công nghệ phần mềm. Những yêu cầu thay đổi khẩn cấp thường xảy ra khi:

- Nếu có một lỗi hệ thống nghiêm trọng xảy ra và cần phải sửa chữa.

- Nếu những thay đổi về môi trường của hệ thống gây ra những hiệu ứng không mong đợi.

- Nếu sự thay đổi về mặt nghiệp vụ yêu cầu phải có đáp ứng nhanh.



Hình 9.6:Quy trình cài đặt thay đổi khẩn cấp

Để cải tiến hệ thống hiện có, người ta đã đề xuất bốn chiến lược cơ bản:

- Tách hệ thống và chỉnh sửa các quy trình nghiệp vụ

- Tiếp tục bảo trì hệ thống

- Biến đổi hệ thống bằng cách tái kỹ nghệ để nâng cấp khả năng bảo trì của nó.

- Thay thế hệ thống bằng một hệ thống mới

Việc lựa chọn chiến lược cải tiến hệ thống phụ thuộc vào chất lượng hệ thống và giá trị nghiệp vụ của nó. Các loại hệ thống hiện có được phân loại dựa trên tiêu chí

chất lượng và giá trị nghiệp vụ mà nó mang lại như sau:

- Chất lượng thấp và giá trị nghiệp vụ thấp: những hệ thống này nền được tách ra.

- Chất lượng thấp và giá trị nghiệp vụ cao: những hệ thống này có giá trị nghiệp vụ cao nhưng chi phí bảo trì khá lớn. Ta nên tái kỹ nghệ hoặc thay thế bởi một hệ thống thích hợp

- Chất lượng cao và giá trị nghiệp vụ thấp: thay thế bằng các thành phần COTS

- Chất lượng cao và giá trị nghiệp vụ cao: tiếp tục sử dụng và bảo trì hệ thống theo cách thông thường.

Việc đánh giá giá trị nghiệp vụ được thực hiện từ nhiều khung nhìn khác nhau. Phỏng vấn các stakeholder khác nhau và đối sánh kết quả thu được. Các stakeholder thường là:

- Người sử dụng cuối

- Khách hàng của doanh nghiệp

- Người quản lý dây chuyền sản xuất

- Người quản lý công nghệ thông tin

- Người quản lý cao cấp

Đánh giá chất lượng hệ thống thông qua:

- Quy trình nghiệp vụ: quy trình nghiệp vụ đã hỗ trợ cho các mục tiêu nghiệp vụ như thế nào?

- Môi trường hệ thống: môi trường hệ thống có hiệu quả như thế nào và chi phí để bảo trì nó.

- Khả năng ứng dụng: chất lượng của ứng dụng?

Để đo hệ thống, chúng ta có thể thu thập dữ liệu định lượng để tạo ra bản đánh giá về chất lượng của hệ thống.

- Số lượng các yêu cầu thay đổi của hệ thống

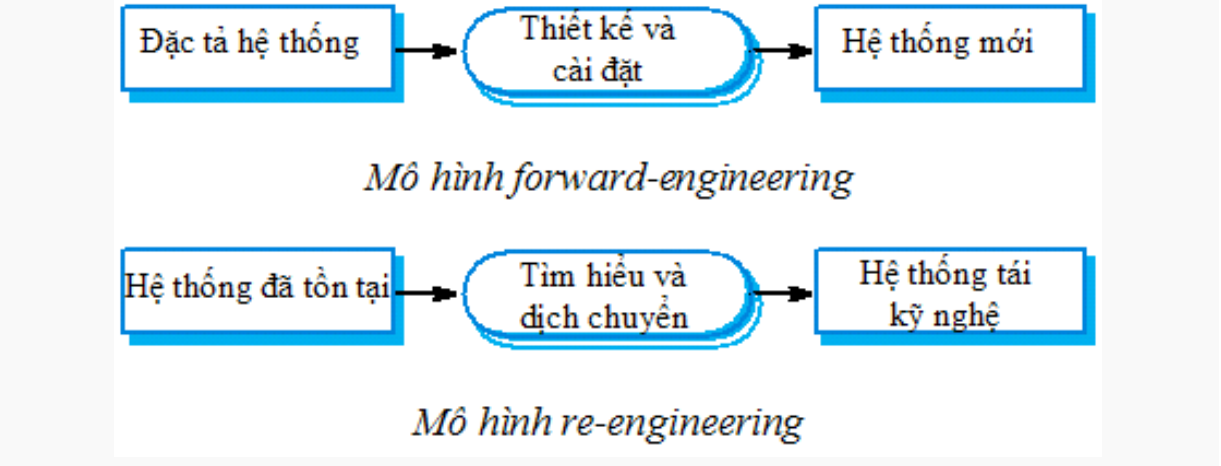
- Số lượng các giao diện người dùng khác nhau

- Số lượng dữ liệu được sử dụng trong hệ thống.

7.3 Tái kỹ nghệ hệ thống (System re-engineering)

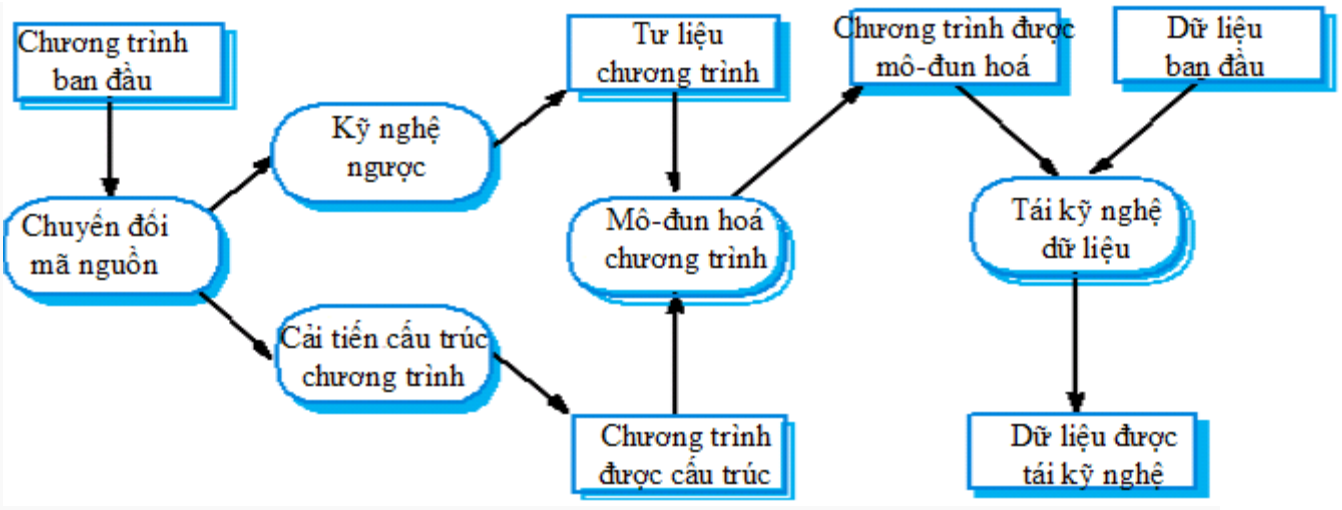
Tái kỹ nghệ hệ thống là kỹ thuật cấu trúc lại hoặc viết lại một phần hoặc toàn bộ hệ thống được thừa kế mà không thay đổi các chức năng của nó. Tái ký nghệ giúp giảm rủi ro vì trong quá trình xây dựng phần mềm mới rủi ro có thể xảy ra là khá cao và giúp giảm chi phí.

Mô hình sau đây giúp phân biệt forward và re-engineering:



Quy trình tái kỹ nghệ bao gồm các hoạt động sau:

- Dịch mã nguồn: chuyển mã lệnh thành ngôn ngữ mới.
- Kỹ nghệ ngược: phân tích chương trình để tìm hiểu nó.
- Cải thiện cấu trúc chương trình
- Mô-đun hoá chương trình: tổ chức lại cấu trúc chương trình
- Tái kỹ nghệ dữ liệu: thu dọn và cấu trúc lại dữ liệu hệ thống



Khi thực hiện hoạt động tái kỹ nghệ hệ thống, chúng ta cần quan tâm đến các nhân tố ảnh hưởng tới chi phí tái kỹ nghệ gồm:

- ✓ Chất lượng của hệ thống được tái kỹ nghệ
- ✓ Các công cụ hỗ trợ tái kỹ nghệ
- ✓ Mức mở rộng cần thiết của việc chuyển đổi dữ liệu
- ✓ Các kỹ năng của nhân viên tái kỹ nghệ hệ thống

CHƯƠNG 8. TỔNG QUAN VỀ QUẢN LÝ DỰ ÁN PHẦN MỀM

MỤC ĐÍCH:

- *Hiểu được các Khái niệm về: Dự án là gì, Quản lý dự án như thế nào?*
- *Nắm được các nội dung trong quản lý dự án phần mềm*
- *Kiến thức, kỹ năng cần thiết cho Quản lý dự án phần mềm*

8.1 Mở đầu

Quản lý dự án là một trong những lĩnh vực kiến thức mang tính kinh nghiệm, có ý nghĩa quan trọng trong các nhiệm vụ hàng ngày của bất kỳ một nhà quản lý hay một cá nhân có tham vọng trở thành nhà quản lý.

Để hiểu rõ và làm chủ được những kiến thức, nội dung xung quanh nhiệm vụ, hoạt đông quản lý dự án, cụ thể là các dự án phần mềm, trước tiên, các bạn cần phải trang bị những kiến thức cơ bản nhằm khai thông khái niệm, thuật ngữ về quản lý dự án phần mềm.

Vài số liệu thống kê về quản lý dự án

- Mỗi năm Mỹ chi 2.3 nghìn tỉ USD vào các dự án, bằng ¼ GDP của Mỹ.
- Toàn thế giới chi gần 10 nghìn tỉ USD cho tất cả các loại dự án, trong số 40.7 nghìn tỉ USD của tổng sản lượng toàn cầu.
- Hơn 16 triệu người xem quản trị dự án là nghề của mình.
- Các chuyên gia ngày càng nhấn mạnh tầm quan trọng của quản lý dự án. Tom Peters đã viết trong cuốn sách của mình "Reinventing Work: the Project 50", "Ngày nay muốn chiến thắng bạn phải nắm vững nghệ thuật quản lý dự án!"

Các Lợi ích của QLDA

- Kiểm soát tốt hơn các tài nguyên tài chính, thiết bị và con người
- Cải tiến quan hệ với khách hàng
- Rút ngắn thời gian triển khai.
- Giảm chi phí
- Tăng Chất lượng và độ tin cậy.
- Tăng Lợi nhuận.
- Cải tiến năng suất lao động
- Phối hợp nội bộ tốt hơn.
- Nâng cao Tinh thần làm việc.

8.2 Dự án là gì?

Theo quan điểm chung dự án là một lĩnh vực hoạt động đặc thù, một nhiệm vụ cần phải thực hiện theo một phương pháp riêng, trong khuôn khổ nguồn lực riêng, kế hoạch tiến độ cụ thể nhằm tạo ra một sản phẩm mới. Từ đó cho thấy, dự án có tính cụ thể, mục tiêu rõ ràng xác định để tạo ra một sản phẩm mới.

Theo PMBOK® Guide 2000, p. 4, dự án là *"một nỗ lực tạm thời được cam kết để tạo ra một sản phẩm hoặc dịch vụ duy nhất"*. Theo cách định nghĩa này, hoạt động dự án tập trung vào 2 đặc tính:

- Nỗ lực tạm thời: mọi dự án đều có điểm bắt đầu và kết thúc cụ thể. Dự án chỉ kết thúc khi đã đạt được mục tiêu dự án hoặc dự án thất bại.
- Sản phẩm và dịch vụ là duy nhất: điều này thể hiện có sự khác biệt so với những sản phẩm, dịch vụ tương tự đã có hoặc kết quả của dự án khác.

Có thể định nghĩa khái niệm về dự án một cách tổng quát nhất: ***"Dự án là một tập hợp các công việc, được thực hiện bởi một tập thể người, nhằm đạt được một kết quả dự kiến, trong một thời gian dự kiến, với một kinh phí dự kiến".***

8.2.1 Dự án Công nghệ Thông tin là gì?

CNTT = Phần cứng + Phần mềm, sự tích hợp phần cứng, Phần mềm và con người.

Dự án CNTT = DA liên quan đến phần cứng, phần mềm, và mạng.

Thí dụ DA CNTT: 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.

Dự án CNTT bắt buộc phải có phần mềm và dữ liệu. Nếu chỉ có phần cứng thì chỉ coi là một dự án mua sắm trang bị.

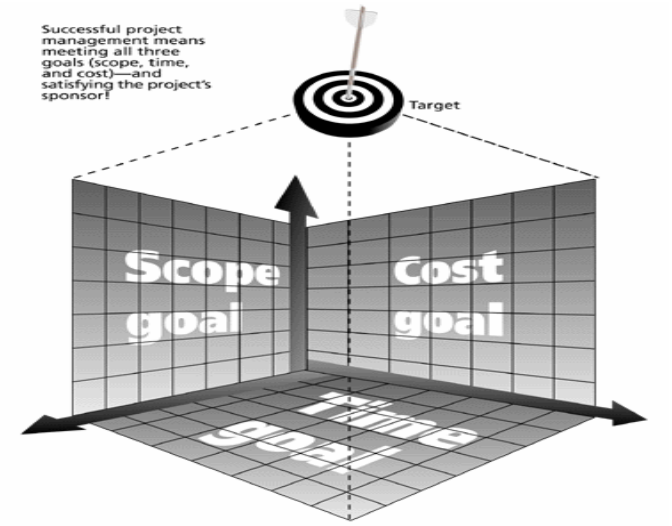
Vì vậy khi nói đến dự án CNTT và quản trị dự án CNTT thì vấn đề chủ yếu là dự án và quản trị dự án phần mềm. Vì vậy người ta quan niệm **dự án CNTT là dự án có phần mềm.**

8.2.2 Các đặc trưng của dự án

- Dự án có mục đích, kết quả rõ ràng
- Thời gian tồn tại của dự án có tính hữu hạn
- Sản phẩm, kết quả của dự án mang tính độc đáo, mới lạ
- Dự án liên quan đến nhiều bên
- Dự án thường mang tính không chắc chắn (tạm thời)
- Môi trường tổ chức, thực hiện

8.2.3 Bộ ba ràng buộc:

- Mọi dự án bị ràng buộc theo nhiều cách, do:
  - Muc tiêu về phạm vi (Scope): Dự án tìm cách đạt được cái gì?
  - Các mục tiêu về thời gian: Dự án mất bao lâu mới hoàn tất?
  - Các mục tiêu về chi phí: Sẽ tốn kém bao nhiêu?
- Nhiệm vụ của người quản lý dự án là phải cân đối những mục tiêu thường hay xung đột này.



Hình 1.1. Bộ ba ràng buộc của QLDA

8.3 Quản lý Dự án là gì?

Phương pháp quản lý dự án lần đầu tiên được áp dụng trong lĩnh vực quân sự của Mỹ vào những năm 50 của thế kỷ trước. Các lực lượng cơ bản thúc đẩy sự phát triển phương pháp quản lý dự án là:

- Nhu cầu thực tế cho thấy khách hàng ngày càng "khắt khe, khó tính" với các hàng hoá, dịch vụ, dẫn tới sự gia tăng độ phức tạp trong quy trình tổ chức, quản lý sản xuất và chất lượng sản phẩm, dịch vụ.
- Kiến thức của con người không ngừng phát triển về tự nhiên, xã hội, kinh tế, kỹ thuật …

Quản lý dự án là **"ứng dụng kiến thức, kỹ năng, công cụ và kỹ thuật vào các hoạt động dự án để thỏa mãn các yêu cầu của dự án."** (PMI, Project Management Body of Knowledge (PMBOK® Guide), 2000, p.6).

(Viện Quản lý Dự án (Project Management Institute - PMI) là một hiệp hội các chuyên gia quốc tế. Website: www.pmi.org.)

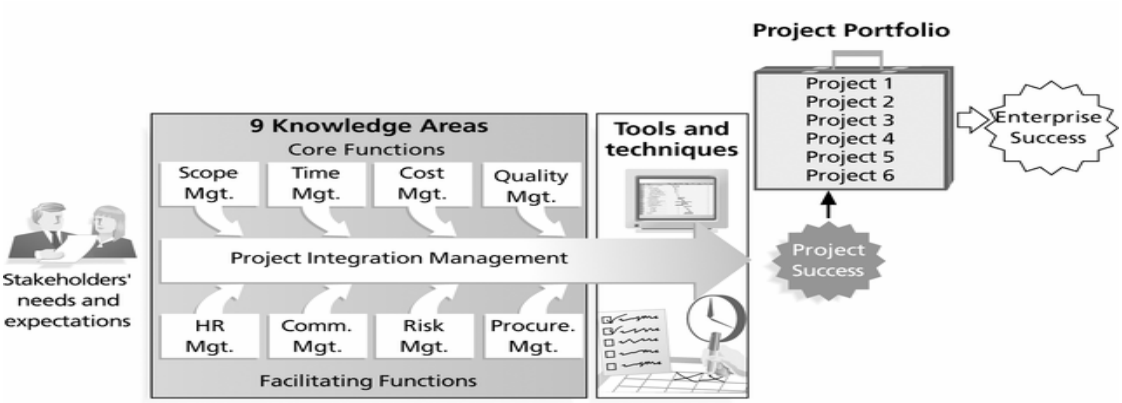


Figure 1-2. Project Management Framework

Hình 1.2. Khung làm việc của QLDA

Định nghĩa khác về QLDA : **Quản lý dự án là việc áp dụng các công cụ, kiến thức và kỹ thuật nhằm định nghĩa, lập kế hoạch, tiến hành triển khai, tổ chức, kiểm soát và kết thúc dự án.**

- Môt dự án được quản lý tốt, tức là khi kết thúc phải thoả mãn được chủ đầu tư về các mặt: thời hạn, chi phí và chất lượng kết quả.
- Một dự án được coi là thất bại nếu chi phí vượt quá dự tính 20%, thời gian vượt quá dự tính 20% hoặc tỉ lệ lỗi lớn. Tuy vậy nhiều người cho rằng nếu chi phí hoặc thời gian vượt quá 30% nhưng chất lượng tốt và đáp ứng được yêu cầu thì nên coi là thành công rực rỡ

Nghĩa là: quản lý dự án là một quá trình lập kế hoạch, điều phối thời gian, nguồn lực và giám sát quá trình phát triển của dự án nhằm đảm bảo cho dự án hoàn thành đúng thời hạn, trong phạm vi ngân sách được duyệt và đạt được các yêu cầu đã định về kỹ thuật, chất lượng của sản phẩm, dịch vụ, bằng các phương pháp và điều kiện tốt nhất cho phép.

❖ **9 Lĩnh vực quản lý dự án:**

• **4 lĩnh vực cơ bản:** Quản lý phạm vi, quản lý thời gian, quản lý chi phí, và quản lý chất lượng

• **4 lĩnh vực hỗ trợ** là phương tiện để đạt các mục tiêu của dự án: Quản lý nguồn nhân lực, quản lý truyền thông, quản lý rủi ro, và quản lý mua sắm trang thiết bị

• **1 lĩnh vực tích hợp** (project integration management) tác động và bị tác động bởi tất cả các lĩnh vực ở trên

Trong đó:

- Quản lý phạm vi: Là việc xác định phạm vi, giám sát việc thực hiện mục đích,

mục tiêu của dự án, xác định công việc nào thuộc về dự án và cần phải thực hiện, công việc nào nằm ngoài phạm vi của dự án.

- Quản lý thời gian: Là việc lập kế hoạch, phân phối và giám sát tiến độ thời gian nhằm đảm bảo thời hạn hoàn thành dự án. Nó chỉ rõ mỗi công việc phải kéo dài bao lâu, khi nào thì bắt đầu, khi nào thì kết thúc và toàn bộ dự án kéo dài bao lâu, phải hoàn thành khi nào.
- Quản lý chi phí: Là quá trình dự toán kinh phí, giám sát thực hiện chi phí theo tiến độ cho từng công việc và toàn bộ dự án. Cụ thể là tổ chức, phân tích số liệu, báo cáo những thông tin về chi phí.
- Quản lý chất lượng: Là quá trình triển khai giám sát những tiêu chuẩn chất lượng cho việc thực hiện dự án, đảm bảo chất lượng kết quả của dự án phải đáp ứng mong muốn của nhà tài trợ (chủ đầu tư).
- Quản lý nguồn nhân lực: Là quá trình hướng dẫn, phối hợp những nỗ lực của mọi thành viên tham gia dự án vào việc hoàn thành mục tiêu của dự án. Nó cho thấy việc sử dụng lực lượng lao động của dự án hiệu quả đến đâu?
- Quản lý thông tin (truyền thông): Là quá trình bảo đảm các dòng thông tin thông suốt, nhanh chóng và chính xác giữa các thành viên dự án và với các cấp quản lý, giữa các tổ nhóm quản lý dự án. Thông qua quản lý thông tin có thể trả lời các câu hỏi: ai cần thông tin về dự án? mức độ chi tiết? các nhà quản lý dự án cần báo cáo cho họ bằng cách nào?
- Quản lý rủi ro: Là việc nhận diện các nhân tố rủi ro trong dự án, sử dụng các phương pháp định tính, định lượng để xác đinh tính chất, mức độ rủi ro và có kế hoạch đối phó cũng như quản lý từng loại rủi ro.
- Quản lý hợp đồng và các mua sắm trang thiết bị: Là quá trình lựa chọn nhà cung cấp hàng hoá và dịch vụ; thương lượng với họ, quản lý các hợp đồng và điều hành việc mua bán nguyên vật liệu, trang thiết bị, dịch vụ nhằm giải quyết cácvấn đề: bằng cách nào cung cấp các hàng hoá, vật liệu cần thiết cho dự án? tiến độ cung cấp, chất lượng cung cấp đến đâu?

8.4 Các giai đoạn của tiến trình quản trị dự án

Gồm 7 giai đoạn:

	Mục đích	Các hoạt động trong từng giai	Tiến hành	Tài liệu và các mốc điểm
XÁC ĐỊNH	Tìm hiểu để có đánh giá khởi đầu.	Mục đích, mục tiêu Trình bày vấn đề. Đáng giá rủi ro. Kế hoạch & ước tính.	Quản Lý DA.	Ý tưởng về DA (NDùng Thông qua) Yêu cầu Ndùng. Bảng các Rủi ro. Kế hoạch Khởi đầu. (Các
PHÂN TÍCH	Hệ thống sẽ làm gì	Giao diện người dùng. Các điều khoản hợp đồng.	Xem xét,	Đặc tả Chức năng (Ndùng thông qua) Kế hoạch cuối cùng

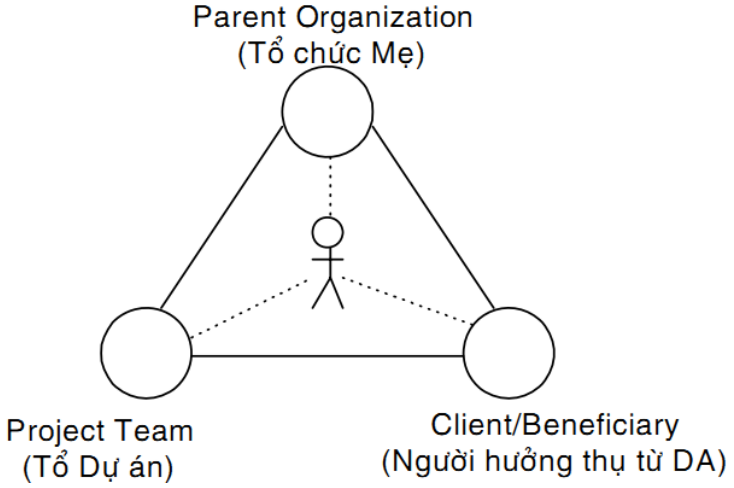
THIẾT KẾ	Các phần của Hệ thống, Hệ thống sẽ làm việc như thế	Quyết định xây dựng/Mua. Thiết kế Xem xét kỹ lưởng.	Báo cáo Tình hình.	Đặc tả Thiết kế (Thông qua KT) Kế hoạch kiểm thử Chấp nhận
THỰC HIỆN	Lắp ráp các thành phần	Lập trình. Xây dựng/Mua. Khách hàng hóa. Kiểm thử từng phần.		Thiết kế các Thành phần. (Thông qua KT) Kế hoạch Kiểm thử Hệ thống. (Thông qua KT) Các Thành phần đã
KIỂM THỬ HỆ THỐNG	Làm việc, Hệ thống hiệu chỉnh những sai	Tích hợp. Kiểm tra chất lượng		Hệ thống làm việc Kiểm thử Hệ thống.

KIỂM THỬ CHẤP NHẬN	Sự chấp nhận của khach hàng.	Qui trình Chấp nhận		Kiểm thử Chấp nhận (Ndùng thông qua)
VẬN HÀNH	Cài đặt rộng rãi và hoàn thành.	Cài đặt rộng rãi. Chuyển đổi. Đào tạo, Hỗ trợ, Xem xét.		Hệ thống mới có được dùng?(Ndùng) Báo cáo Đào tạo Kế hoạch Hỗ trợ. (Ndùng thông qua)

Bảng 1-1. Các công việc trong từng giai đoạn vòng đời dự án

8.5 Vai trò, trách nhiệm của người quản lý dự án

8.5.1 Vị trí của nhà quản lý dự án trong bối cảnh chung của dự án



Nhà quản lý dự án trong một môi trường đầy mâu thuẫn:

- Các dự án cạnh tranh về nguồn lực
- Mâu thuẫn giữa các thành viên trong dự án
- Khách hàng muốn thay đổi yêu cầu
- Các nhà quản lý của tổ chức "mẹ" muốn giảm chi phí

Người quản lý giỏi sẽ phải giải quyết nhiều mâu thuẫn trên.

8.5.2 vai trò của nhà quản lý dự án

- Quản lý các mối quan hệ giữa người và người trong các tổ chức của dự án.
- Phải duy trì sự cân bằng giữa chức năng: quản lý dự án và kỹ thuật của dự án.
- Đương đầu với rủi ro trong quá trình quản lý dự án.
- Tồn tại với điều kiện ràng buộc của dự án.

Do đó, nhà quản lý dự án phải lập kế hoạch, tổ chức, lãnh đạo và kiểm tra.

8.5.3 Các kỹ năng cần thiết của người quản lý dự án

- Kỹ năng tổ chức: lập kế hoạch, xác định mục tiêu, phân tích. Nhà quản lý dự án phải là người chịu trách nhiệm về kế hoạch tổng thể trước nhà tài trợ và khách hàng. Vì vậy, nhà quản lý dự án phải có kỹ năng lập lịch trình dự án và xác định các tiêu chí để đánh giá công việc hoàn thành. Đồng thời, nàh quản lý dự án phải biết thiết lập các quy trình hệ thống để đánh giá và kiểm soát mức độ thành công của bảng kế hoạch.
- Kỹ năng xây dựng nhóm: thấu hiểu, thúc đẩy, tinh thần đồng đội.
- Kỹ năng lãnh đạo: năng động, có tầm nhìn, biết giao nhiệm vụ, lạc quan. Lãnh đạo là kỹ năng cơ bản để nhà quản lý dự án chỉ đạo, định hướng, khuyến khích và phối hợp các thành viên trong nhóm cùng thực hiện dự án. Đây là kỹ năng quan trọng nhất. Nó đòi hỏi các nhà quản lý dự án có những phẩm chất cần thiết, có quyền lực nhất định để thực hiện thành công mục tiêu dự án.
- Kỹ năng giao tiếp và thông tin trong quản lý dự án: lắng nghe, thuyết phục. Nhà quản lý dự án có trách nhiệm phối hợp, thống nhất các hoạt động giữa các bộ phận chức năng và những cơ quan liên quan để thực hiện các công việc của dự án nên bắt buộc phải thành thạo kỹ năng giao tiếp. Nhà quản lý dự án phải có kiến thức, hiểu biết các công việc của các phòng chức năng, có kiến thức rộng về một số lĩnh vực kỹ thuật. Nhà quản lý dự án cũng cần giỏi kỹ năng thông tin, truyền thông, kỹ năng chia sẻ thông tin giữa các thành viên dự án và những người liên quan trong quá trình triển khai dự án.
- Kỹ năng đối phó: linh hoạt, sáng tạo, kiên trì, chịu đựng.
- Kỹ năng công nghệ: kinh nghiệm, kiến thức về dự án
- Kỹ năng thương lượng và giải quyết khó khăn vướng mắc: Nhà quản lý dự án trong quá trình thực hiện trọng trách của mình có quan hệ với rất nhiều nhóm. Đồng thời, cùng với sự phát triển tổ chức của dự án, trách nhiệm của nhà quản lý dự án ngày càng tăng nhưng quyền lực của họ được cấp không tương xứng. Do thiếu quyền lực, bắt buộc các nhà quản lý phải có kỹ năng thương lượng giỏi với các nhà quản lý cấp trên và những người đứng đầu các bộ phận chức năng chuyên môn nhằm tranh thủ tối đa sự quan tâm, ủng hộ của cấp trên, người đứng đầu trong việc giành đủ nguồn lực cần thiết cho hoạt động của dự án.
- Kỹ năng tiếp thị và quan hệ khách hàng: Một trong những nhiệm vụ quan trọng nhất của nhà quản lý dự án là trợ giúp các đơn vị, doanh nghiệp trong hoạt

động Marketing. Làm tốt công tác tiếp thị sẽ giúp đơn vị giữ được khách hàng hiện tại, tăng thêm khách hàng tiền năng.

- ❖ Kỹ năng ra quyết định: Lựa chọn phương án và cách thức thực hiện các công việc dự án là những quyết định rất quan trọng, đặc biệt trong những điều kiện thiếu thông tin và có nhiều thay đổi, biến động. Để ra được quyết định đúng và kịp thời cần nhiều kỹ năng tổng hợp của nhà quản lý như: kỹ năng tổ chức bao gồm lập kế hoạch, xác định mục tiêu, phân tích; kỹ năng xây dựng nhóm như thấu hiểu, thúc đẩy, tinh thần đồng đội và kỹ năng công nghệ liên quan đến kinh nghiệm, kiến thức về dự án.

8.5.4 Phẩm chất của nhà quản lý dự án

- Thật thà và chính trực (Honesty and Integrity).
- Khả năng ra quyết định (Decision Making Ability).
- Hiểu biết các vấn đề về con người (Understanding of Personal Problem).
- Tính chất linh hoạt, đa năng, nhiều tài (Versatility).

8.6 Ước lượng dự án

8.6.1 Ước lượng thời gian các hoạt động

Một số phương pháp được dùng để ước lượng thời gian thực hiện dự án là:

- Ước lượng thời gian dự án theo mốc thời gian (Milestone Schedule).
- Ước lượng thời gian dự án theo bảng cấu trúc phân rã công việc (WBS)
- Ước lượng thời gian dự án theo sơ đồ Gantt (sơ đồ thanh ngang)
- Ước lượng thời gian dự án theo sơ đồ mạng (Network System)

8.6.2 Ước lượng chi phí dự án

Nghiên cứu trong ngành chỉ ra rằng phần lớn các dự án công nghệ thông tin theo nguồn lực hơn là theo lịch trình. Điều đó có nghĩa là khi đẩy mạnh thì chi phí dự án quan trọng hơn việc dự án mất bao lâu.

Đầu ra quan trọng của quản lý chi phí dự án là ước tính chi phí. Có nhiều loại phương pháp ước tính chi phí và theo đó có những công cụ kỹ thuật giúp tính toán.

Một số phương pháp ước lượng chi phí:

a) Ước lượng chính quy

Ước lượng chính quy được dùng để chỉ ước lượng gần đúng. Ước lượng chính quy dựa trên sự phân tích. Trong một thế giới lý tưởng, phân tích này sẽ được tiến hành theo chiều sâu. Ít nhất là một phân tích mở đầu phải được tiến hành. Một ước lượng gồm có 3 thành phần chính:

- Danh sách các giả định được sử dụng trong việc xây dựng ước lượng (Ví dụ

như các chi phí đầu vào về lao động và nguyên vật liệu).

- Phạm vi biến động cho ước lượng được đưa ra (Ví dụ +/- 50%)
- Khoảng thời gian ước lượng có hiệu lực (Ví dụ như ước lượng này có hiệu lực trong vòng 60 ngày).

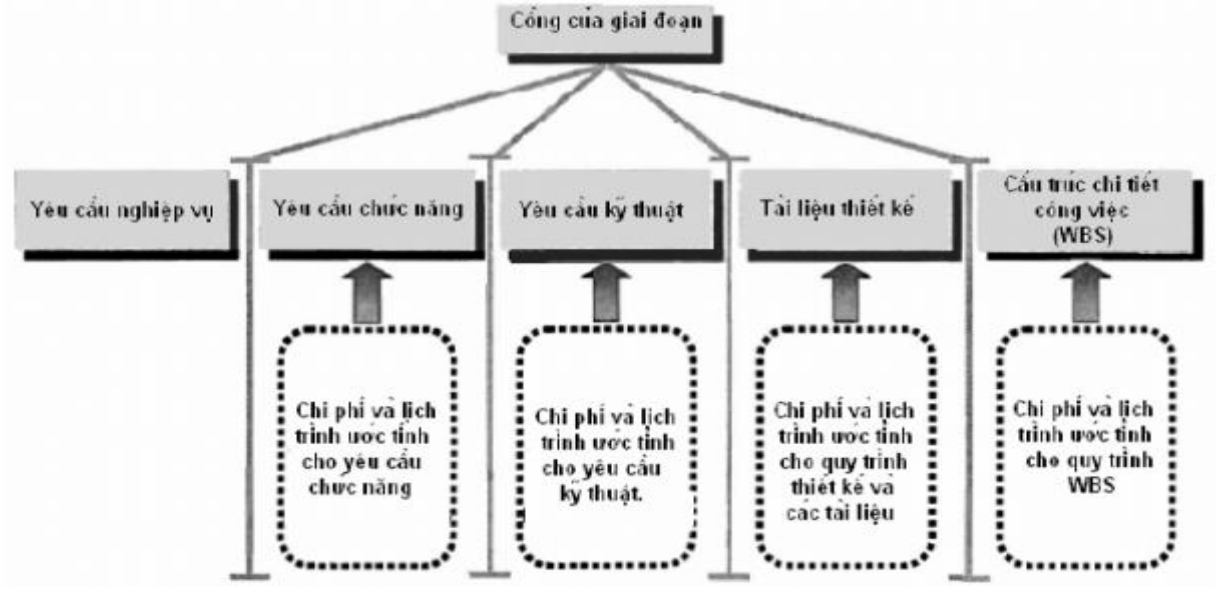
Ngược lại, ước lượng không chính quy là ước đoán dựa trên sự suy đoán, phỏng đoán và bản năng.

b) Ước lượng theo giai đoạn

Xác định giai đoạn là phương pháp tách các nhóm hoạt động của dự án thành hàng loạt các giai đoạn liên tiếp. Đánh giá hiệu quả và các phần có thể chuyển giao của dự án diễn ra ở cuối mỗi giai đoạn trước khi dự án chuyển sang giai đoạn tiếp theo. Đôi khi các đánh giá này chỉ các cổng của giai đoạn. Phương pháp này được sử dụng đầu tiên khi không thật chắc chắn về những thứ thực sự liên quan đến dự án hay thực hiện vòng đời sản phẩm và tách thành các vòng đời dự án nhỏ hơn.

Ước lượng theo giai đoạn là một kỹ thuật trong đó ước tính chi phí và lịch trình được xây dựng riêng cho từng giai đoạn của dự án. Phương pháp này được sử dụng đầu tiên khi không thật chắc chắn về những thứ thực sự liên quan đến dự án. Hơn nữa xây dựng một ước tính lớn hầu như chỉ là công việc dự đoán, dự án được chia thành các phần và ước tính mới được xây dựng cho từng phần của dự án.

Xác lập cổng của giai đoạn: Không có nguyên tắc bất di bất dịch nào về vị trí cổng của giai đoạn được xác lập trong dự án nhưng các phần có thể chuyển giao và các quyết định cần thiết phải được phác thảo rõ ràng cho từng giai đoạn.



c) Ước lượng theo tham số

Ước lượng theo tham số lấy kiến thức thu được từ các dự án tương tự nhưng

không chính xác, đồng thời sử dụng các tham số như chi phí trên đơn vị để ước tính thông tin lịch trình và chi phí.

Ước lượng theo tham số có thể sử dụng cho các dự án lớn bằng cách phân chia chúng thành các đơn vị công việc nhỏ và đưa vào một mô hình toán học.

Một số tổ chức sử dụng ước lượng theo tham số ở các mức độ cấu trúc chi tiết công việc thấp hơn, ở đó họ có nhiều dữ liệu chính xác và chi tiết hơn và sau đó kết hợp kết quả vào các mẫu đã được xây dựng trước hợp lại thành ước lượng chi tiết có độ chính xác cao.

Các phương pháp ước lượng theo tham số phổ biên hiện nay là:

- Phương pháp COCOMO dựa trên KLOC (Kilo Line Of Codes)
- Phương pháp điểm chức năng – Function Point
- Phương pháp điểm trường hợp sử dụng – UseCase Point
- Phương pháp COSMIC FFP: Full Function Point
- Ngoài ra, còn những phương pháp khác như điểm đối tượng, điểm đặc tính (Feature Point)…

d) Ước lượng dưới lên

Ước lượng dưới lên là một kỹ thuật ước lượng mất nhiều thời gian nhưng cực kỳ chính xác. Ước lượng dưới lên ước tính chi phí và lịch trình ở mức độ gói công việc của cấu trúc chi tiết công việc và sau đó tổng hợp các con số này để tính tổng số cho dự án. Phương pháp này cần một cấu trúc chi tiết công việc và dựa vào một số giả định:

- **Khả năng**. Người đang ước tính chi phí và lịch trình cho các gói công việc phải biết công việc thực sự được tiến hành như thế nào.
- **Tính chính trực**. Nếu người đang thực hiện công việc tham gia vào ước tính thì họ không thể đánh giá quá cao hoặc quá thấp về thời gian cần để hoàn thành công việc.
- **Độ chính xác.**

e) Ước lượng trên xuống

Ước lượng trên xuống là một kỹ thuật bắt đầu bằng một ước tính cho toàn bộ dự án và sau đó chia ra thành tỉ lệ phần trăm trong tổng số đối với mỗi giai đoạn hay loại công việc dự án. Điều này được thực hiện dựa vào công thức thu được từ các dữ liệu lịch sử do các dự án tương tự cung cấp. Phương pháp này cần một cấu trúc chi tiết công việc và dựa vào một số giả định:

- Tính tương tự của dự án. Công thức phân chia các nguồn lực dựa vào các dữ liệu lịch sử của một loại dự án cụ thể. Nếu dự án đang được ước tính khác nhau về cơ bản so với dự án dùng để xây dựng công thức thì công thức sẽ không chính xác.
- Độ chính xác của toàn bộ ước tính. Do kỹ thuật trên xuống phân chia ước tính cho toàn bộ dự án thành các giai đoạn khác nhau nên độ chính xác của toàn bộ ước tính mang tính chất quyết định.

Do ước lượng trên xuống cần có thông tin lịch sử nên không thể thực hiện ước lượng trên xuống cho một dự án chưa từng được thực hiện trước đây.

Có ba cơ sở lập luận giải thích lý do tại sao nhiều ước lượng trên xuống cho các dự án công nghệ thông tin có xu hướng thất bại:

- Sự hiểu biết rõ ràng của quản lý về quy trình trên xuống biến nó trở thành một kỹ thuật phổ biến nhất dùng trong ước lượng và dự toán các dự án công nghệ thông tin.
- Phần lớn các dự án công nghệ thông tin chưa từng được thực hiện trước đây.
- Ước lượng trên xuống cần có một cấu trúc chi tiết công việc và các dữ liệu lịch sử, do đó không thể dùng cho dự án chưa từng được thực hiện trước đây.

Vì vậy kỹ thuật ước lượng trên xuống ít khi được sử dụng trong các dự án công nghệ thông tin.