

**TRƯỜNG ĐH CNTT&TT
KHOA CÔNG NGHỆ THÔNG TIN**



**Giáo trình:
MÔN HỌC CÔNG NGHỆ PHẦN MỀM**

Năm 2012-2013

MỤC LỤC

LỜI NÓI ĐẦU	5
Chương 1 TỔNG QUAN VỀ PHẦN MỀM VÀ KỸ NGHỆ PHẦN MỀM	6
1.1. Tổng quan về phần mềm	6
1.1.1 Phần mềm (software) là gì	6
1.1.2 Vai trò của phần mềm	7
1.1.3 Những đặc trưng của phần mềm	8
1.1.4 Phân loại phần mềm.....	10
1.1.5 Tiết hóa phần mềm và những thách thức đặt ra	14
1.2.Kỹ nghệ phần mềm (Software Engineering)	17
1.2.1 Lý do ra đời	17
1.2.2 Các định nghĩa về Kỹ nghệ phần mềm	18
1.2.3 Quá trình phát triển của Kỹ nghệ phần mềm	22
1.3 FQAs về Công nghệ phần mềm	23
1.3.1 Công nghệ (engineering) là gì ?	23
1.3.2 Sự khác biệt giữa công nghệ phần mềm và khoa học máy tính là gì?.....	23
1.3.3. Phân biệt các lĩnh vực tính toán liên quan đến kỹ nghệ phần mềm	24
1.3.4 Kỹ nghệ hệ thống là gì?	24
1.3.6. Vòng đời phát triển của một hệ thống phần mềm.....	26
1.3.7 Tiết trình phần mềm là gì?	27
1.3.8. Mô hình quy trình phần mềm là gì?.....	29
1.3.9. Chi phí của kỹ nghệ phần mềm là gì?.....	29
1.3.10. Phần mềm kỹ nghệ tốt là gì.....	32
1.3.11.CASE (Computer-Aided Software Engineering) tools là gì.....	33
1.3.12. Những thách thức chính đối với công nghệ phần mềm là gì?.....	34
1.4 Các trách nhiệm đạo đức và nghề nghiệp ([6])	34
1.4.1 Các vấn đề về trách nhiệm nghề nghiệp	34
1.4.2 Tập các chuẩn mực đạo đức.....	35
1.5 Nhân tố con người và sự phân hóa nghề nghiệp trong CNPM	35
1.5.1 Nhân tố con người trong ngành công nghệ phần mềm	35
1.5.2 Phân loại nghề nghiệp trong CNPM	36
Chương 2 TIẾN TRÌNH PHẦN MỀM	42
2.1 Giới thiệu	42
2.2 Các loại mô hình tiến trình phần mềm	43
2.2.1 Mô hình thác nước (Waterfall model).....	44
1.1.2. Các mô hình phát triển tiến hóa	45
1.1.3 Phát triển các hệ thống hình thức hóa (Formal System Development).....	52
1.1.4.Phát triển hướng sử dụng lại	53
2.2. Các hoạt động trong tiến trình phần mềm	58
2.2.1. Đặc tả yêu cầu phần mềm	58
2.2.2 Thiết kế phần mềm và cài đặt	60
2.2.3 Dánh giá phần mềm	64
2.2.4. Cải tiến phần mềm	65
2.3 Công nghệ CASE	66
2.4 Các vấn đề liên quan đến tiến trình phần mềm	68
2.5 Quan hệ giữa tiến trình và sản phẩm	69
Chương 3 PHÂN TÍCH VÀ ĐẶC TẢ CÁC YÊU CẦU PHẦN MỀM	71
3.1 Tổng quan về phân tích và đặc tả yêu cầu.....	71
3.2 Các yêu cầu phần mềm và mục tiêu	71
3.2.1 Yêu cầu phần mềm là gì.....	71
3.2.1 Phân loại các yêu cầu phần mềm.	74

3.3 Tiết trình kỹ nghệ yêu cầu	77
3.3.1 Phân tích tính khả thi của dự án	78
3.3.2 Phân tích và rút ra các yêu cầu	81
3.3.3 Đặc tả yêu cầu	89
3.3.4. Đánh giá yêu cầu	98
3.3.5. Quản lý các yêu cầu	99
Chương 4 CÁC MÔ HÌNH HỆ THỐNG	102
4.1 Giới thiệu	102
4.2 Mô hình hóa hệ thống	102
4.2.1. Mô hình ngũ cảnh	103
4.2. 2 Mô hình ứng xử	104
4.2.3. Các mô hình dữ liệu	108
4.4. Các mô hình đối tượng	110
4.4.1. Mô hình thừa kế	110
4.4.2. Mô hình kết hợp	112
4.4.3. Mô hình ứng xử	113
4.5. Phương pháp hướng cấu trúc	114
Chương 5 THIẾT KẾ VÀ CÀI ĐẶT HỆ THỐNG	118
5.1 Thiết kế phần mềm	118
5.1.1 Thiết kế phần mềm là gì	118
5.1.2 Vai trò của thiết kế	118
5.1.3 Các khái niệm thiết kế cơ sở	118
5.1.4 Thiết kế thiết kế	127
5.1.5 Nguyên lý thiết kế	127
5.1.6 Tiến trình thiết kế	128
5.2. Thiết kế kiến trúc	130
5.2.1 Tổng quan về thiết kế kiến trúc	130
5.2.2 Các lợi ích của kiến trúc hệ thống rõ ràng	131
5.2.3 Kiến trúc và các đặt tính hệ thống	131
5.2.4 Các xung đột kiến trúc	131
5.2.5 Các quyết định thiết kế kiến trúc	132
5.2.6 Sử dụng lại kiến trúc	132
5.2.7 Các mô hình kiến trúc	133
5.2.8 Tiến trình thiết kế kiến trúc	133
5.2.9 Xây dựng kiến trúc chương trình từ các biểu đồ luồng dữ liệu	145
5.2.10 Thiết kế hướng đối tượng	150
5.3 Thiết kế giao diện người dùng	152
5.3.1 Vai trò, tầm quan trọng của giao diện người - máy	152
5.3.2 Tác nhân con người trong thiết kế giao diện	153
5.3.3 Các nguyên tắc thiết kế giao diện	153
5.3.4 Các vấn đề trong thiết kế giao diện UI	155
5.3.5 Biểu diễn thông tin	156
5.3.6	Tiến trình thiết kế giao diện chung
	162
5.4 Thiết kế cấu trúc dữ liệu	167
5.5 Thiết kế thuật toán/thủ tục	168
Chương 6 LẬP TRÌNH	171
6.1 Giới thiệu	171
6.2 Các yêu cầu đối với lập trình viên	171
6.3 Tiến hóa của kỹ thuật lập trình	171
6.3.1 Lập trình tuần tự/tuyến tính	172
6.3.2	Lập trình có cấu trúc/thủ tục
	172

6.3.3	Lập trình hướng hàm
.....	172
6.3.4	Lập trình hướng đối tượng
.....	173
6.3.5	Lập trình logic
.....	173
6.3.6 Kỹ thuật lập trình thế hệ thứ 4	173
6.4 Chọn ngôn ngữ lập trình cho ứng dụng	174
6.5 Một số nguyên tắc lập trình	175
Chương 7 XÁC MINH VÀ THẨM ĐỊNH PHẦN MỀM	177
7.1 Giới thiệu về xác minh và thẩm định	177
7.1.1 Khái niệm về xác minh và thẩm định	177
7.1.2 Các hoạt động xác minh	177
7.1.3 Các hoạt động thẩm định	178
7.1.4 Các hình thức thẩm định và xác minh	178
7.2 Rà soát phần mềm	179
7.2.1 Khái niệm rà soát phần mềm	179
7.2.2 Các hình thức rà soát và tiến trình rà soát	179
7.3 Kiểm thử phần mềm	182
7.3.1 Khái niệm	182
7.3.2 Một số khái niệm cơ bản	183
7.3.3 Kế hoạch kiểm thử	184
7.3.4 Tiến trình kiểm thử	184
7.3.5 Phương pháp và chiến lược kiểm thử	187
7.4 Kiểm thử trong tiến trình phát triển phần mềm	203
7.4.1 Kiểm thử trong mô hình chữ V	203
7.4.2 Kiểm thử thành phần (unit Testing)	203
7.4.3 Kiểm thử tích hợp (Integration Testing)	204
7.4.4 Kiểm thử hệ thống	206
7.4.5 Kiểm thử hợp thức hóa (validation testing)	207
7.4.6 Kiểm thử hồi quy (Regression Testing)	208
CHƯƠNG 8 BẢO TRÌ, CẢI TIẾN PHẦN MỀM	210
8.1 Bảo trì phần mềm	210
8.1.1	FQA về bảo trì phần mềm
.....	210
8.1.2. Dự đoán bảo trì	214
8.1.3. Dự đoán thay đổi	214
8.2 Bảo trì cải tiến phần mềm	215
8.2.1 Quy trình bảo trì cải tiến phần mềm	215
8.2.2 Tái kỹ nghệ hệ thống	218
Chương 9 QUẢN LÝ DỰ ÁN	220
9.1 Tổng quan về quản lý dự án	220
9.1.1 Định nghĩa	220
9.1.2 Các đặc trưng của dự án phần mềm	220
9.1.3 Thực trạng các dự án phần mềm	220
9.1.4 Mục tiêu, phương châm quản lý dự án	221
9.1.5 Tiến trình tổng quan triển khai dự án	221
8.1.6 Các chức năng quản lý dự án	222
9.2 Xác định dự án	223
9.2.1 Giới thiệu	223
9.2.2 Tiến trình xác định dự án	223
9.3 Lập kế hoạch dự án	227
TÀI LIỆU THAM KHẢO	231

LỜI NÓI ĐẦU

Chương 1 TỔNG QUAN VỀ PHẦN MỀM VÀ KỸ NGHỆ PHẦN MỀM

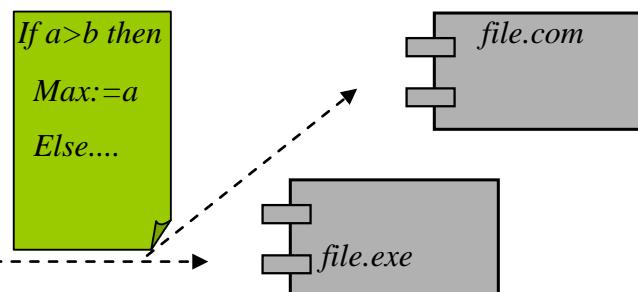
1.1. Tổng quan về phần mềm

1.1.1 Phần mềm (software) là gì

Nhiều người đánh đồng phần mềm với các chương trình máy tính. Ở đây chúng ta muốn có một khái niệm rộng hơn về phần mềm, ở đó phần mềm không chỉ là các chương trình máy tính mà còn được đính kèm bởi các tài liệu, 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:

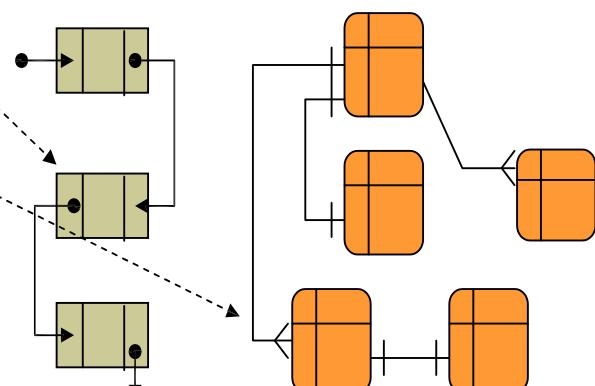
- ✓ Các *chương trình máy tính riêng lẻ*:

- Các file mã nguồn
- Các file mã máy (file text)



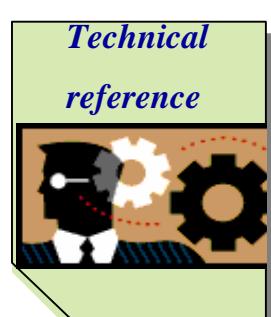
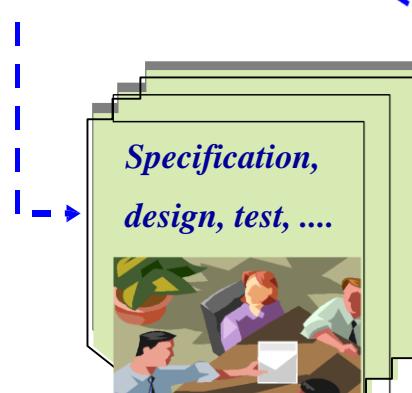
- ✓ Các *cấu trúc dữ liệu*:

- 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*)



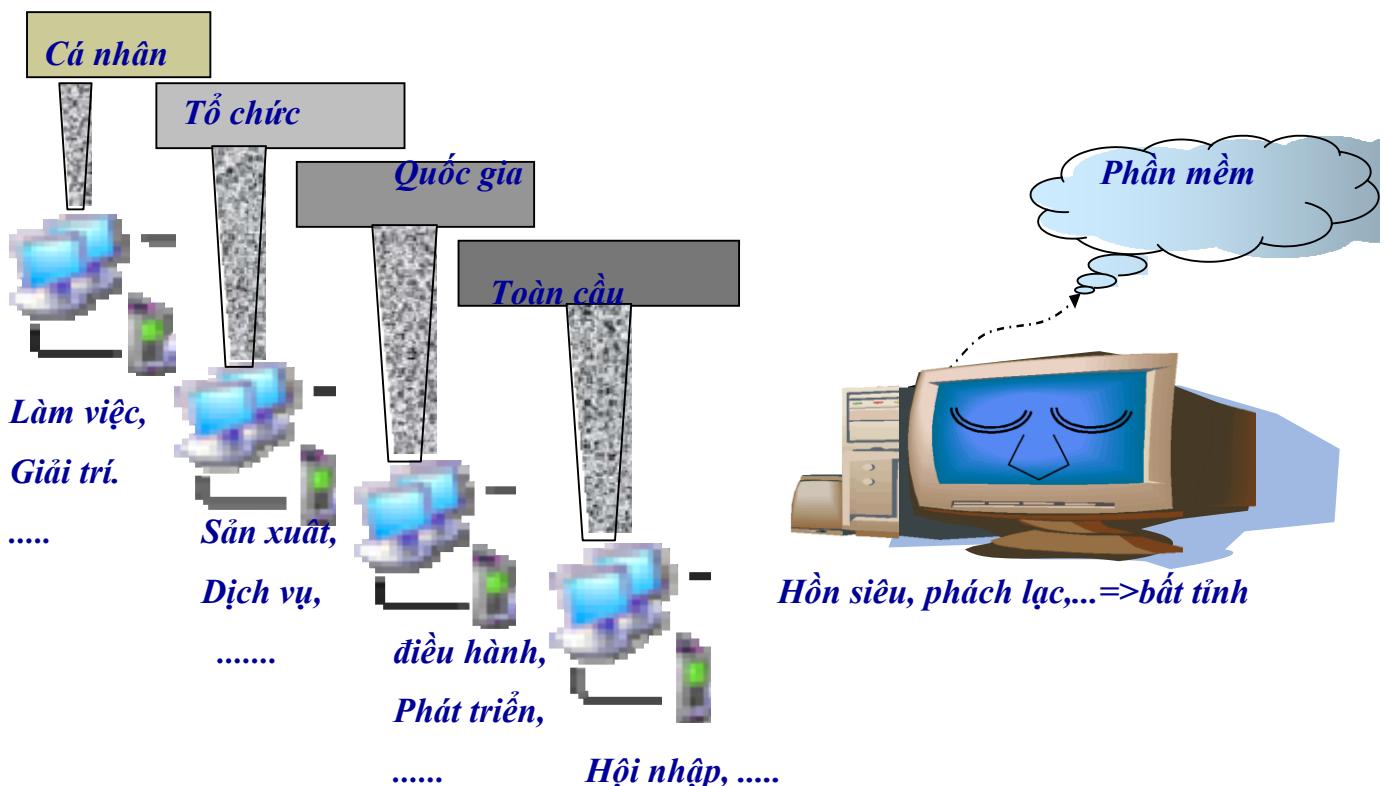
✓ Ngoài ra cần một Website cho người dùng để download thông tin sản phẩm hiện thời. Khi xây dựng một phần mềm, chúng ta cần sinh ra các thành phần này. Các thành phần vận hành được bao gồm các cấu trúc dữ liệu, các file mã nguồn, mã máy. Những thành phần này với các kỹ thuật hiện tại ta có thể phát sinh tự động được. Tuy nhiên với các thành phần không vận hành được (các thành phần còn lại) hiện tại hầu như chưa được tự động hóa vì độ phức tạp vốn có của nó. Thực tế cho thấy việc viết những tài liệu này là công việc cực nhọc, tốn nhiều thời gian và công sức. Chúng ta chỉ có thể tự động hóa được hoạt động phát sinh tài liệu khi ta có thể hình thức hóa được các tài liệu này.

Một 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.

1.1.2 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



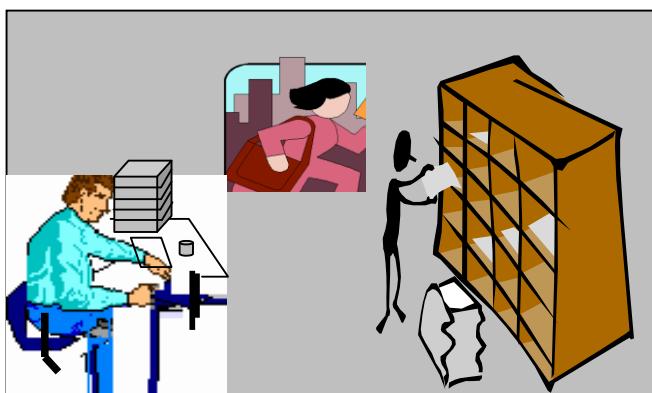
Mọi nền kinh tế đều phụ thuộc rất lớn vào phần mềm.

- Phần % thu, chi từ phần mềm chiếm đáng kể trong tổng GNP của mọi quốc gia:
 - ✓ 2006 ước độ xuất gần 30 tỉ USD phần mềm

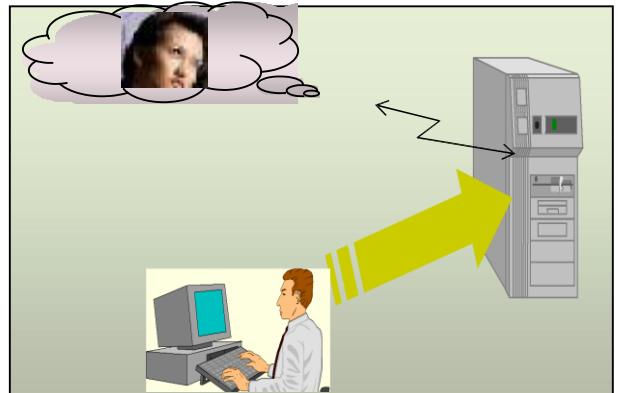
- ✓ Thế giới có >7 triệu kỹ sư CNTT tạo ra **600 tảng \$/năm**
- ✓ Chi phí cho phần mềm năm 2000 lên tới: **770 tảng \$**
- Phần mềm sai hỏng, kinh tế tổn thất lớn
 - ✓ Vệ tinh Ariane 5 hỏng do lỗi phần mềm (1996) thiệt hại 500 triệu \$.
 - ✓ Website dùng 1 ngày mất hàng triệu \$.

[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 lao động:



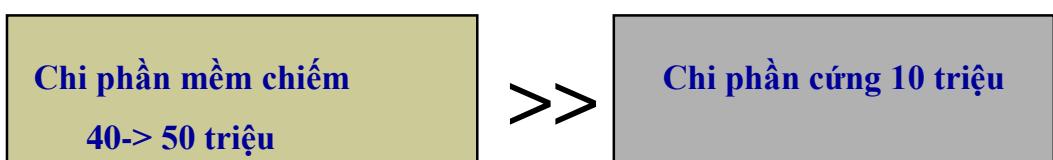
Trời ơi!



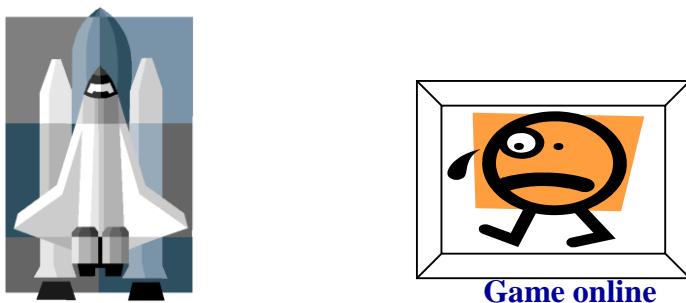
Tuyệt vời!

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,

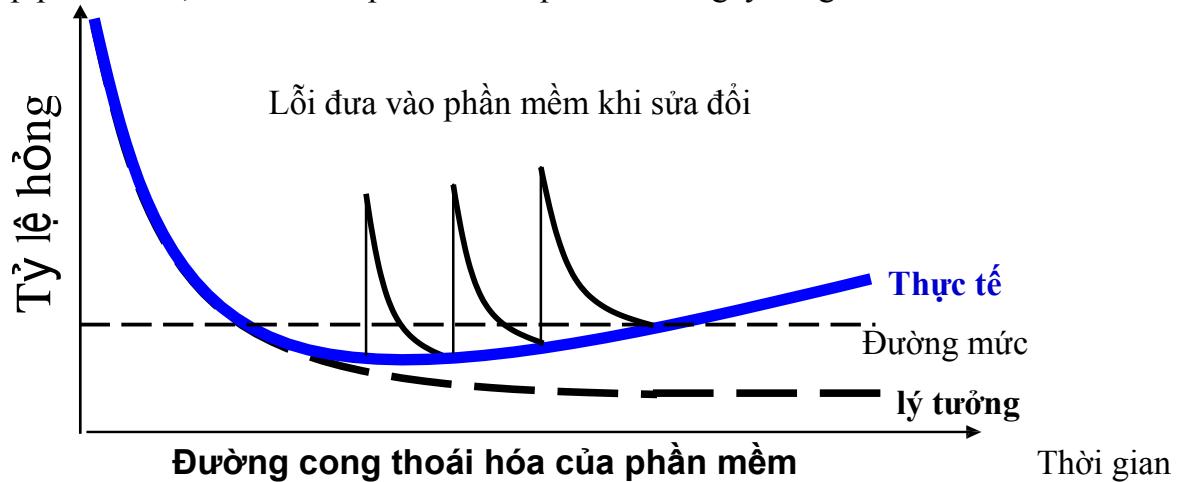


1.1.3 Những đặc trưng của phần mềm

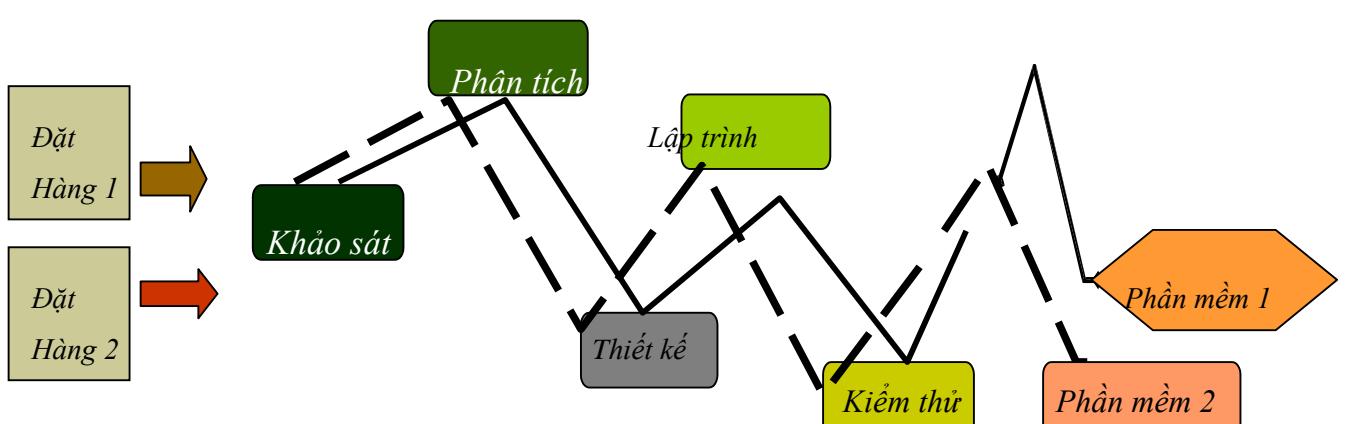
Một phần mềm thường có những đặc trưng riêng so với những sản phẩm khác như sau:

- ✓ Không mòn cũ, nhưng thoái hóa theo thời gian. Khi môi trường sử dụng thay đổi dẫn đến các nhu cầu đổi mới với phần mềm cũng thay đổi. Nếu phần mềm không được tiến

hóa, nó sẽ không được sử dụng nữa. Hơn nữa ngày càng nhiều lỗi phát sinh khi ta nâng cấp phần mềm, dẫn đến chi phí tiến hóa phần mềm ngày càng lớn.



- ✓ Phần mềm không được lắp ráp từ những mảnh có sẵn:
 - Không có danh mục chi tiết cho trước
 - Sản phẩm đặt hàng theo từng yêu cầu riêng lẻ



- ✓ Bản chất của phần mềm là phức tạp, khó hiểu, vô hình.
 - Phần mềm là hệ thống logic khó hiểu. Chúng bao gồm nhiều khái niệm khác nhau, chứa những mối liên kết logic không thể thấy được. Để hiểu được chúng ta cần phải tư duy trừu tượng
 - Phần mềm không thể nhìn thấy được. Nó không phải là vật thể vật lý. Mỗi biểu diễn của nó chỉ một khía cạnh (dữ liệu, hành vi, cấu trúc, giao diện) mà không phải là một hệ thống tổng thể.
- ✓ Thay đổi là bản chất
 - Phần mềm là mô hình thế giới thực, do đó nó luôn thay đổi theo thời gian. Một trong số các lý do do đó là khi môi trường nghiệp vụ thay đổi, nhu cầu con người thay đổi thì phần mềm phải thay đổi để đáp ứng những nhu cầu này
 - Phần mềm cũng phải thay đổi để thích ứng với môi trường vận hành nó (như các hệ phần mềm nền – hệ điều hành thay đổi, phần cứng thay đổi, ...)

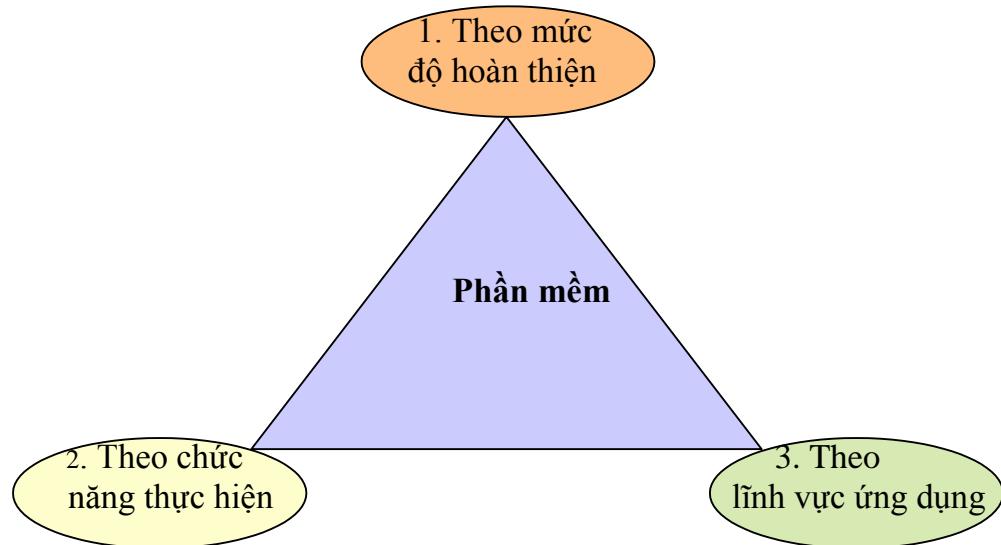
- ✓ Phần mềm cần được phát triển theo nhóm do quy mô của phần mềm ngày càng lớn, phải chuyển giao nhanh chóng phần mềm đến người dùng. Để phát triển phần mềm đòi hỏi cần phải có các kỹ năng khác nhau. Tuy nhiên cần lưu ý ở đây đó là năng xuất nhóm không tỉ lệ thuận với số thành viên trong nhóm (1 người giỏi > 5 người trung bình).

Một số vấn đề nảy sinh khi làm việc theo nhóm chúng ta cũng cần quan tâm như:

- Trao đổi thông tin lớn (10000 email/ngày)
- Khó kiểm soát được tính đồng bộ sản phẩm
- Khó tăng tốc độ phát triển phần mềm bằng cách thêm người. Cá nhân ảnh hưởng lớn đến kết quả của cả nhóm.

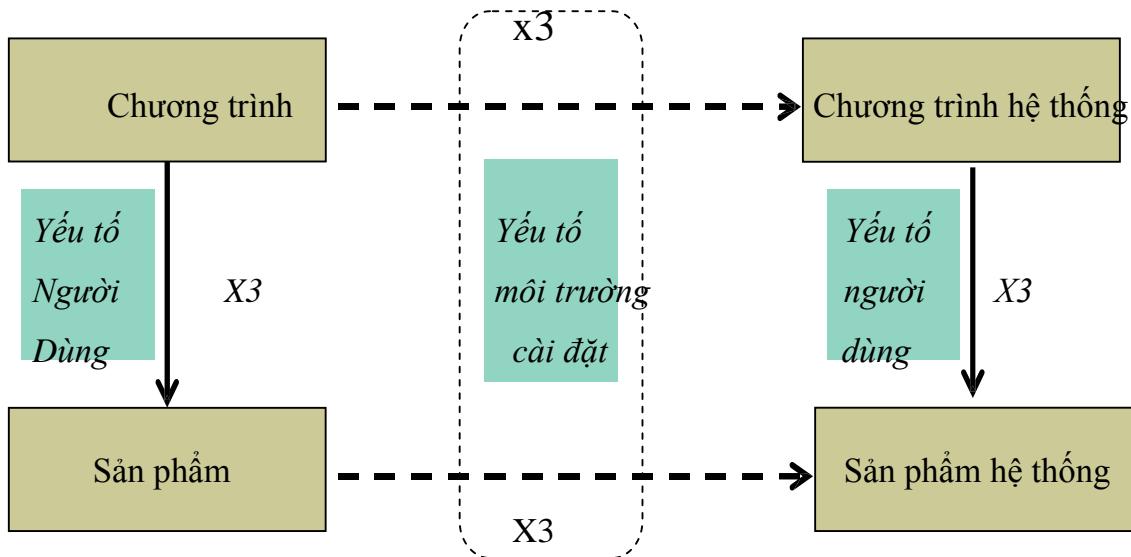
1.1.4 Phân loại phần mềm

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



a. 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.

b. Phân loại phần mềm 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

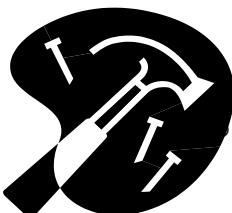
thông tin, hệ thống quản lý,). Chúng có đặc trưng đơn chiết, được sản xuất theo các yêu cầu đặc thù riêng (dễ nhận dạng).

- 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 hàng 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,

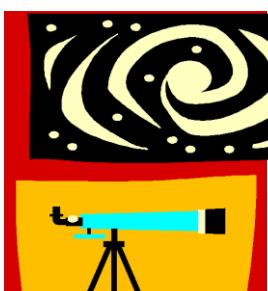
c. Phân loại theo lĩnh vực ứng dụng

- ✓ Phần mềm hệ thống (System Software)



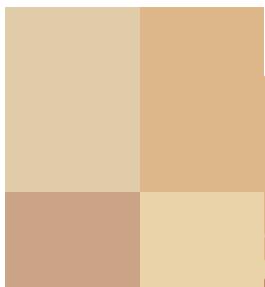
- Phục vụ cho các chương trình khác
- Tương tác trực tiếp với phần cứng, ví dụ hệ điều hành,
- Phục vụ nhiều người dùng.

- ✓ Phần mềm thời gian thực (real time System)



Nhằm thu thập, xử lý (phân tích) các dữ kiện thế giới, kiểm soát, điều khiển và điều phối các dữ kiện này. Chúng thường phải đáp ứng các yêu cầu chặt chẽ về mặt thời gian.

✓ Phần mềm nghiệp vụ (Business software)



Xử lý các thông tin nghiệp vụ, chúng thường được gắn với các cơ sở dữ liệu. Chúng cũng nhằm xử lý các giao tác (ví dụ mạng bán hàng,). Lĩnh vực ứng dụng của loại phần mềm này là rất lớn (ví dụ hệ điều khiển vũ trụ,).

✓ Phần mềm khoa học kỹ thuật (Scientific software):



Dùng các thuật toán phức tạp (vật lý, mô phỏng). Năng lực tính toán cao

✓ Phần mềm nhúng (Embeded software)



- Những phần mềm này chỉ thực thi/đọc ra khi thiết bị khởi động.
- Chúng thường thực hiện các chức năng tương đối hạn chế (nhằm điều khiển sản phẩm).
- Chúng là sự kết hợp giữa hệ thống và thời gian thực.

✓ Phần mềm máy tính cá nhân



- Thường giải quyết các bài toán nghiệp vụ nhỏ. Chúng phục vụ chủ yếu cho các hoạt động học tập, vui chơi, giải trí.
- Giao diện đồ họa phát triển
- Chúng thường có nhu cầu rất cao trong thực tế

✓ Phần mềm trí tuệ nhân tạo (Intelligent software)\

- Thường sử dụng các thuật toán phi số (logic) như suy luận, tìm kiếm...
- Các phần mềm thuộc loại này như các hệ chuyên gia, phần mềm nhận dạng, các phần mềm trò chơi như chơi cờ,

- ✓ Phần mềm dựa trên nền Web (Web – based Software)



- Những phần mềm này thường cung cấp các dịch vụ khai thác thông tin
- Chương trình khai thác là chung (trình duyệt)

1.1.5 Tiến hóa phần mềm và những thách thức đặt ra

Phần mềm luôn tiến hóa không ngừng cùng với những tiến bộ của phần cứng máy tính như về quy mô, sự phức tạp, về tốc độ, về chức năng và mức độ hoàn thiện, ... Công nghệ phần cứng là không ngừng phát triển, do đó nhu cầu phần mềm ngày càng tăng. Ngày càng nảy sinh nhiều khó khăn, thách thức cần giải quyết.

a. Các giai đoạn tiến hóa phần mềm

* *Giai đoạn 1: từ 1950 -> 1960:*

- ✓ Giai đoạn này các chương trình nhỏ, tính toán chuyên dụng. Các hoạt động xử lý của chương trình là xử lý 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), ..

* *Giai đoạn 2: -> giữa thập kỷ 70*

- ✓ 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.

* *Giai đoạn 3: -> 1990*

- ✓ Xuất hiện các phần mềm trên máy tính cá nhân, phần mềm trên mạng, các hệ thống phần mềm lớn, chia sẻ được thông tin và các thao tác. Ra đời phần mềm nhúng. Các xử lý gồm xử lý số, ký tự, âm thanh, hình ảnh, theo lô, thời gian thực, phân tán, song sóng. Truy cập dữ liệu phát triển, cả từ xa.
- ✓ Ngôn ngữ: Bậc cao, hướng đối tượng, logic, ..
- ✓ Các tiêu chí đánh giá phần mềm gồm: Tính tiện dụng, tính tin cậy và tính dễ bảo trì.
- ✓ Công nghệ: mạch tích hợp lớn, vi mạch, các cấu hình mạng, internet, CSDL quan hệ.

* *Giai đoạn 4: từ năm 1990 đến nay*

- ✓ 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 càng được mở rộng.
- ✓ Cơ sở dữ liệu hướng đối tượng, kho dữ liệu phát triển.
- ✓ Ngôn ngữ: Hướng đối tượng, thẻ hệ 4, lập trình trực quan.
- ✓ Các tiêu chí đánh giá phần mềm gồm: Tính tiện dụng, độ tinh vi, tính tin cậy, tính dễ bảo trì.
- ✓ Công nghệ: vi mạch siêu tích hợp, internet, mạng không dây tốc độ cao, hướng đối tượng, Web.

* *Tiêu chí phần mềm tốt hiện nay*

Một phần mềm được xem là tốt có thể được nhìn nhận từ hai phía:

1. Phía người dùng gồm các tiêu chí:

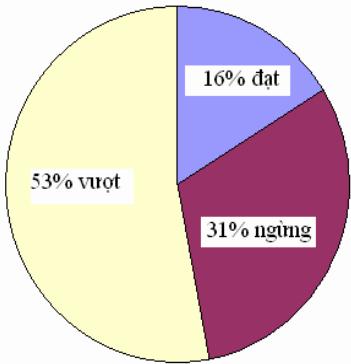
- Đầy đủ các chức năng nghiệp vụ
- Dễ sử dụng, tinh vi (tính thông minh)
- Tin cậy, an toàn

2. Phía nhà phát triển: tính dễ bảo trì.

b. Khó khăn và thách thức

* **Thực trạng các sản phẩm phần mềm và thực trạng phát triển.**

Theo con số thống kê từ [Standish Group. The CHAOS Report, 1995],
[\[http://www.pm2go.com/sampleresearch/chaos\]](http://www.pm2go.com/sampleresearch/chaos) cho thấy:



- 16% dự án đúng lịch, trong ngân sách, sản phẩm chất lượng
- 31 % dự án bị ngừng
- 53% dự án vượt ngân sách, quá hạn, ít tính năng.
- Năm 1995, Mỹ chi 81 tỷ \$ cho các dự án bị hủy, 59 tỷ \$ chi thêm cho các dự án sai kế hoạch.

Công nghiệp phần mềm trở thành ngành khổng lồ:

- Chi phí phát triển OS 360 (năm 1963 – 1966) là 200 triệu \$
- Chi phí cho phần mềm (2000): 770 tỉ \$, tăng 12%/năm.

Năng xuất lập trình vẫn thấp:

- Phát triển mang tính thủ công, giá thành cao
- Vấn đề chất lượng phần mềm trở thành trọng tâm

* Bản chất của vấn đề

- ✓ Phần mềm bản chất là phức tạp
- ✓ Yêu cầu ngày càng tăng về mặt số lượng, quy mô, sự tiện ích. Nhu cầu phần mềm tăng gần 20%/năm, Window 2k gần 100 triệu dòng mã lệnh, ...
- ✓ Sự tiến bộ nhanh phần mềm và phần cứng. Cơ sở hạ tầng và môi trường vận hành thay đổi. Ví dụ hệ điều hành Window từ 95, 98, 2000, 2003, ...Năng lực máy tính tăng gấp 2 lần sau 18 tháng và hơn nữa

* Những lý do chính

- ✓ 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

* 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.Kỹ nghệ phần mềm (Software Engineering)

1.2.1 Lý do ra đời

Phần mềm được viết ngay từ khi xuất hiện các hệ máy tính và ngôn ngữ lập trình đầu tiên nhằm khai thác thế mạnh của phần cứng. Công nghệ phần cứng từ khi ra đời cho đến nay phát triển nhanh và mạnh. Tính năng và tiềm lực của nó ngày càng tăng . Do đó, các ứng dụng trên nó 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 dưới dạng các sản phẩm phần mềm ngày càng tăng. Mọi ngành nghề kinh tế, mọi hoạt động xã hội và đời sống đều cần đến các phần mềm hỗ trợ. Dẫn đến các sản phẩm phần mềm ngày càng lớn và phức tạp.

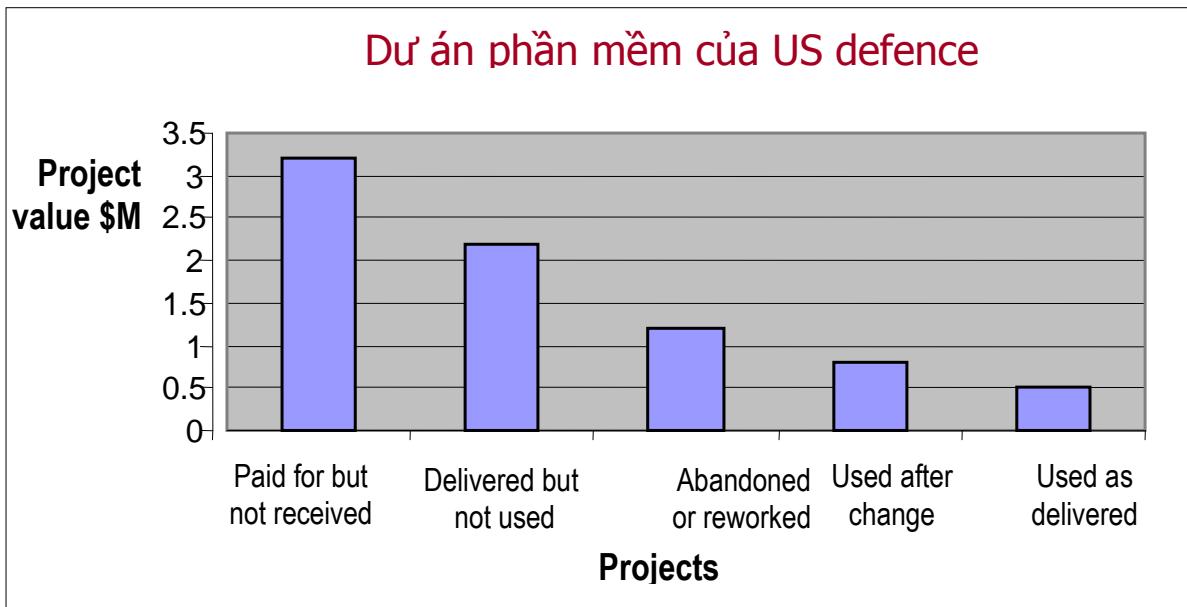
Thực tế các sản phẩm phần mềm không đáp ứng kịp các yêu cầu của người sử dụng. Việc sản xuất phần mềm gặp thất bại quá nhiều. Hầu hết các sản phẩm phần mềm đều rơi vào tình trạng:

- + 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
- Trung bình thời gian thực hiện thực tế bị kéo dài 50% (cách biệt lên tới 200 – 300%)
- Các đề án lớn dễ bị thất bại
- 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

Chi phí chi trả cho việc phát triển các sản phẩm phần mềm chiếm phần lớn tổng chi phí bình quân (GNP) cho các hoạt động ngành nghề kinh tế khác của đất nước, nhưng hiệu quả ứng dụng của p/m mang lại lại không cao. Theo con số thống kê của mỹ về các dự án phần mềm của bộ quốc phòng năm 1970 cho thấy:



⇒ Khủng hoảng phần mềm!

Tại Hội nghị TG bàn về khủng hoảng phần mềm diễn ra năm 1968, người ta đã xác định các nguyên nhân gây ra khủng hoảng phần mềm. Nguyên nhân chính ở dẫn đến tình trạng khủng hoảng đó là việc sản xuất các sản phẩm phần mềm vẫn 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

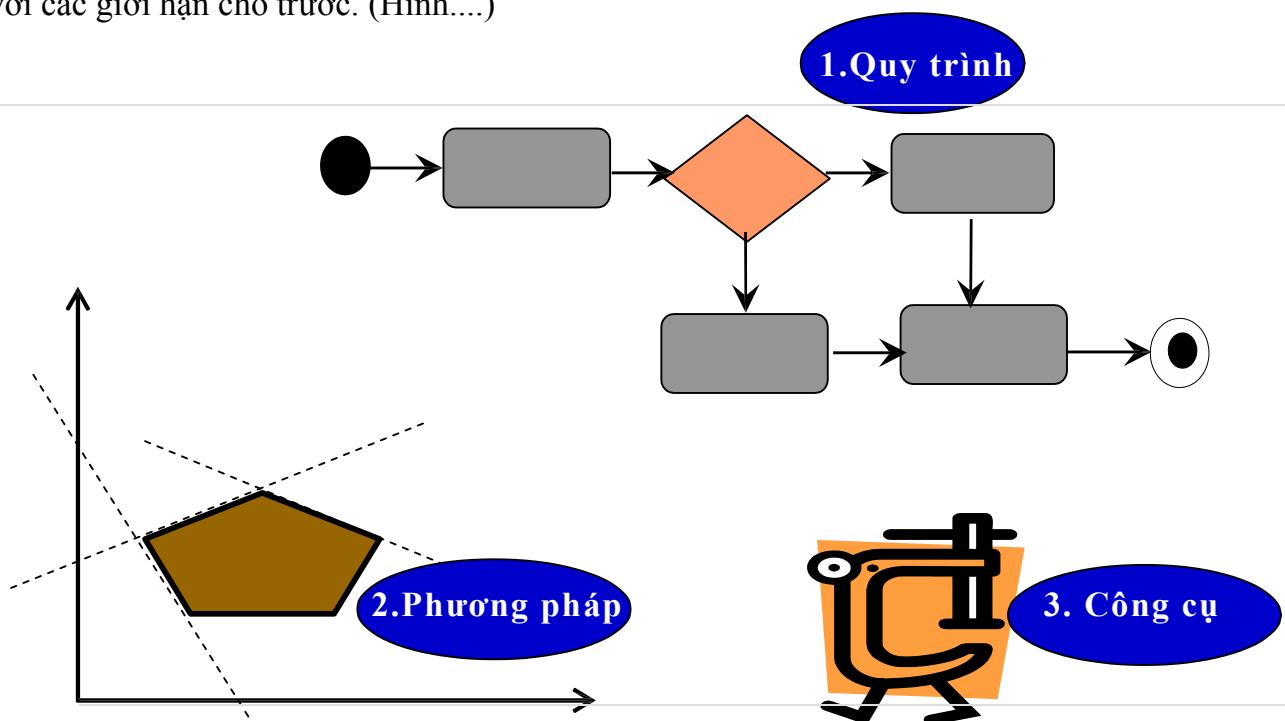
Để giải khắc phục các nguyên nhân trên, chúng ta cần xây dựng phần mềm theo công nghệ ~ công nghiệp hóa quá trình sản xuất phần mềm. Khái niệm công nghệ phần mềm được đưa ra, và từ đó công nghệ phần mềm thực sự trở thành một ngành nghiên cứu không thể thiếu được trong lĩnh vực CNTT, nhằm khắc phục những khủng hoảng trên. 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 để nhằm đến mục tiêu sản xuất phần mềm độc lập, đúng hạn, phù hợp kinh phí và đáp ứng mọi yêu cầu người sử dụng.

1.2.2 Các định nghĩa về Kỹ nghệ phần mềm

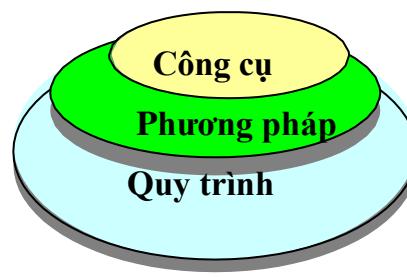
Sau đây làm một số định nghĩa về kỹ nghệ/công nghệ phần mềm

- ✓ **Theo Bauer [1969]:** SE là thiết lập và sử dụng các nguyên lý công nghệ đúng đắn để được phần mềm một cách kinh tế, vừa tin cậy vừa làm việc hiệu quả trên các máy tính.
- ✓ **Theo Sommerville [1997]:** SE là nguyên lý kỹ nghệ liên quan đến tất cả các mặt *lý thuyết, phương pháp và công cụ* của phần mềm.
- ✓ **Theo Pressman [1995]:** SE là bộ môn tích hợp cả quy trình, các phương pháp và các công cụ để phát triển phần mềm máy tính.

Từ các định nghĩa trên chúng ta thấy, kỹ nghệ phần mềm là một quá trình kỹ nghệ tích hợp của các quy trình/thủ tục, các phương pháp và các công cụ nhằm tạo ra phần mềm hiệu quả, với các giới hạn cho trước. (Hình....)



Công nghệ phần mềm được xem như 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 (hình)



Trong đó:

- 1. Các thủ tục (procedures):** Đề cập đến các quy trình phát triển và quản lý phần mềm như:
 - ✓ Xác định trình tự các công việc cần thực hiện
 - ✓ Xác định các tài liệu, sản phẩm cần bàn giao và cách thức thực hiện
 - ✓ Xác định các mốc thời gian (milestones) và sản phẩm đưa ra (theo các chuẩn?) có

thể ở mức chung cho nhiều dự án hoặc cho một dự án cụ thể.

2. Các phương pháp (methods):

Phương pháp đề cập đến cách làm cụ thể để xây dựng phần mềm. Mỗi công đoạn phát triển phần mềm có một phương pháp riêng:

- ✓ Phân tích: xác định và đặc tả yêu cầu;
- ✓ Thiết kế: đặc tả kiến trúc, giao diện, dữ liệu và các chức năng;
- ✓ Lập trình: Theo phương pháp có cấu trúc, hoặc hướng đối tượng,
- ✓ Kiểm thử: Hộp đen, hộp trắng, áp lực, hồi quy, luân sợi, ...
- ✓ Quản lý dự án: PERT, GANTT, COMOCO,...

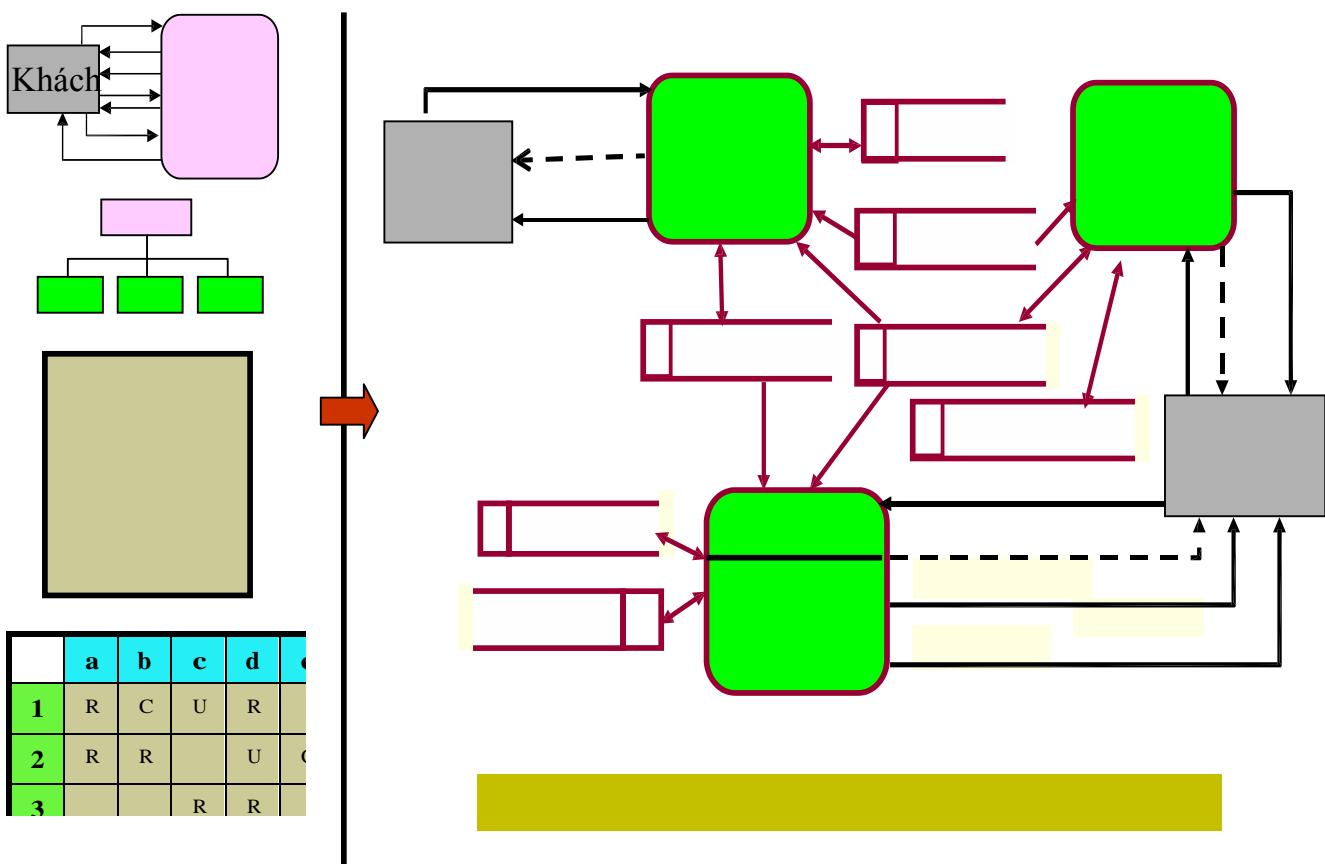
Nội dung của một phương pháp bao gồm:

- ✓ *Các phần tử của mô hình*: Mô tả khái niệm ;
- ✓ *Các ký pháp*: Đặc tả phần tử;
- ✓ *Các quy tắc*: Liên kết các phần tử;
- ✓ *Quy trình xây dựng*: Trình tự tạo ra một mô hình;
- ✓ *Lời khuyên, cách dùng*: Khi nào nên dùng, và dùng phương pháp như thế nào.

Ví dụ: Xét mô hình (phương pháp) luồng dữ liệu

Khái niệm	Ký pháp	Quy tắc	Quy trình
Tác nhân	Tên tác nhân		
Tiến trình	i Tên tiến trình	<ul style="list-style-type: none"> ✓ Tiến trình i là duy nhất. ✓ Các luồng vào 1 tiến trình phải ≠ các luồng ra; ✓ Không có các luồng dữ liệu sau: <ul style="list-style-type: none"> ○ Kho -> kho; ○ Kho -> tác nhân; 	<ul style="list-style-type: none"> ✓ Vẽ luồng dữ liệu mức 0 từ mô hình nghiệp vụ; ✓ Vẽ biểu đồ mức i: từ mỗi tiến trình mức i-1 chưa là cơ sở.
Kho dữ liệu	Tên kho		
Luồng dữ liệu	Tên luồng dữ liệu		

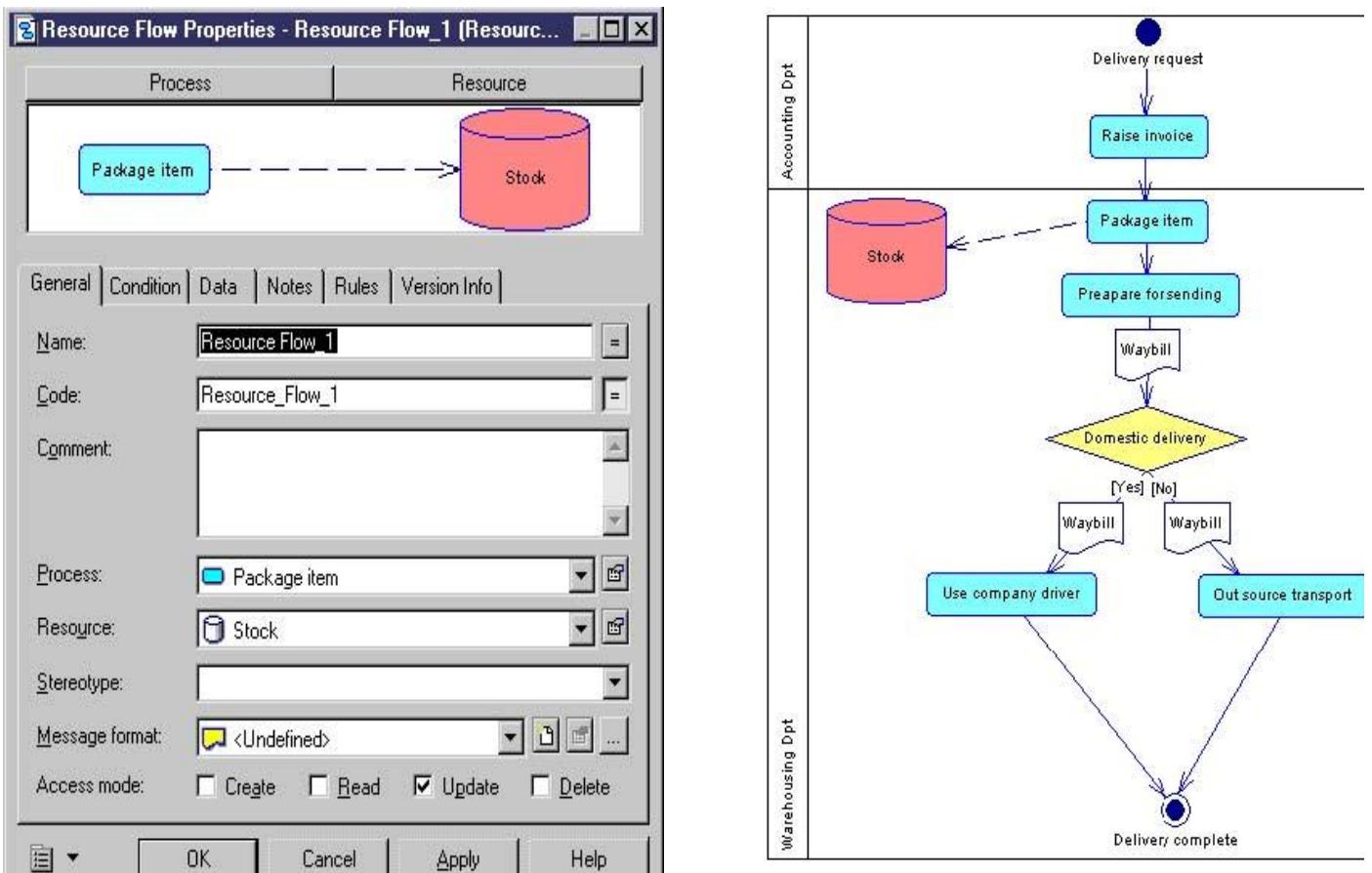
Ví dụ: Biểu đồ luồng dữ liệu mức 0 cho bài toán trông và gửi xe:



3.Các công cụ (Tools)

Là những phần mềm trợ giúp tự động hoặc bán tự động cho các phương pháp. CASE là các công cụ trợ giúp cho các công đoạn khác nhau của tiến trình phát triển phần mềm như các ngôn ngữ lập trình gồm các công cụ:

- ✓ Sinh giao diện: C Builder, ...
- ✓ Hỗ trợ phân tích, thiết kế: Rwin, Modeler, *Oracle Designer*, *Rational Rose*, ...
- ✓ Trợ giúp lập trình: C Compiler, debugger
- ✓ Trợ giúp quản lý: *project management*.



Như vậy, kỹ 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.

1.2.3 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 khôi, 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.

1.3 FQAs về Công nghệ phần mềm

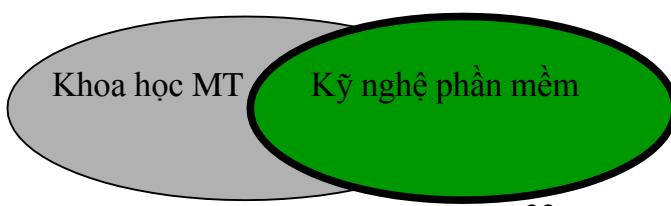
1.3.1 Công nghệ (engineering) là gì?

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(process) nhằm 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.

1.3.2 Sự khác biệt giữa công nghệ phần mềm và khoa học máy tính là gì?

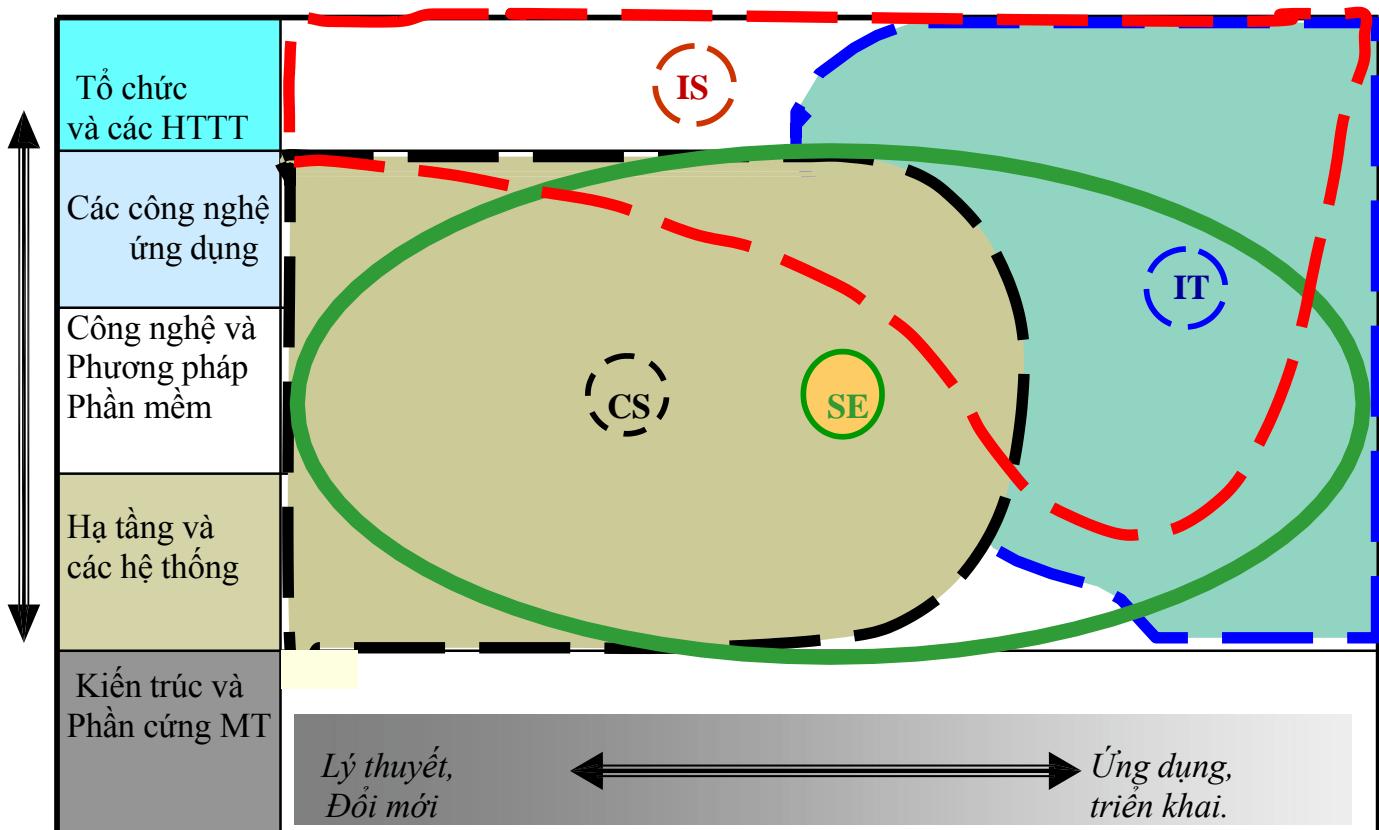
Khoa học máy tính đề cập tới lý thuyết và những vấn đề cơ bản liên quan đến MT; còn công nghệ phần mềm đề cập tới thực tiễn của việc phát triển và đưa ra một phần mềm hữu ích.

Khi sự phát triển của phần mềm trở lên mạnh mẽ thì các lý thuyết của khoa học máy tính không đủ cho các hoạt động của kỹ nghệ phần mềm.



1.3.3. 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 hình[computing curricula 11/2004 -ACM, AIS, IEEE]



Mỗi lĩnh vực tính toán được giới hạn bằng một vùng biên có màu 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.

1.3.4 Kỹ nghệ hệ thống là gì?

Kỹ nghệ hệ thống là toàn bộ công việc phát triển hệ thống dựa trên máy tính (computer based system):

- ✓ Phần cứng
- ✓ Phần mềm
- ✓ Tổ chức, quản lý.



Kỹ nghệ hệ thống bao gồm các hoạt động:

- ✓ Đặc tả

- ✓ Thiết kế,
- ✓ Triển khai;
- ✓ Tích hợp,
- ✓ Thẩm định,
- ✓ Bảo trì.

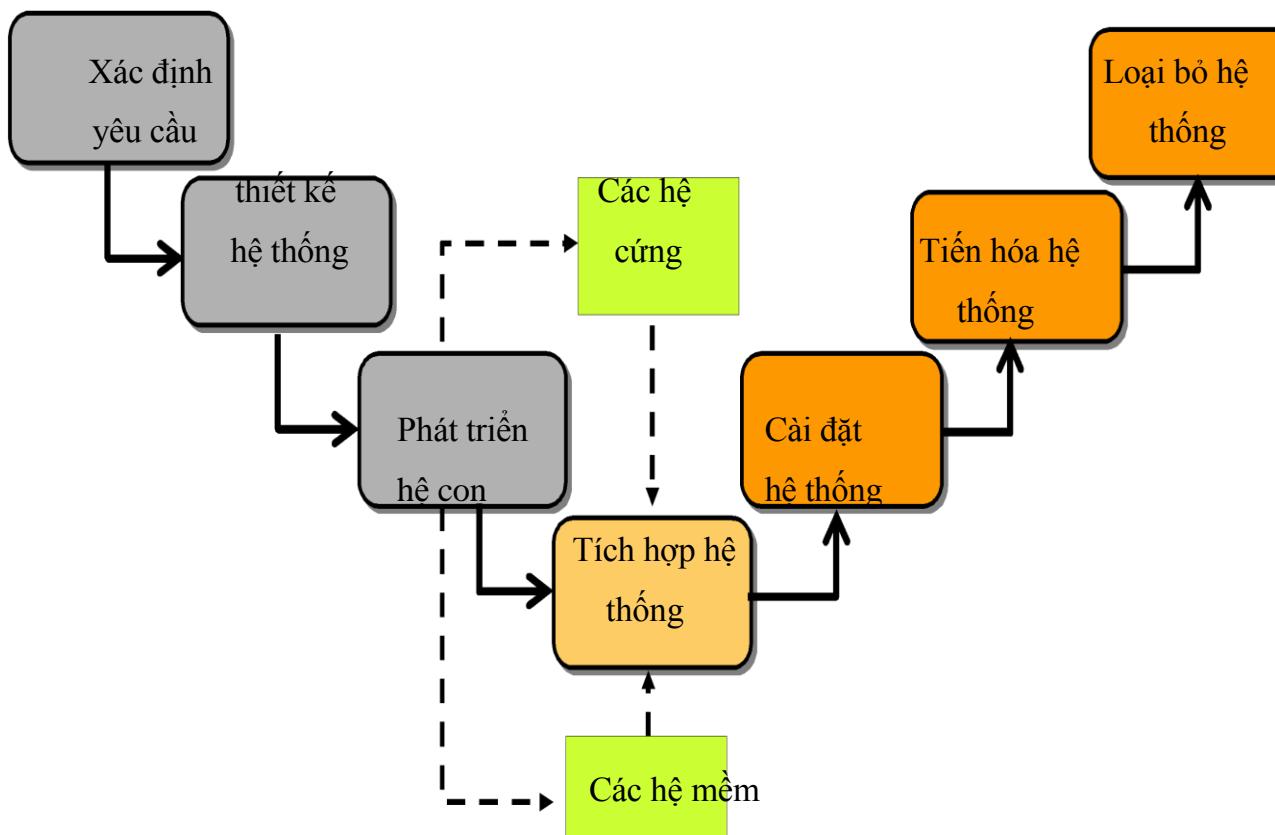
Các đặc tính nổi trội của hệ thống dựa trên máy tính:

Thuộc tính chức năng: Điều khiển một hệ

Thuộc tính phi chức năng: độ tin cậy, an toàn

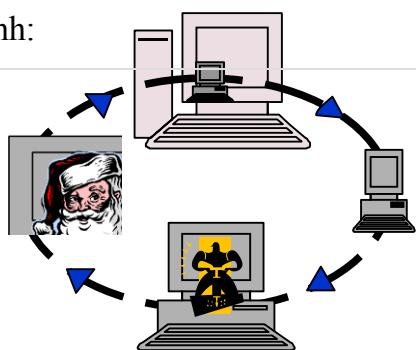
Mà chúng quyết định đến toàn bộ hệ thống, mà từng bộ phận riêng lẻ không thể có được.

Kỹ nghệ hệ thống liên quan đến việc áp dụng các nguyên lý tích hợp nhằm làm giảm chi phí làm lại toàn bộ hệ thống. Tiến trình kỹ nghệ hệ thống được biểu diễn như hình...



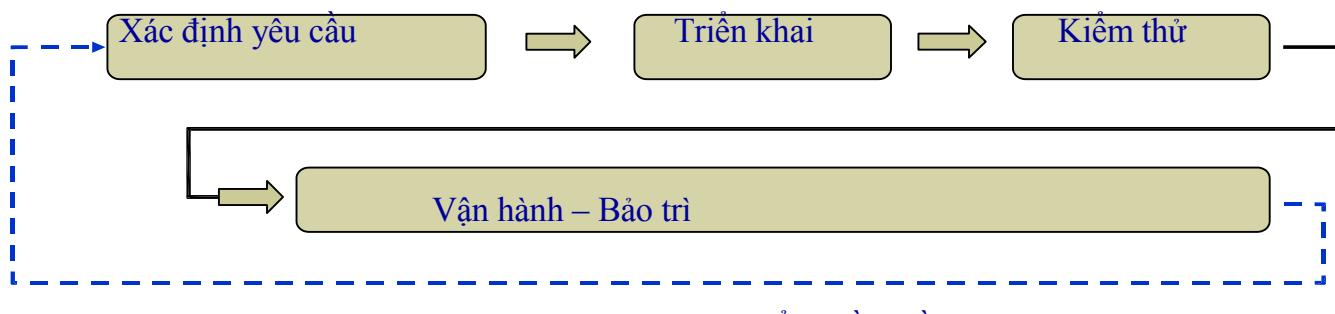
1.3.6. Vòng đời phát triển của một hệ thống phần mềm

Vòng đời phát triển của một hệ thống phần mềm bao gồm các hoạt động từ khi được đặt hàng, phát triển, sử dụng đến khi loại bỏ (nghỉ hưu) nó. Vòng đời được chia thành các giai đoạn chính:



- Xác định yêu cầu,
- Triển khai,
- Kiểm thử,
- Vận hành, bảo trì và lặp lại.

Nội dung, thứ tự và các nội dung của các giai đoạn này khác nhau tùy theo từng sản phẩm và dự án. Tùy theo mô hình áp dụng, việc phân chia các pha, các bước có thể khác nhau giới hạn từ 3 đến 20 bước.



Các pha chính của vòng đời phát triển phần mềm

Các giai đoạn phát triển phần mềm chung nhất gồm:

1.Xác định yêu cầu: ✓ Phân tích hệ thống ✓ Lập kế hoạch ✓ Phân tích yêu cầu ✓ Đặc tả yêu cầu	2.Phát triển ✓ Thiết kế ✓ Mã hóa ✓ Kiểm thử, ✓ Làm tài liệu	
3.Tiến hóa (vận hành, bảo trì) ✓ Sửa lỗi ✓ Thích nghi ✓ Nâng cao, ✓ Bổ sung.		

1.3.7 Tiến trình phần mềm là gì?

Tiến trình phần mềm là một tập hợp các hoạt động có cấu trúc nhằm 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:

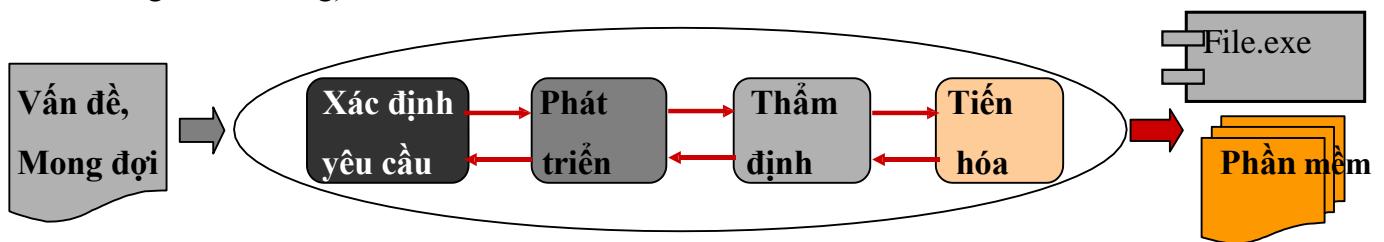
Gắn với mỗi dự án

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

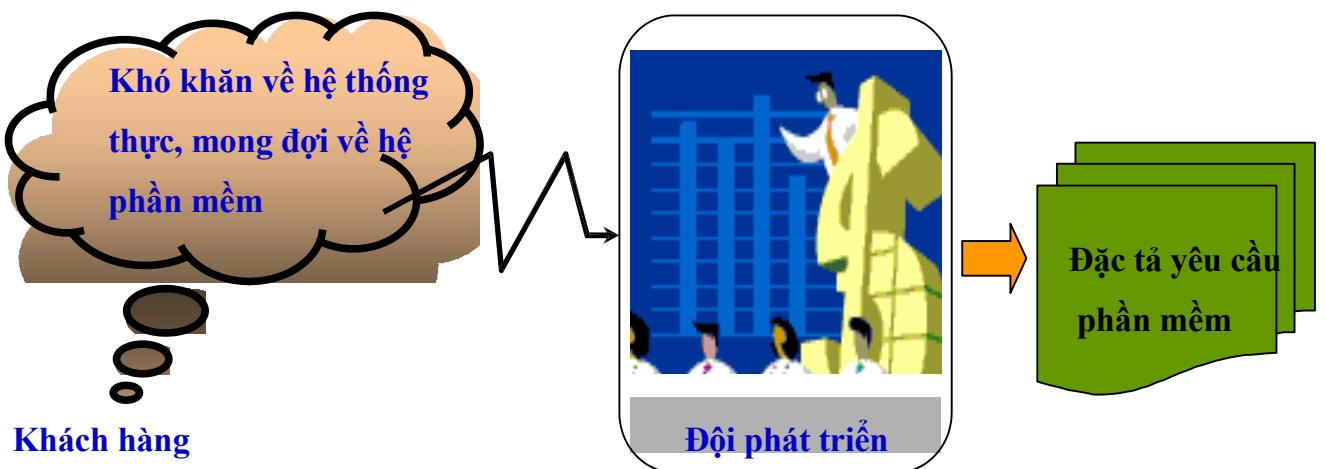
Sản phẩm cuối cùng là phần mềm bàn giao.

Các hoạt động chính của mọi tiến trình phần mềm:

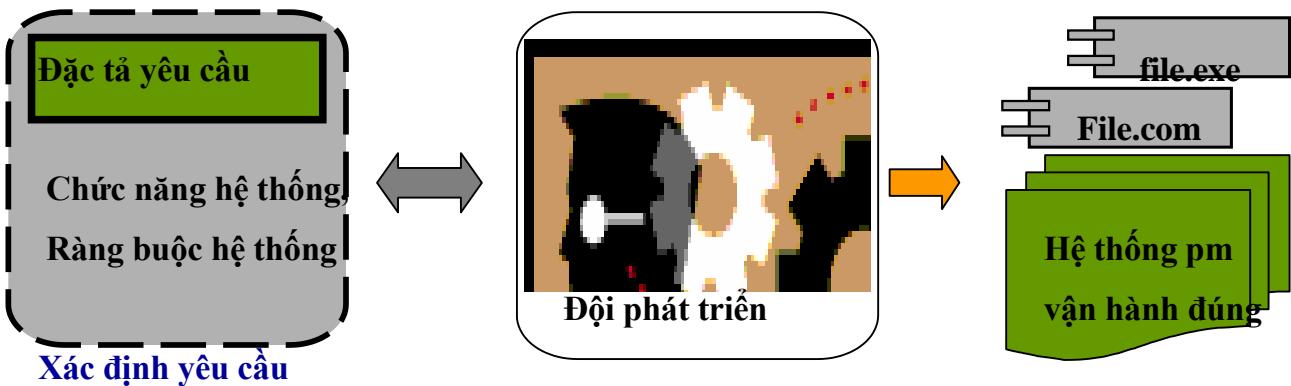
- ✓ *Xác định yêu cầu*: Xác định rõ các yêu cầu sản phẩm
- ✓ *Phát triển*: Tạo ra sản phẩm
- ✓ *Thẩm định*: Phần mềm có đáp ứng được các yêu cầu không
- ✓ *Tiến hóa phần mềm*: Thay đổi phần mềm nhằm đáp ứng các yêu cầu thay đổi (người dùng, môi trường).



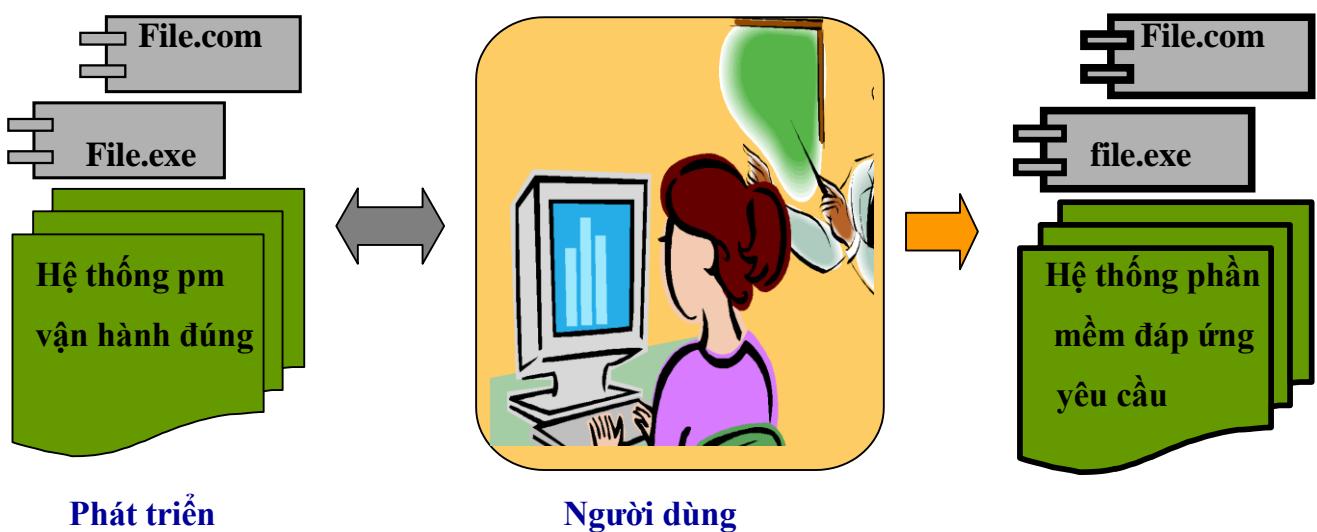
a. Xác định yêu cầu (Requirements)



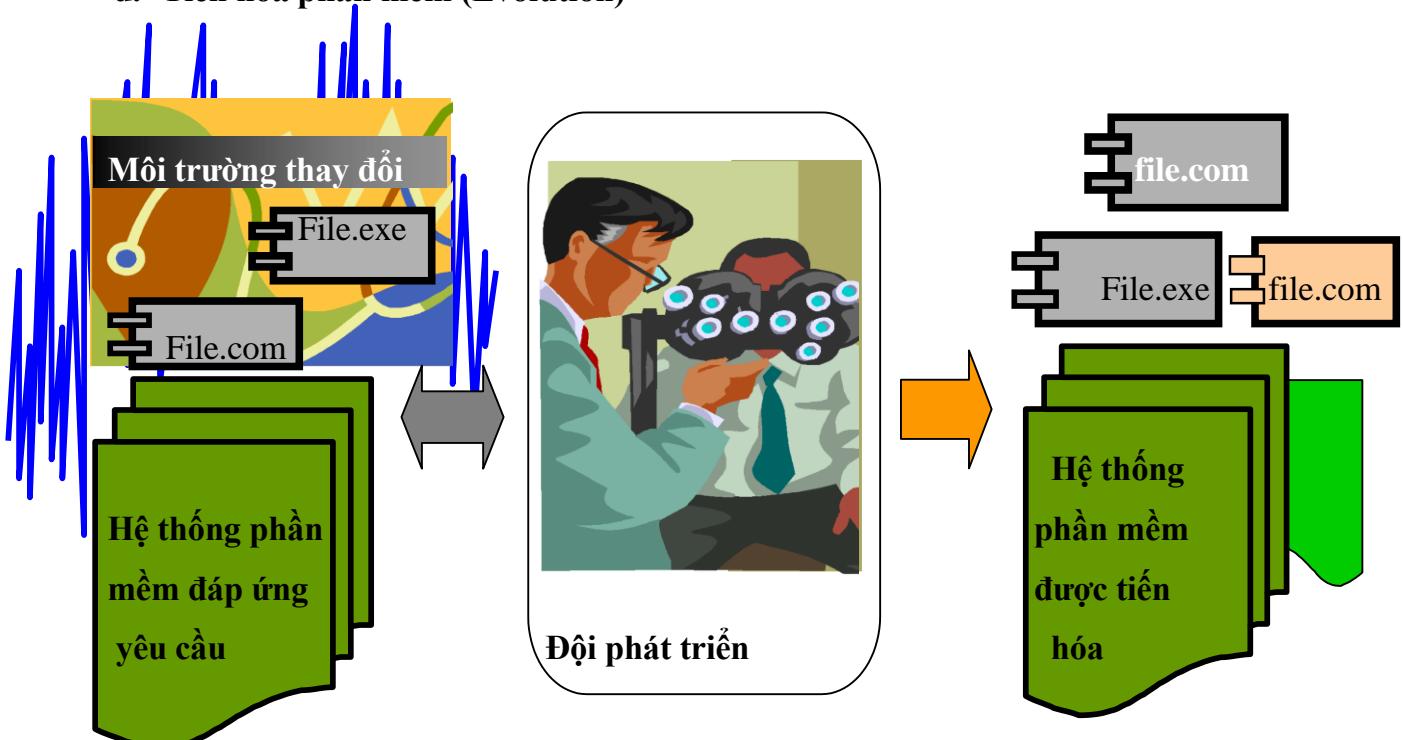
b. Phát triển (Development)



c. Thẩm định phần mềm (Validation)



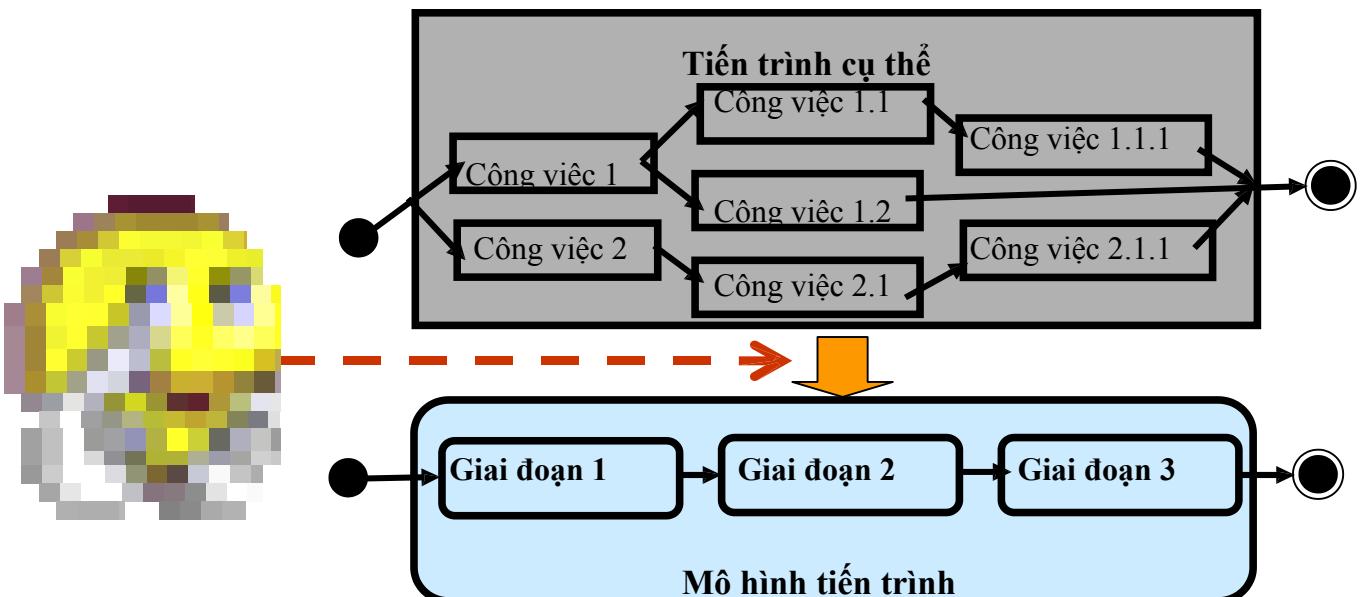
d. Tiến hóa phần mềm (Evolution)



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. Do đó, ta cần sử dụng quy trình phát triển hệ thống thích hợp cho dự án, phần mềm xác định.

1.3.8. Mô hình quy trình phần mềm là gì?

Mô hình quy trình phát triển phần mềm là một các biểu diễn trừu tượng tiến trình phần mềm theo cách nhìn cụ thể.



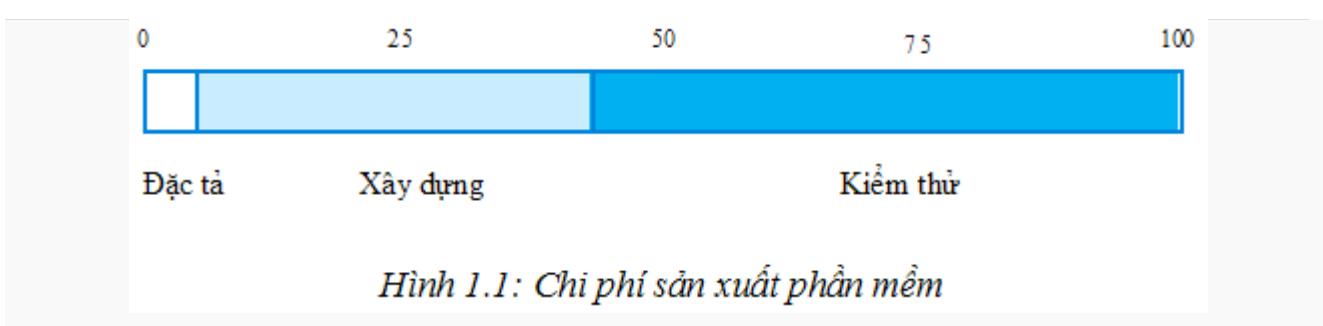
1.3.9. Chi phí của kỹ nghệ phần mềm là gì?

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:

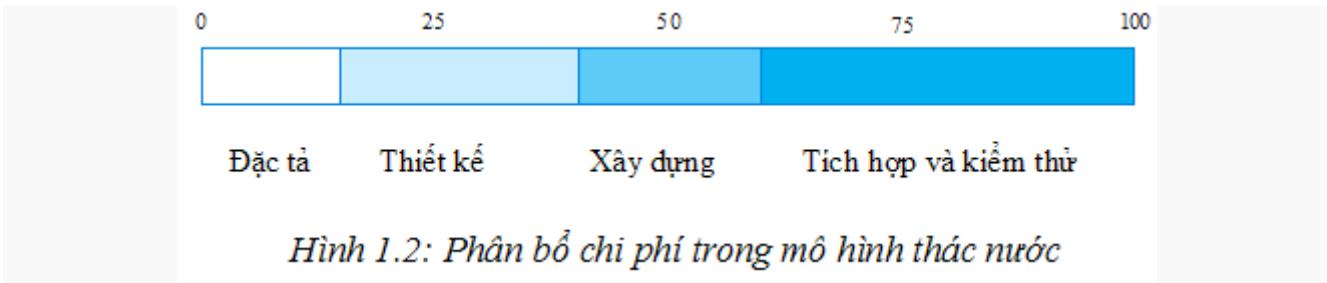
- ✓ Loại hệ thống (là đơn giản hay phức tạp);
- ✓ Yêu cầu đặt ra (nhiều, ít, cao, thấp);
- ✓ Mức độ hoàn thiện (hiệu năng, độ tin cậy, an toàn, ...);
- ✓ Năng lực của tổ chức (nhân lực, công cụ, công nghệ, kỹ năng có được,);
- ✓ Loại tiến trình sử dụng.

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



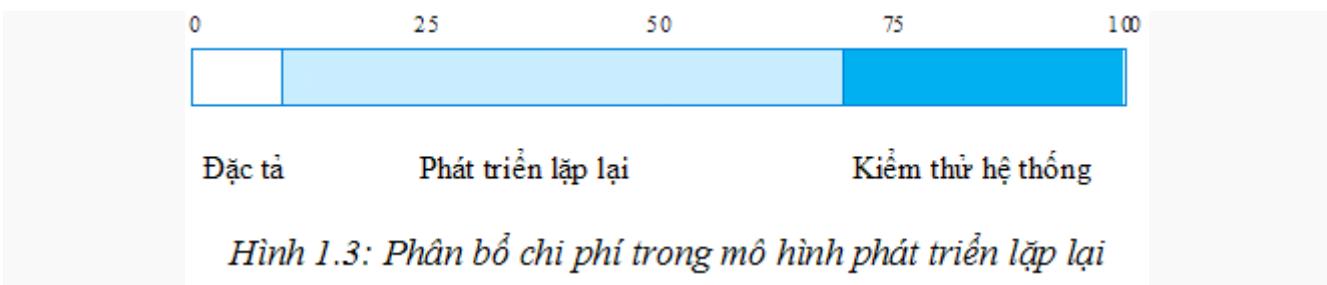
Việc phân bổ chi phí cũng phụ thuộc vào mô hình phát triển hệ thống được sử dụng. Sau đây là bảng so sánh chi phí của 3 mô hình phổ biến nhất, thường được sử dụng:

- ✓ 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.



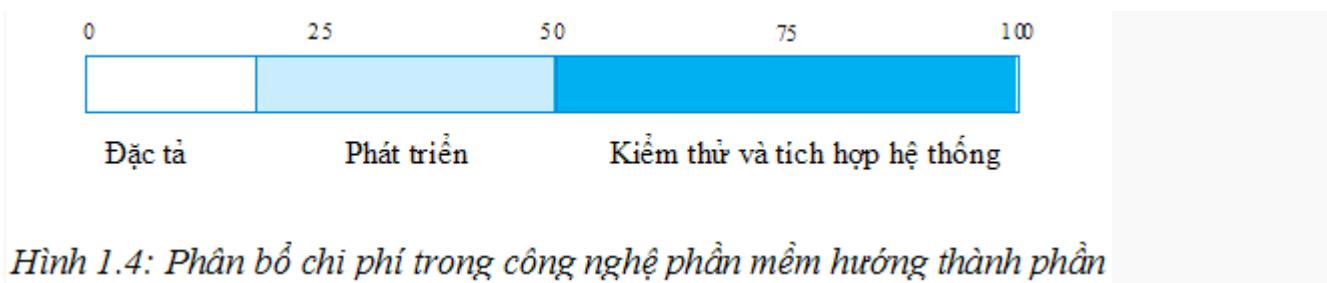
Hình 1.2: Phân bổ chi phí trong mô hình thác nước

- ✓ 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.



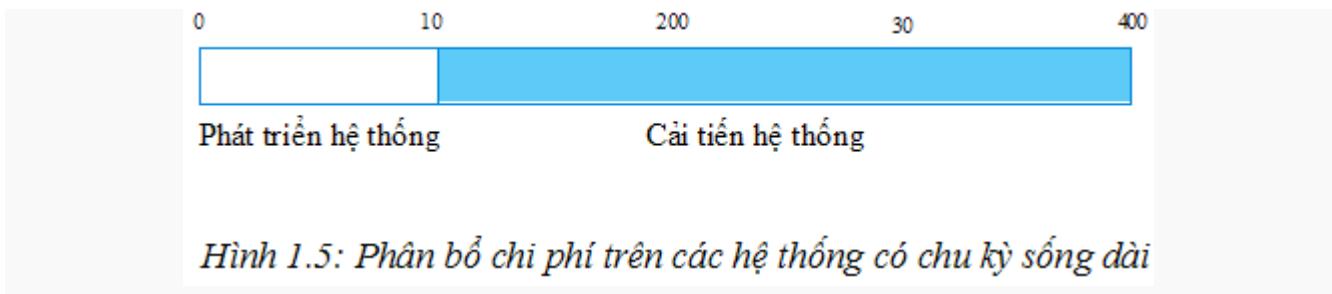
Hình 1.3: Phân bổ chi phí trong mô hình phát triển lặp lại

- ✓ 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.



Quy mô phần mềm:

Kích cỡ phần mềm ảnh hưởng lớn đến chi phí phần mềm. Kích cỡ thường đo bằng số dòng lệnh của chương trình và nó phụ thuộc vào bài toán cần giải quyết cũng như kết quả thiết kế và ngôn ngữ lập trình lựa chọn. Ngoài ra còn phụ thuộc vào trình độ của người lập trình. Với cùng bản thiết kế, dùng ngôn ngữ khác nhau sẽ cho ra chương trình với kích cỡ khác nhau. Sau đây là một số ví dụ về số đo ngôn ngữ:

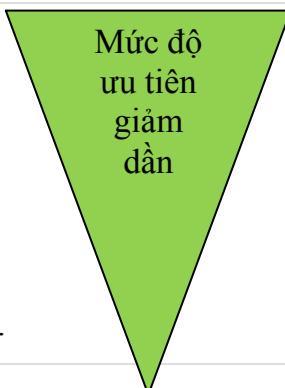
Ngôn ngữ	Dòng lệnh/chức năng
Assembly	320
C FORTRAN 77	128
COBOL 85	105
Ada 83	91
C++ Ada 95	71
Java	56
Visual Basic	55
SQL	35
	?

Mỗi ngôn ngữ lập trình có một năng lực thể hiện nhất định. Chọn ngôn ngữ lập trình tốt, phù hợp sẽ góp phần giảm kích cỡ chương trình.

1.3.10. Phần mềm kỹ nghệ tốt là gì

Phần mềm kỹ nghệ tốt được xét theo quan điểm nhà phát triển cần có các thuộc tính sau:

- ✓ Tính bảo trì được.
- ✓ Tính đáng tin cậy.
- ✓ Tính hiệu quả.
- ✓ Tính tiện dụng.
- ✓ Giá cả phải chăng.
- ✓ Một số tiêu chí khác.



a. Tính bảo trì được

“*Phần mềm luôn có yêu cầu sửa đổi*”. Để sửa đổi (bảo trì) được, phần mềm cần:

- ✓ Có kiến trúc tốt: Tính kết dính chặt, ghép nối lỏng lẻo. Tạo thuận lợi cho hoạt động bảo trì như dễ đọc, dễ sửa, dễ phát triển, chỉ ảnh hưởng cục bộ khi sửa đổi
- ✓ Cài đặt bằng ngôn ngữ cấp cao. Điều này làm cho chương trình dễ đọc, dễ hiểu viết mã nhanh.
- ✓ Có tài liệu đầy đủ và tốt. Điều này giúp nhà phát triển dễ theo dõi, dễ hiểu, có cơ sở sửa chữa,

Phần mềm có tuổi thọ càng cao, nó phục vụ được càng nhiều. Chi phí bảo trì thấp sẽ tạo nên tính hiệu quả của chương trình.

b. Tính đáng tin cậy

Phần mềm được gọi là đáng tin cậy nếu:

- ✓ Nó có ít khiếm khuyết về:
 - Lỗi lập trình,
 - Lỗi phân tích,
 - Lỗi thiết kế sai,
 - Thiếu chức năng,
 - Hoạt động không hiệu quả.
- ✓ Đáp ứng được nhu cầu của người sử dụng:
 - Đảm bảo các chức năng cần có,
 - Ôn định, thời gian làm việc không gây ra các lỗi lớn,
 - Cho ra kết quả chính xác.

c. Tính hiệu quả

Tính hiệu quả của phần mềm thể hiện ở chỗ nó không sử dụng lãng phí tài nguyên phần cứng. Không đòi hỏi bộ nhớ lớn, không đòi hỏi CPU tốc độ cao, không chiếm không gian đĩa lớn. Tất cả các tiêu chí này cần tối ưu một cách hợp lý. Để tối ưu, chúng ta cần:

- ✓ Dùng ngôn ngữ bậc thấp
- ✓ Truy cập trực tiếp tới thiết bị
- ✓ Sử dụng thuật toán tốt

Điều này dễ dẫn tới khó bảo trì, giá thành phần mềm cao.

d. Tính tiện dụng, tinh vi

Tính tiện dụng, tinh vi của phần mềm thể hiện ở những tiêu chí sau:

Giao diện phù hợp với trình độ người dùng.

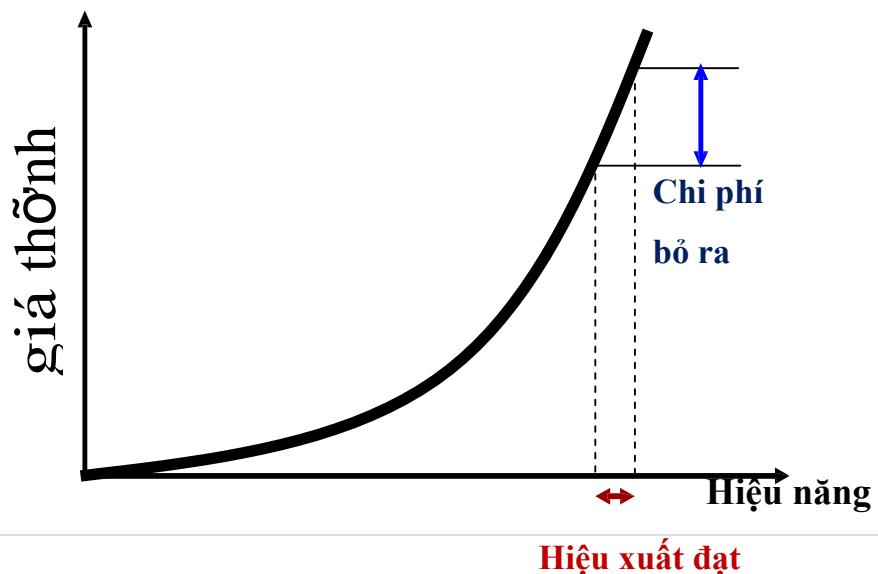
Học nhanh, nhớ lâu, dễ thao tác,

Có đủ tài liệu, nhiều tiện ích, trợ giúp thông minh.

Điều lưu ý ở đây khi thiết kế giao diện đó là giao diện phần mềm quyết định sự thành công của một sản phẩm.

e. Giá cả phải chăng

Để có được phần mềm với giá cả hợp lý, chúng ta cần cân đối, thỏa hiệp giữa các yêu cầu sau: Giá cả, hiệu quả >>><<< dễ bảo trì, dễ sử dụng, tinh vi. Chi phí cho tối ưu không phải là hàm tuyến tính mà là hàm mũ



1.3.11. CASE (Computer-Aided Software Engineering) tools là gì?

CASE là các hệ thống phần mềm được phát triển với mục đích hỗ trợ tự động cho các hoạt động kỹ nghệ phần mềm. Một số sản phẩm CASE điển hình như:

- ✓ Các bộ soạn thảo đồ họa giúp cho việc xây dựng các mô hình hệ thống;

- ✓ Từ điển dữ liệu để quản lý các thực thể thiết kế;
 - ✓ Bộ biên dịch UI đồ họa để xây dựng giao diện người dùng,
 - ✓ Bộ gỡ rối hỗ trợ chương trình tìm lỗi,
 - ✓ Bộ chuyển đổi tự động để sinh ra các phiên bản mới của chương trình,
 - ✓
-

1.3.12. Những thách thức chính đối với công nghệ phần mềm là gì?

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

- Không đồng nhất: phát triển các kỹ thuật xây dựng phần mềm để giải quyết sự không đồng nhất về môi trường thực hiện và nền tảng hạ tầng.
- Chuyển giao: phát triển các kỹ thuật nhằm dẫn tới việc chuyển giao phần mềm tới người sử dụng nhanh hơn.
- Độ tin cậy: phát triển các kỹ thuật để chứng minh rằng phần mềm được người sử dụng nó tin tưởng.

1.4 Các trách nhiệm đạo đức và nghề nghiệp ([6])

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.

1.4.1 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 chơi game trên MT người dùng đến những hành động nghiêm trọng như reo rắc virus trên MT của người khác.

1.4.2 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. Public

2. Client and employer

3. Product

4. Judgment

5. Management

6. Profession

7. Colleagues

8. Self

Bạn đọc có thể tham khảo tập chuẩn mực đạo đức này ở dạng dài và ngắn trong cuốn tài liệu [1].

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

1.5.1 Nhân tố con người trong ngành công nghệ phần mềm

Đối với một sản phẩm 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. Do đặc điểm nghề nghiệp, người kỹ sư phần mềm phải có những kỹ năng cơ bản như:

- Xác định, 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.

để phát triển, đưa vào sử dụng, bảo trì, và thay thế phần mềm. Hoạt động phát triển

(development) được tính bắt đầu từ khi có quyết định xây dựng sản phẩm phần mềm và kết thúc khi sản phẩm phần mềm được chuyển giao cho người sử dụng. Hoạt động sử dụng (operations) là hoạt động xử lý, vận hành hàng ngày sản phẩm phần mềm. Hoạt động bảo trì (maintenance) thực hiện những thay đổi mang tính logic đối với hệ thống và chương trình để chữa những lỗi cố định, cung cấp những thay đổi về công việc, hoặc làm cho phần mềm được hiệu quả hơn. Hoạt động loại bỏ - thay thế (retirement) phần mềm là việc thay thế các ứng dụng hiện thời bởi các ứng dụng mới.

Ngoài ra các kỹ sư CNPM phải nắm vững các trách nhiệm về đạo đức và nghề nghiệp

1.5.2 Phân loại nghề nghiệp trong CNPM

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 cá 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) Loại 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 luôn 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 hóa 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 trùm

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 họa.

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 hóa và để tiêu chuẩn hóa những yêu cầu thành những chính sách và thủ tục chuẩn hóa 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.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 1

Chương 2 TIẾN TRÌNH PHẦN MỀM

2.1 Giới thiệu

Tiến trình phần mềm là cách chúng ta tạo ra sản phẩm phần mềm. Nó bắt đầu với giải thích về mặt khái niệm và kết thúc với một sản phẩm được nghỉ hưu cuối cùng. Trong những giai đoạn trước đó, sản phẩm đi qua một chuỗi các bước như phân tích các yêu cầu - Requirements Analysis (specification), thiết kế, cài đặt, tích hợp, bảo trì sau phát hành, và cuối cùng là nghỉ hưu. Tiến trình phần mềm bao gồm các công cụ, các kỹ thuật mà chúng ta sử dụng để phát triển và bảo trì phần mềm.

Một sản phẩm phần mềm mới có thể được phát triển bằng cách mở rộng và sửa đổi các hệ thống đang tồn tại và bằng cách cấu hình và tích hợp các thành phần hệ thống hoặc phần mềm bán sẵn.

Các tiến trình phần mềm là phức tạp và giống như tất cả các tiến trình sáng tạo, tư duy, cần dựa trên các quyết định, các phán quyết được tạo bởi con người. Do đó, các cố gắng để tự động hóa các tiến trình phần mềm mới đạt được những thành công hạn chế. Các công cụ kỹ nghệ phần mềm được hỗ trợ bởi máy tính (CASE) có thể hỗ trợ một số hoạt động của tiến trình. Tuy nhiên, không có khả năng, ít nhất là trong một vài năm tới, việc mở rộng hơn phạm vi tự động mà ở đó phần mềm tiếp quản việc thiết kế sáng tạo từ các kỹ sư trong tiến trình phần mềm.

Một lý do khác về tính hiệu lực của các công cụ CASE còn hạn chế là vì tính đa dạng mênh mông của các tiến trình phần mềm. Không có tiến trình lý tưởng, và nhiều tổ chức đã phát triển cách tiếp cận riêng của họ để phát triển phần mềm. Các tiến trình được cải tiến để khai thác khả năng của con người trong một tổ chức và những đặc tính cụ thể của các hệ thống đang được phát triển. Với một số hệ thống, như các hệ thống then chốt, tiến trình phát triển hướng cấu trúc được yêu cầu. Với các hệ thống kinh doanh, mà các yêu cầu thay đổi nhanh chóng, tiến trình nhanh, linh hoạt là có thể hiệu quả hơn. Mặc dù, có nhiều tiến trình phần mềm, một số hoạt động cơ bản là chung đối với tất cả các tiến trình:

1. **Phân tích và đặc tả yêu cầu phần mềm:** Chức năng của phần mềm và các ràng buộc trên sự vận hành của nó phải được định nghĩa.
2. **Thiết kế và cài đặt phần mềm** Phần mềm được sản xuất phải thỏa mãn bẩn đặc tả nó.
3. **Thẩm định phần mềm** Phần mềm phải được thẩm định để đảm bảo rằng nó làm những gì mà khách hàng mong muốn

4. Bảo trì và cải tiến phần mềm Phần mềm phải được sửa chữa và tiến hóa để thỏa mãn các nhu cầu thay đổi của khách hàng.

5. Nghỉ hưu phần mềm.

Chúng ta thảo luận các hoạt động trên một cách ngắn gọn trong chương này và thảo luận chúng chi tiết hơn trong các chương sau của môn học

Mặc dù không có tiến trình phần mềm lý tưởng, nhưng có một cơ hội cho việc cải tiến tiến trình phần mềm trong nhiều tổ chức. Các tiến trình có thể chứa nhiều kỹ thuật lỗi thời hoặc có thể không đạt lợi ích thực tiễn tốt nhất trong kỹ nghệ phần mềm công nghiệp. Sự thật rằng, nhiều tổ chức vẫn không nắm bắt được các lợi ích của các phương pháp kỹ nghệ phần mềm trong phát triển phần mềm của họ.

Các tiến trình phần mềm có thể được cải thiện bởi việc chuẩn hóa tiến trình, ở đó tính đa dạng trong các tiến trình cắt chéo/across trong tổ chức được giảm. Điều này dẫn đến việc cải thiện trong giao tiếp, giảm thời gian đào tạo, tạo sự hỗ trợ tiến trình tự động hiệu quả hơn. Việc chuẩn hóa cũng là một bước quan trọng đầu tiên trong việc đưa ra các phương pháp, kỹ thuật kỹ nghệ phần mềm mới và các thực hành kỹ nghệ phần mềm tốt hơn. Chúng ta thảo luận ở các chương sau.

2.2 Các loại mô hình tiến trình phần mềm

Như đã giải thích ở chương 1, mô hình tiến trình phần mềm là một biểu diễn trừu tượng, trực quan của một tiến trình phần mềm. Mỗi mô hình tiến trình biểu diễn một tiến trình từ khía cạnh đặc biệt, và do đó, chỉ cung cấp các thông tin thành phần về tiến trình đó. Trong phần này, chúng ta giới thiệu một số mô hình tiến trình phổ biến (đôi khi được gọi là các biểu đồ tiến trình) và trình bày các mô hình này từ khía cạnh kiến trúc. Tức là, chúng ta xem xét framework của tiến trình chứ không xem xét các hoạt động cụ thể một cách chi tiết của nó.

Các mô hình phổ biến này không là các mô tả rõ ràng về các tiến trình phần mềm. Hơn nữa, chúng là các trừu tượng của tiến trình mà có thể được sử dụng để giải thích các cách tiếp cận khác nhau để phát triển phần mềm. Bạn có thể nghĩ về chúng như các framework tiến trình mà có thể được mở rộng và thích nghi để tạo các tiến trình kỹ nghệ phần mềm cụ thể hơn.

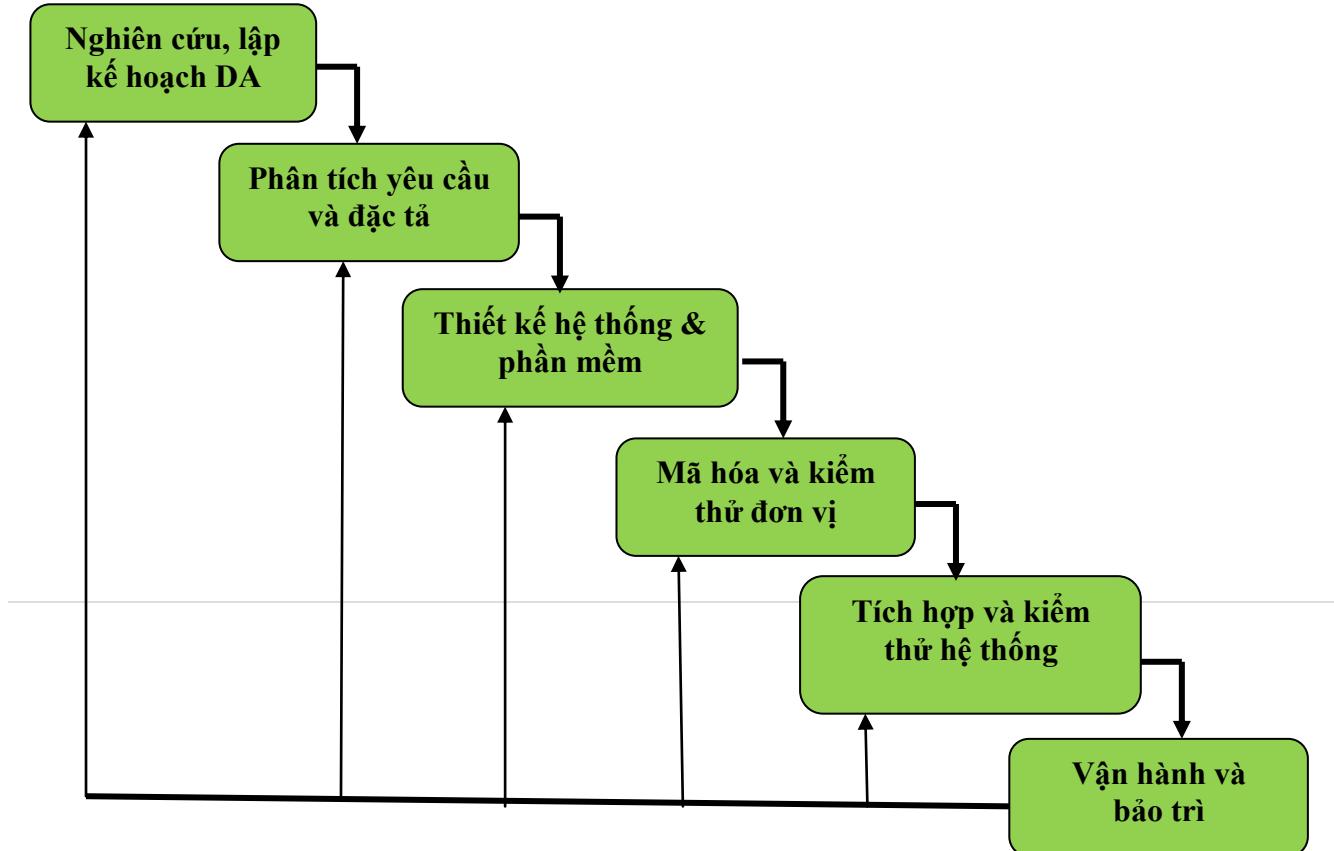
Trong chương này ta tập trung tìm hiểu về năm loại mô hình tiến trình phần mềm tiêu biểu:

- ✓ Mô hình thác nước,
- ✓ Các mô hình phát triển tiến hóa,
- ✓ Các mô hình phát triển hình thức,
- ✓ Mô hình phát triển dựa trên sử dụng lại và các mô hình khác.

Mỗi loại mô hình này bao gồm một số mô hình tiêu biểu.

2.2.1 Mô hình thác nước (Waterfall model)

Là mô hình được công bố đầu tiên, đôi lúc còn được gọi là mô hình kinh điển (classic model) hay mô hình tuyến tính (linear 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. Hình ...biểu diễn mô hình thác nước



Trong mô hình thác nước, các hoạt động phát triển phải được thực hiện một cách tuần tự; kết thúc pha trước, rồi mới được thực hiện pha tiếp theo.

1. Nghiên cứu lập kế hoạch dự án

Hoạt động này nhằm nghiên cứu tính khả thi của dự án phần mềm cần triển khai. Đầu ra của hoạt động này chính là báo cáo khả thi chỉ rõ quyết định dự án có khả thi không. Nếu dự án là khả thi, bản báo cáo này sẽ được đính kèm với bản kế hoạch phát triển dự án.

2. Phân tích yêu cầu và đặc tả

Các dịch vụ của hệ thống, các ràng buộc và mục tiêu được thiết lập qua việc bàn bạc với những người dùng hệ thống. Sau đó chúng được định nghĩa chi tiết và xem như tài liệu đặc tả yêu cầu của hệ thống.

3. Thiết kế hệ thống và phần mềm

Phân chia các yêu cầu cho hệ thống phần mềm hoặc cho hệ thống phần cứng. Thiết lập kiến trúc hệ thống tổng thể. Thiết kế phần mềm bao gồm việc xác định và mô tả các hệ thống phần mềm trừu tượng cơ bản và mối quan hệ giữa chúng

4. Cài đặt và kiểm thử đơn vị

Chuyển bản thiết kế phần mềm thành một tập các chương trình hoặc các đơn vị chương trình. Kiểm thử đơn vị nhằm phát hiện lỗi và chỉ ra rằng nó đã được cài đặt theo đúng đặc tả của nó

5. Tích hợp và kiểm thử hệ thống

Các đơn vị chương trình, hoặc các chương trình riêng lẻ được tích hợp và được kiểm tra như một hệ thống đầy đủ để đảm bảo rằng các yêu cầu phần mềm đã được thỏa mãn. Sau khi kiểm thử hệ thống được phát hành đến khách hàng

6. Vận hành và bảo trì

Đây là giai đoạn lâu nhất trong chu kỳ sống của phần mềm, hệ thống được đưa vào sử dụng trong thực tế. Việc bảo trì gồm sửa chữa các lỗi được phát hiện, cải tiến bản cài đặt các đơn vị hệ thống, nâng cấp các dịch vụ hệ thống khi có các yêu cầu mới

Về nguyên tắc, kết quả của mỗi giai đoạn là một hoặc nhiều tài liệu được phê chuẩn. Giai đoạn sau sẽ không được bắt đầu đến tận khi giai đoạn trước được hoàn thành. Trong thực tế các giai đoạn này là chồng chéo lên nhau, tiến trình phần mềm không chỉ là mô hình tuyến tính chứa một chuỗi các lần lặp về các hoạt động phát triển.

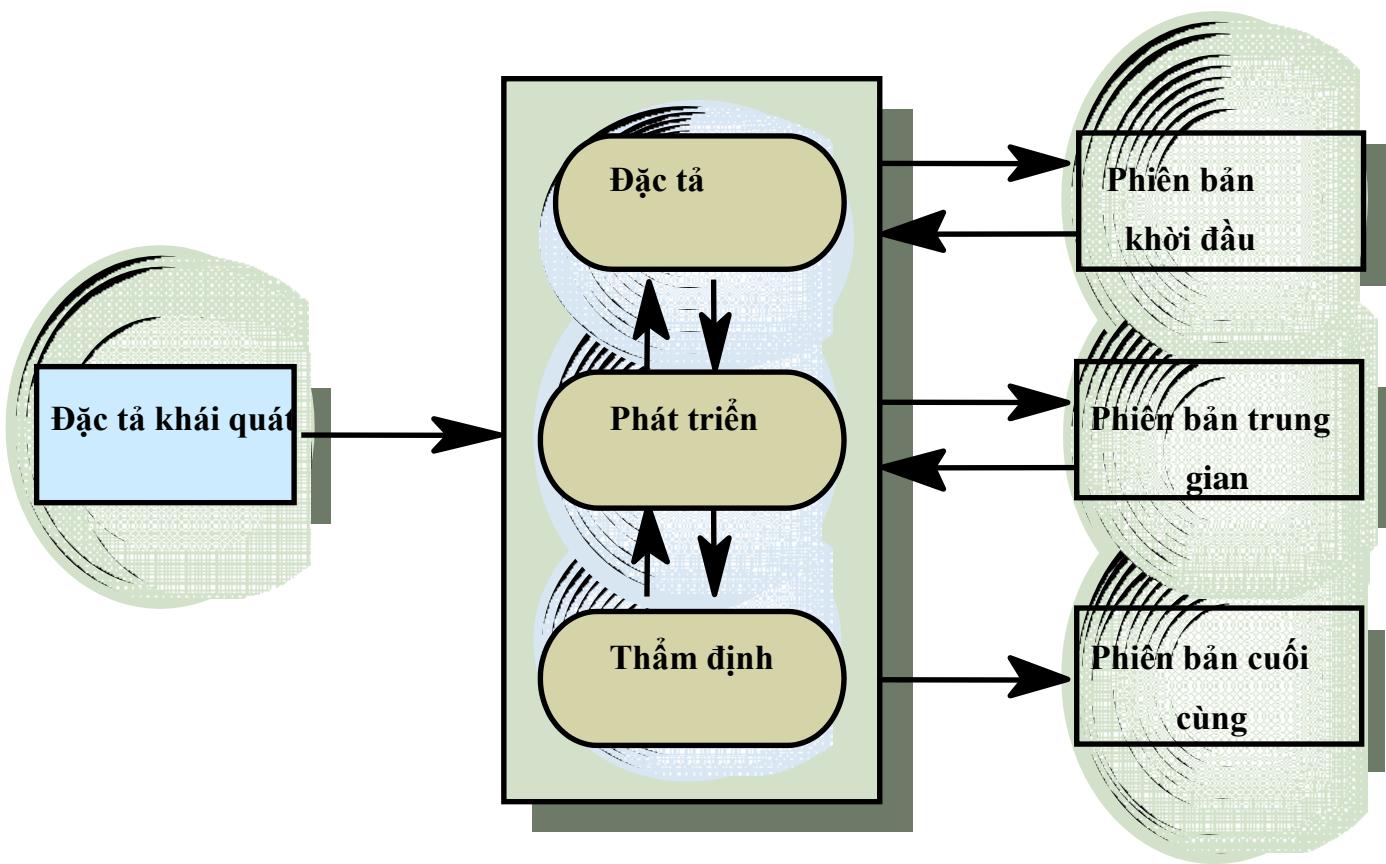
Các đặc điểm của mô hình thác nước:

- ✓ Tách biệt giữa các pha, được tiến hành tuần tự:
 - Khó tuân thủ tuần tự: Dự án lớn thường phải quay lại
 - Khó đáp ứng các yêu cầu thường hay thay đổi của khách hàng.
 - Tốn thời gian chờ đợi.
- ✓ Chậm có phiên bản thực hiện được:
 - Đòi hỏi khách hàng phải kiên nhẫn
 - Sai sót phát hiện muộn có thể là thảm họa
- ✓ Đặc tả kỹ, phân công chuyên trách, hướng tài liệu.
 - Tài liệu quá nhiều, tốn sức người, thời gian dài.

Mô hình thác nước là mô hình ra đời đầu tiên, mang tính tự nhiên, do đó nó được sử dụng khá rộng rãi. Mô hình này khá thích hợp khi hệ thống có các yêu cầu rõ ràng, ổn định. Nó có thể áp dụng cho các phần mềm lớn, phức tạp. Hoạt động bảo trì tương đối thuận lợi.

1.1.2. Các mô hình phát triển tiến hóa

Lược đồ chung nhất của loại mô hình này được biểu diễn như hình vẽ



- ✓ Phát triển ban đầu: Làm việc với khách hàng, đặc tả khái quát hệ thống. Thường bắt đầu với những hiểu biết chưa đầy đủ về hệ thống.
- ✓ Thực hiện phát triển bằng cách làm mẫu: Mục tiêu của hoạt động này là để hiểu hệ thống. Bản mẫu đầu tiên có thể còn sơ sài.
- ✓ Thẩm định phiên bản có được, lặp lại các bước cho đến khi có phiên bản cuối cùng.

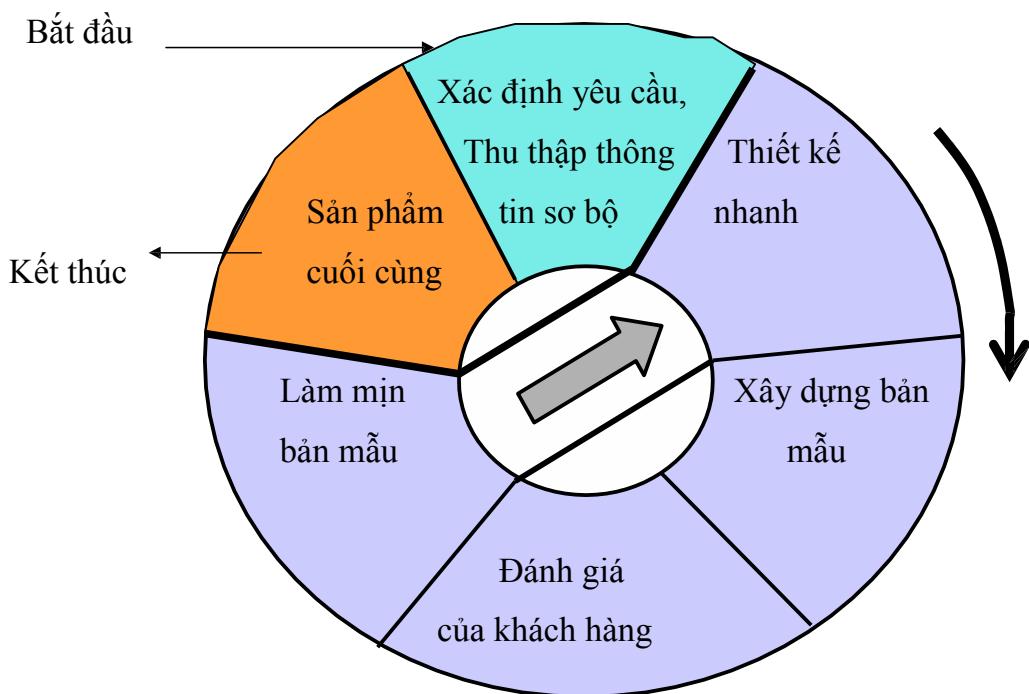
a. Mô hình làm bản mẫu (prototype model)

Khi thu thập các yêu cầu từ khách hàng, thực tế cho thấy, họ thường đưa ra mục tiêu xây dựng hệ thống một cách chung chung, không phát biểu rõ các yêu cầu của mình. Trong trường hợp này, mô hình bản mẫu có thể là sự lựa chọn tốt hơn cho người lập trình. Mô hình này thử 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. Mô hình này thường được sử dụng với hai mục đích sau:

- ✓ **Phát triển thăm dò:** mục của tiến trình là làm việc với khách hàng để thăm dò/khảo sát tỉ mỉ các yêu cầu của họ và phát hành hệ thống cuối cùng. 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 tiêu của tiến trình là để tìm hiểu các yêu cầu của khách hàng. 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.

Mô hình làm bản mẫu được mô tả như hình...

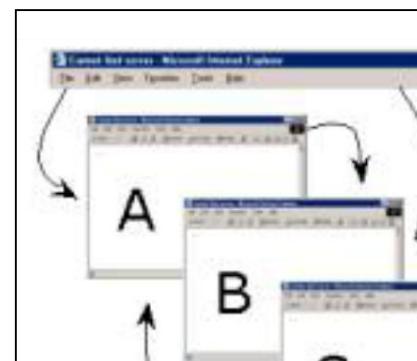


Các loại mẫu gồm:

- ✓ Mẫu trên giấy
- ✓ Mẫu mô tả một phần chức năng
- ✓ Mẫu giao diện
- ✓ Mẫu hướng tới sản phẩm

Các mức độ mẫu:

- ✓ Mẫu dùng xong thì bỏ đi (throw - away approach)
- ✓ Mẫu dùng tiếp cho bước sau (CASE chuyên dụng)
- ✓ Mẫu là phần hệ thống vận hành được (dựa trên thành phần dùng lại)



Mô hình làm bản mẫu có những ưu điểm và nhược điểm chính như sau:

Ưu điểm

- ✓ Nhanh chóng xác định được các yêu cầu
- ✓ Tạo cơ sở ký kết hợp đồng

- ✓ Giúp đào tạo huấn luyện người sử dụng.



Nhược điểm:

- ✓ Tính cấu trúc không cao
- ✓ Khách hàng ít tin tưởng

Khả năng ứng dụng: Thích hợp với

- ✓ Các yêu cầu chưa rõ ràng,
- ✓ Input/output chưa rõ ràng,
- ✓ Khó đánh giá tính hiệu quả của thuật toán.

b. Mô hình xoắn ốc (Spiral model)

Mô hình này xoắn ốc 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 bản 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 xoắn ốc biểu diễn một giai đoạn của tiến trình phần mềm. Do đó, vòng lặp bên trong nhất có thể liên quan đến tính khả thi của hệ thống, vòng lặp tiếp theo định nghĩa các yêu cầu, vòng lặp tiếp theo thiết kế hệ thống, ... Mô hình xoắn ốc là quá trình lặp hướng mở rộng, hoàn thiện dần. Mô hình xoắn ốc được biểu diễn như hình...

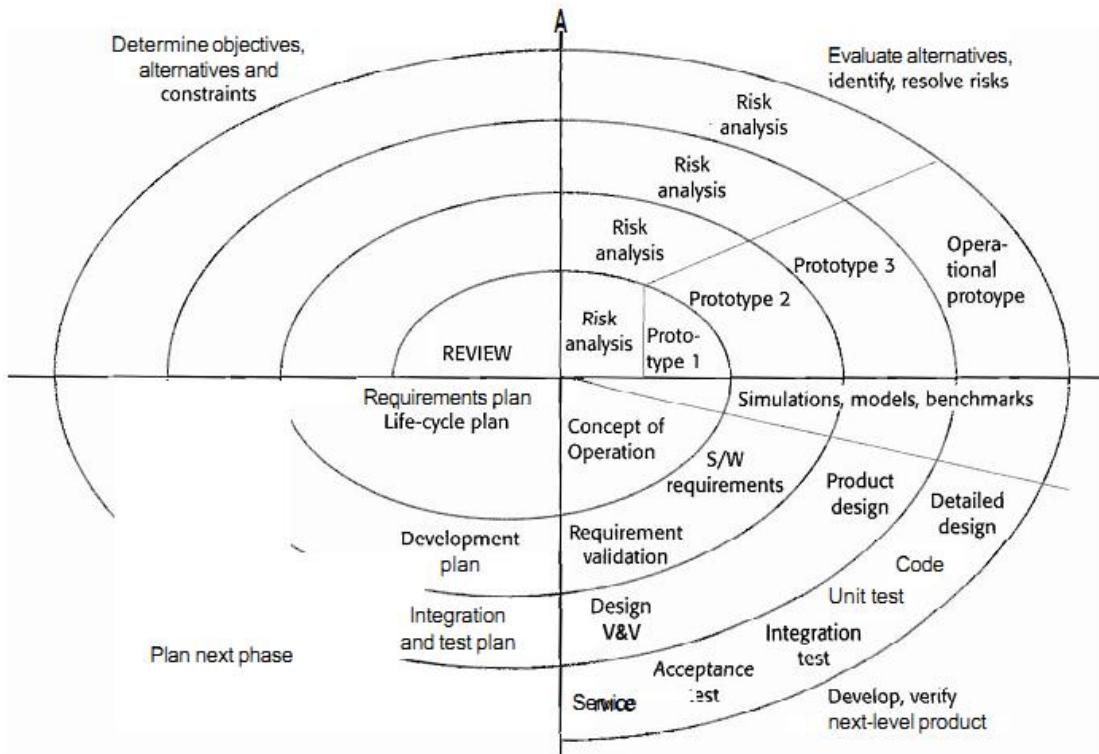


Figure 4.5 Boehm's spiral model of the software process (IEEE, 1988)

Mỗi vòng lặp xoắn ốc được chia làm 4 phần:

- Thiết lập mục tiêu** mục tiêu cụ thể cho các giai đoạn của dự án được định nghĩa. Các ràng buộc trên tiến trình và sản phẩm được định nghĩa, lập kế hoạch quản lý chi tiết dự án được vạch ra. Các rủi ro dự án được xác định. Các chiến lược thay thế, phụ thuộc vào các rủi ro này, có thể được lên kế hoạch.
- Đánh giá và giảm rủi ro** Với mỗi rủi ro dự án đã được xác định, một phân tích chi tiết được thực hiện. Các bước được thực hiện để giảm rủi ro. Ví dụ, nếu có một rủi ro rằng các yêu cầu không phù hợp, mẫu thử hệ thống có thể được phát triển.
- Phát triển và thẩm định** Sau khi đánh giá rủi ro, một mô hình phát triển cho hệ thống được lựa chọn. Ví dụ, nếu các rủi ro giao diện người dùng bị ảnh hưởng lớn, mô hình phát triển phù hợp có lẽ là nguyên mẫu tiến hóa. Nếu các rủi ro an toàn được quan tâm chính, việc phát triển dựa trên các biến đổi hình thức có thể là phù hợp nhất, vv....mô hình thác nước là phù hợp nhất nếu rủi ro chính được xác định là việc tích hợp các hệ thống con.
- Lập kế hoạch** Dự án được xem xét lại và quyết định liệu có tiếp tục vòng xoắn ốc tiếp theo hay không được tạo. Nếu quyết định là có, các kế hoạch được vạch ra cho giai đoạn tiếp theo của dự án.

Điểm khác nhau chính giữa mô hình thác nước và các mô hình tiến trình phần mềm khác là việc nhận dạng rõ ràng các rủi ro trong mô hình xoắn ốc. Một cách phi hình thức, rủi ro đơn giản là một thứ gì đó mà làm sai. Ví dụ, nếu mục đích là sử dụng một ngôn ngữ lập trình mới, một rủi ro đó là các chương trình dịch có thể không đáng tin cậy, hoặc không tạo mã đích hiệu quả và đầy đủ. Các rủi ro dẫn đến các vấn đề dự án như chấm trễ lịch biểu, vượt quá chi phí....do đó tối thiểu rủi ro là hoạt động quản lý dự án rất quan trọng. Quản lý rủi ro, là một phần bản chất của dự án, được trình bày trong chương 5.

Một chu kỳ của vòng ốc bắt đầu bằng việc dựng lên các mục tiêu ví dụ như sự thực hiện và về mặt chức năng. Nhiều cách lựa chọn thay thế khác nhau để đạt được các mục tiêu này và các ràng buộc được đưa ra trên mỗi chu trình tiếp theo được liệt kê ra. Mỗi lựa chọn thay thế được đánh giá đối với mỗi mục tiêu và các nguồn rủi ro dự án được xác định. Bước tiếp theo là giải quyết các rủi ro này bằng các hoạt động tập hợp thông tin như phân tích chi tiết hơn, mẫu thử hoặc mô phỏng. Một khi các rủi ro đã được đánh giá, việc phát triển được thực hiện, theo sau là hoạt động lập kế hoạch cho giai đoạn tiếp theo của tiến trình.

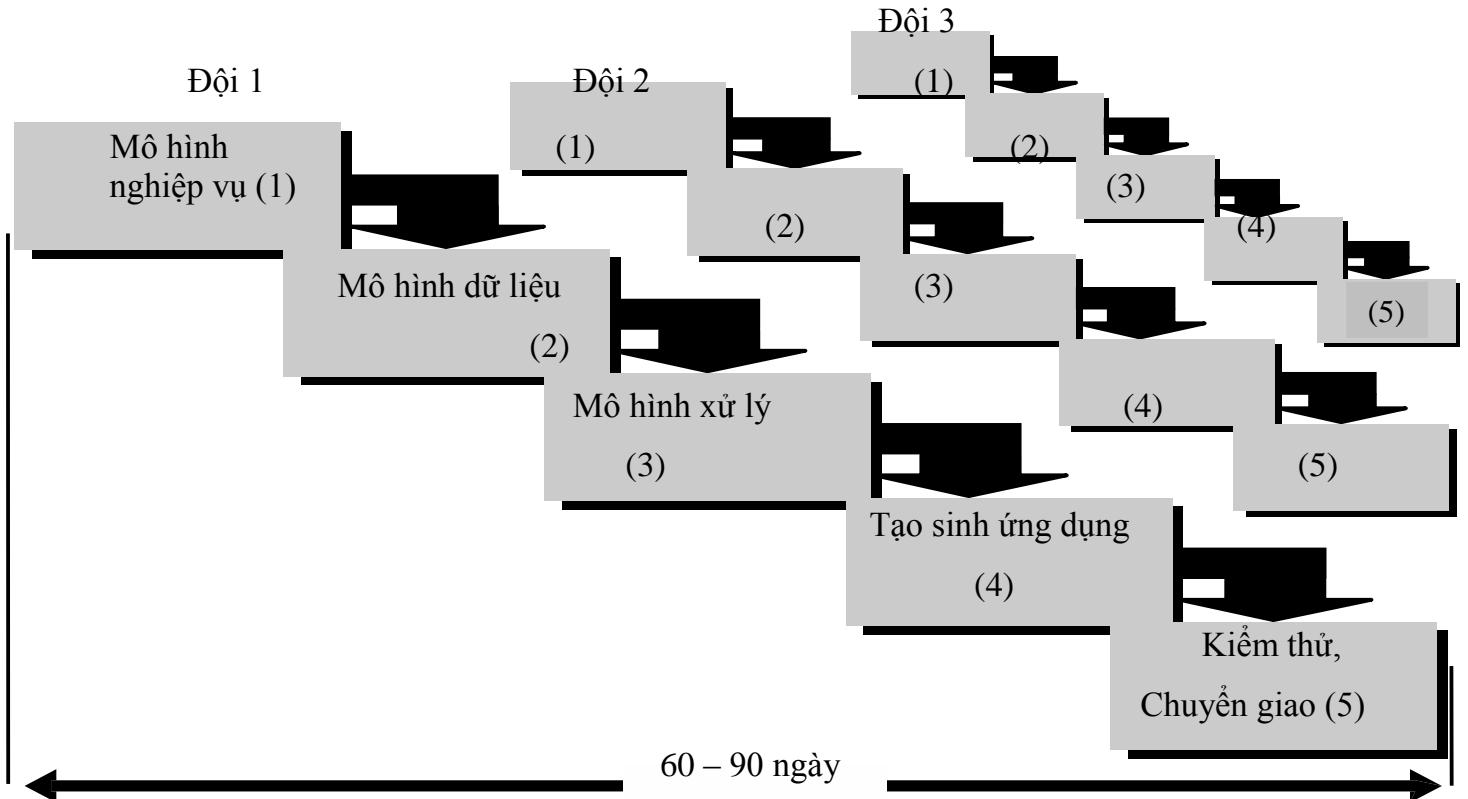
Các đặc điểm của mô hình xoắn ốc

- ✓ Hợp với hệ thống lớn. Có thể phân chia hệ thống thành các thành phần cốt yếu đến các thành phần thứ yếu. Có thể kiểm soát rủi ro ở từng mức tiến hóa
- ✓ Sau mỗi lần tăng vòng thì có thể chuyển giao kết quả thực hiện được cho khách hàng 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 chưa được dùng rộng rãi như mô hình thác nước hoặc mô hình làm bản mẫu

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

Mô hình RAD được biểu diễn như hình...

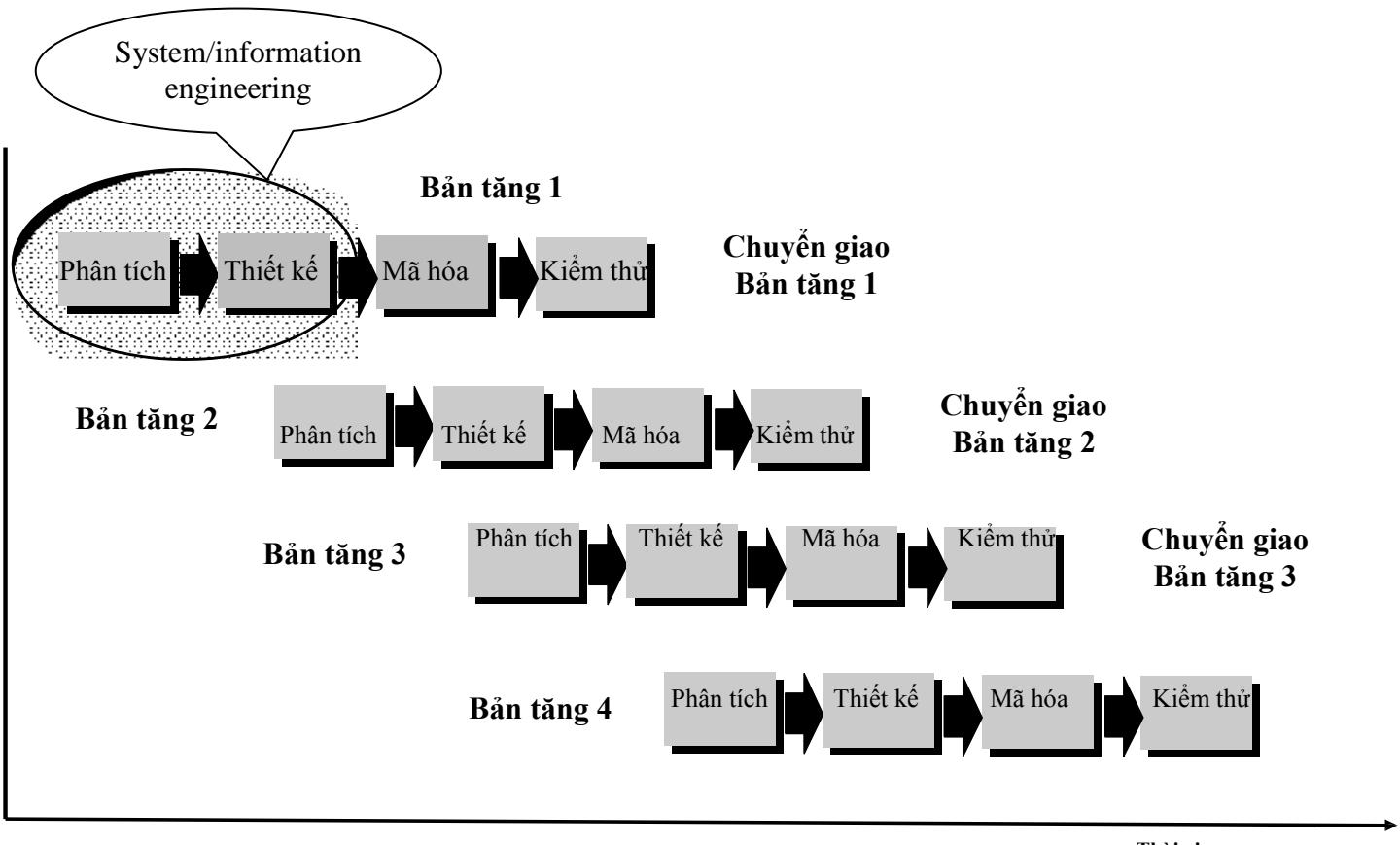


Đặc điểm của mô hình RAD:

- ✓ 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.

d. Mô hình tăng trưởng (incremental model)

Mô hình tăng trưởng được biểu diễn như hình...

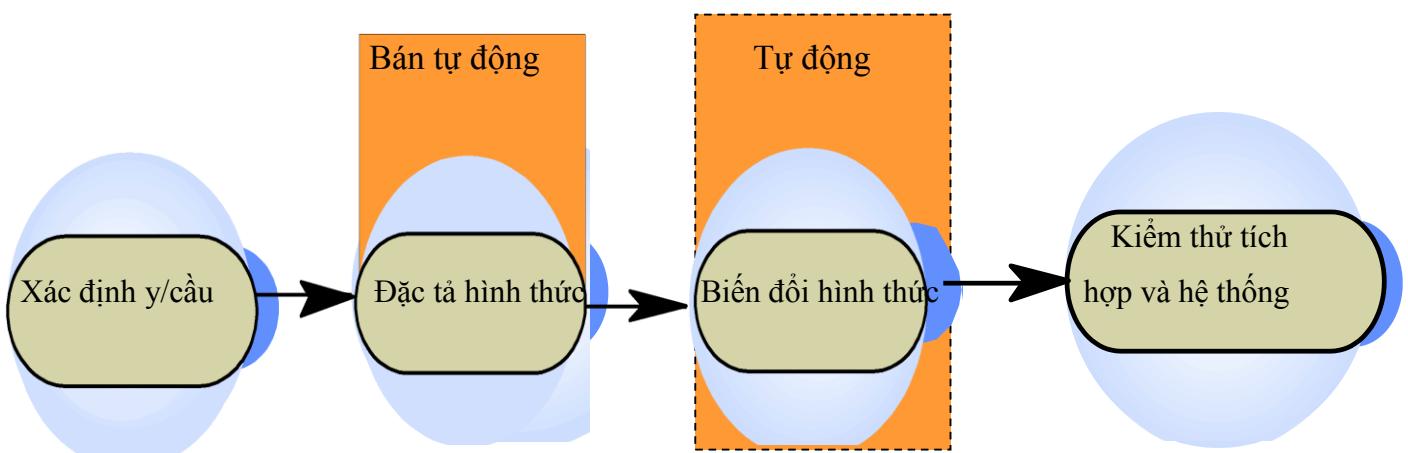


Đặc điểm của mô hình tăng trưởng

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

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

Mô hình phát triển được biểu diễn như hình...

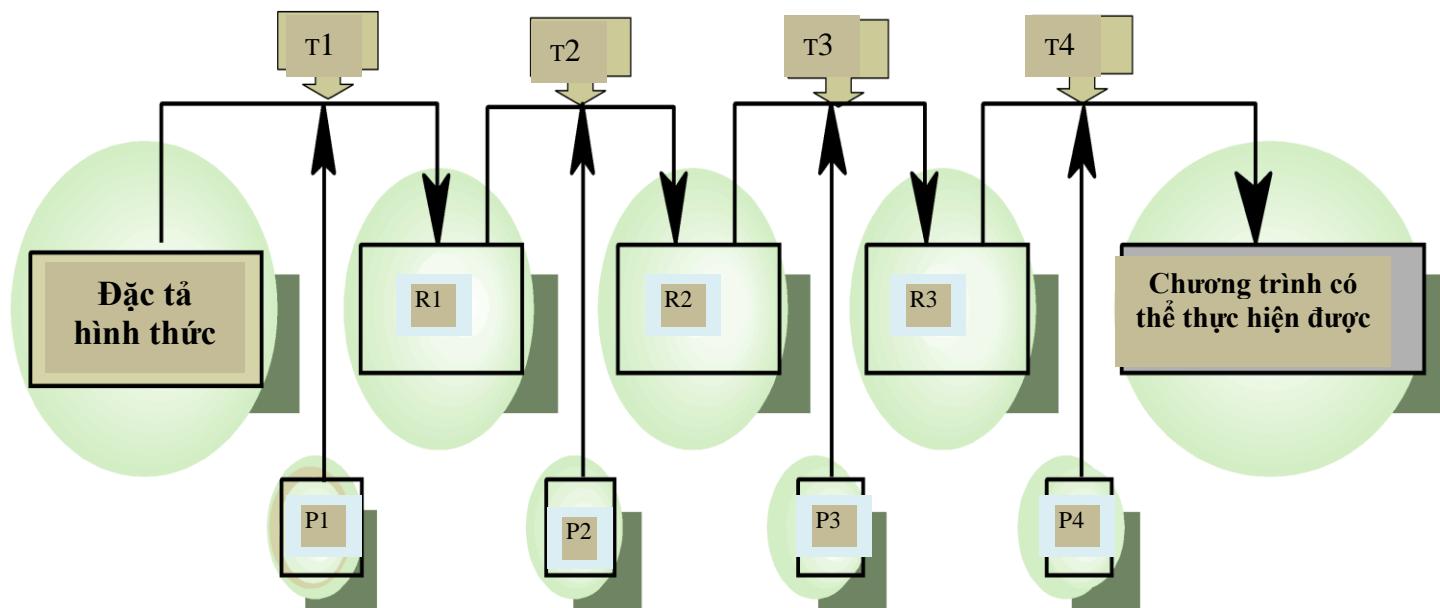


Các bước đặc trưng của tiến trình phát triển:

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

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.

1.1.4. 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:

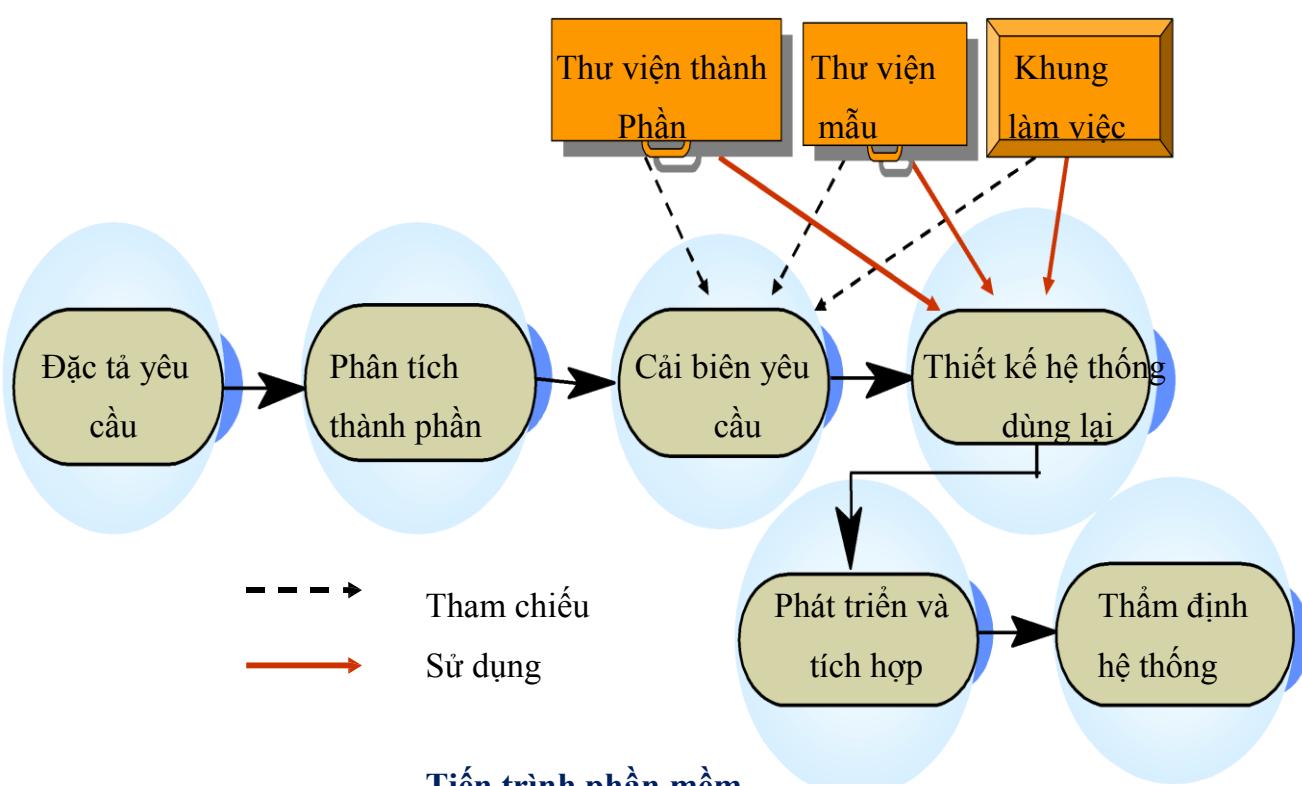
- ✓ Hệ thống được cấu thành từ các đối tượng
- ✓ Đối tượng bao gói cả dữ liệu và các xử lý trên dữ liệu đó
- ✓ Các đối tượng liên kết với nhau bằng truyền thông điệp.

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.
 - Tác động cục bộ, dễ bảo trì, dễ dùng lại
- ✓ *Tính kế thừa*:
 - Xây dựng được các lớp cơ sở (chung),
 - 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...

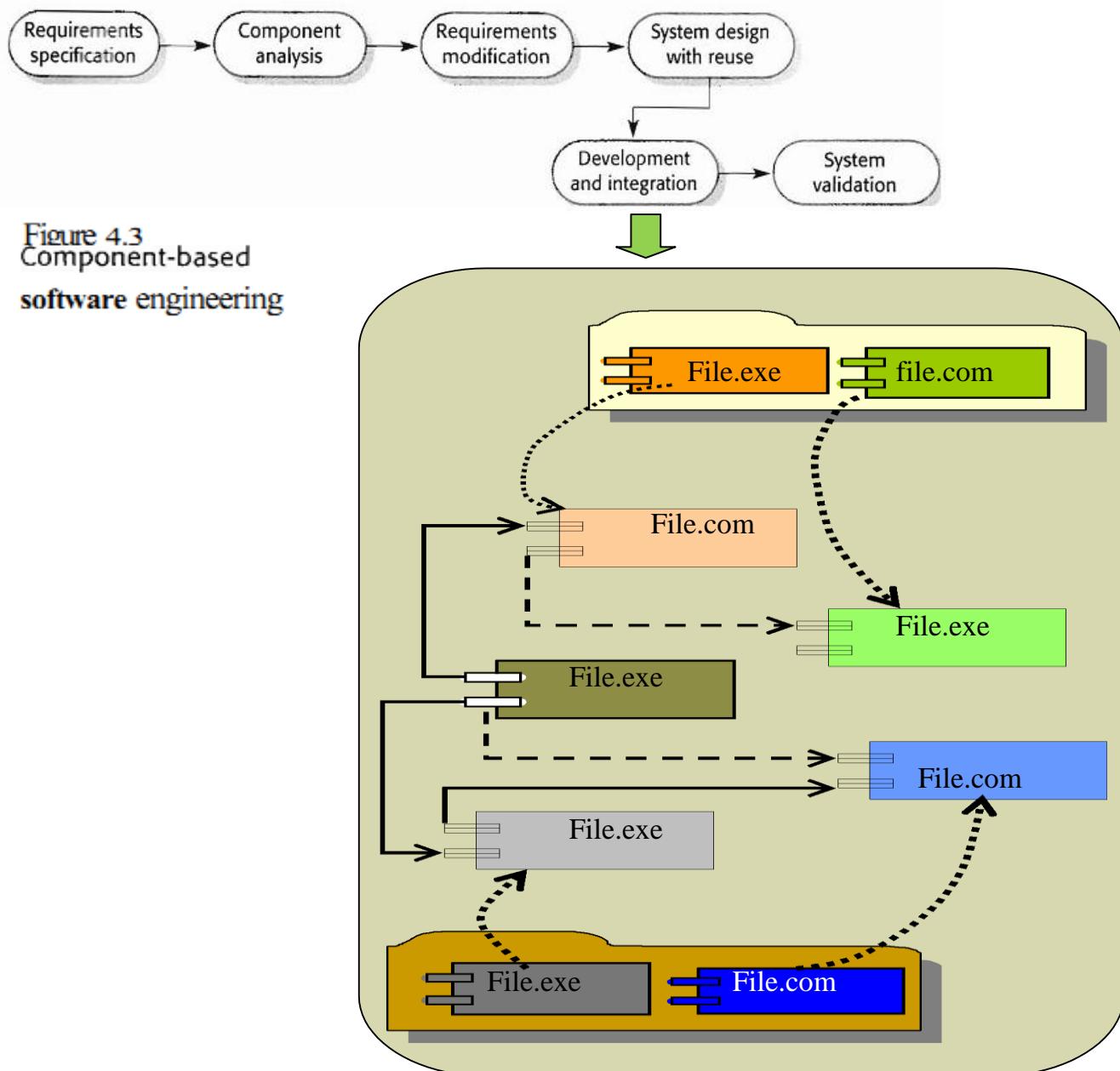


Các giai đoạn của tiến trình được tiến hành tuần tự, trước tiên cần xác định và đặc tả các yêu cầu hệ thống. Phân tích thành phần nhằm phân tích hệ thống thành các phần yêu cầu nhỏ hơn. Tiếp theo ta cần cải biến các yêu cầu theo hướng thành phần, hướng mẫu hoặc hướng khung, => Đây là hướng tiếp cần phát triển phần mềm quan trọng, cần kinh nghiệm, các công cụ hỗ trợ còn hạn chế.

b. Các hướng sử dụng lại

1. Kỹ nghệ phần mềm dựa trên cấu phần (Component Based Software Engineering - CBSE)

Trong khoảng mươi 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.



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 kiểm 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 thỏa 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.

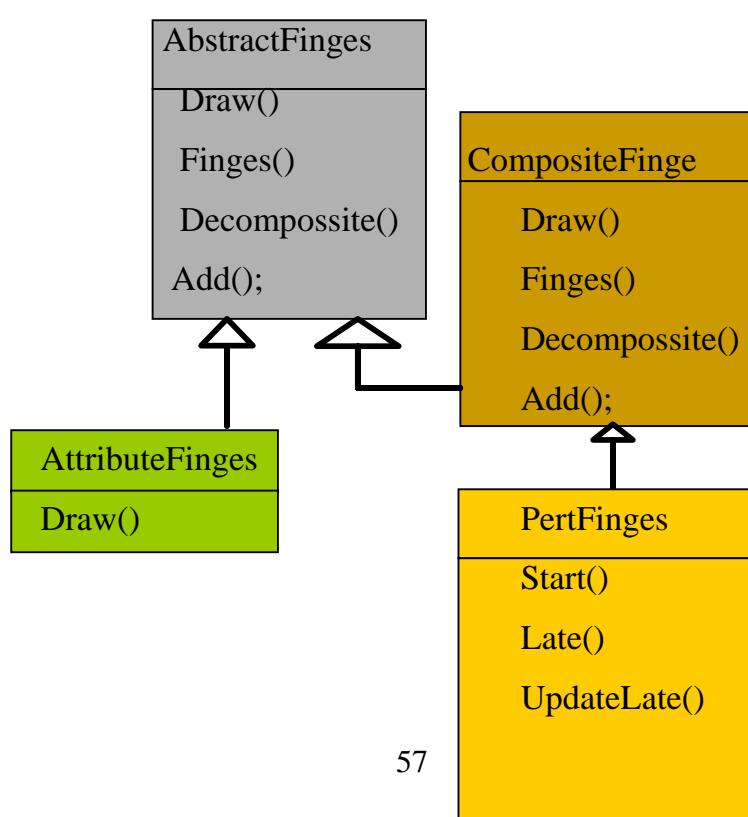
Thành phần phần mềm được sử dụng lại có độ chính xác cao và có thể nói là không chứa lỗi. Mặc dù không thường xuyên được chứng minh về mặt hình thức nhưng với việc sử dụng lại, lỗi được tìm thấy và được loại trừ; chất lượng của thành phần được cải thiện là một tất yếu. Phần mềm được phát triển theo phương 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 kiểm thành phần hiệu quả.

2. 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)

Một số mẫu trong thư viện được mô tả như hình...

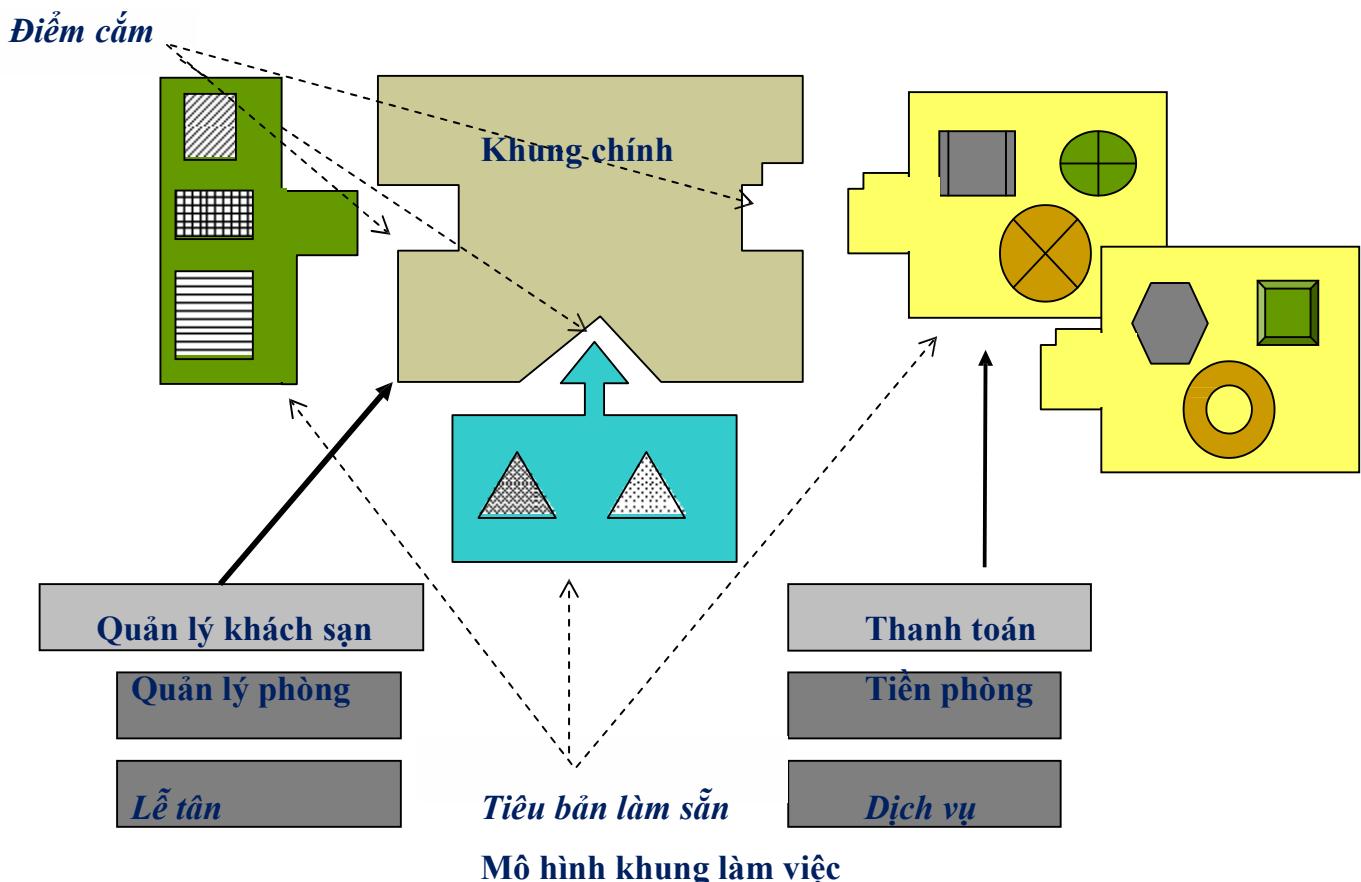


3. Phát triển khung làm việc (Frame Development)

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

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

Hình sau đây chỉ ra mô hình một Framework của hệ thống quản lý khách sạn.



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 tiến trình phần mềm

2.2.1. Đặc tả yêu cầu phần mềm

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

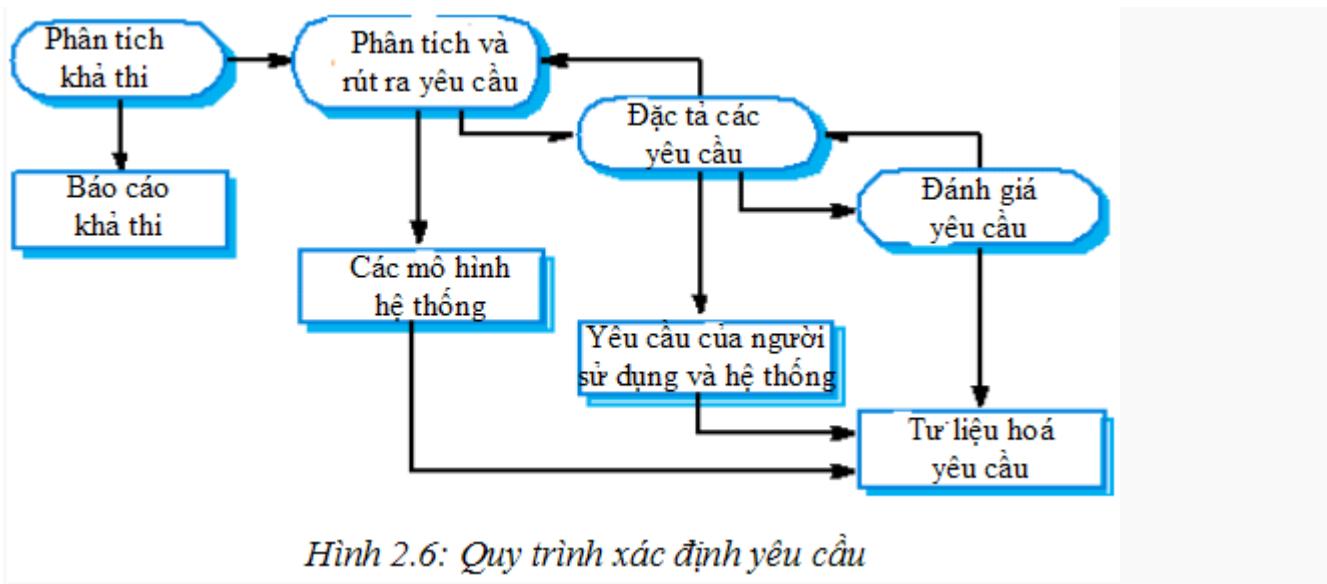
* **Yêu cầu hệ thống:** là những tài liệu có cấu trúc, được mô hình hoá, mô tả chi tiết về các chức năng, dịch vụ và các ràng buộc vận hành của hệ thống. Yêu cầu hệ thống sẽ định nghĩa những gì cần phải xây dựng, cho nên nó có thể trở thành bản hợp đồng giữa khách hàng và nhà thầu. Các yêu cầu hệ thống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

+ Các yêu cầu miền ứng dụng

- **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.



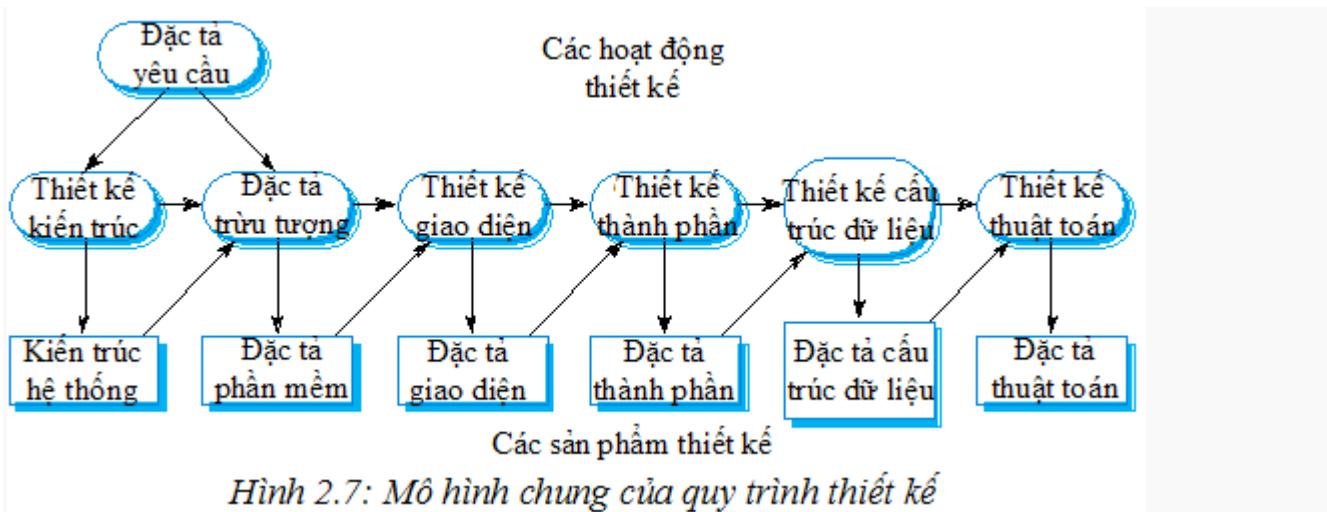
Hình 2.6: Quy trình xác định yêu cầu

2.2.2 Thiết kế phần mềm và cài đặt

a. Thiết kế phần mềm

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 trùu 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

Bản thiết kế phần mềm là một mô tả về cấu trúc của phần mềm được cài đặt, dữ liệu là một phần của hệ thống, giao diện giữa các thành phần hệ thống, các giải thuật được sử dụng. Người thiết kế không đi tới một thiết kế hoàn thiện ngay lập tức mà phát triển thiết kế lặp lại qua một số phiên bản. Tiến trình thiết kế thêm dần tính hình thức và tính chi tiết, do đó thiết kế được phát triển theo vết liên tục để sửa các thiết kế sớm hơn.

Tiến trình thiết kế có thể bao gồm việc phát triển một số mô hình hệ thống với các mức độ trừu tượng khác nhau. Khi một thiết kế được phân tích, các lỗi và các thiếu sót trong các giai đoạn trước được phát hiện. Chúng phản hồi lại để cho phép các mô hình thiết kế trước được cải tiến. Hình 2.7 làm một mô hình của tiến trình này chỉ ra các mô tả thiết kế mà có thể được tạo tuần tự. Trong thực tế, các hoạt động tiến trình thiết kế là đan xen nhau. Các phản hồi từ giai đoạn này đến giai đoạn khác dẫn đến phải làm lại thiết kế là không thể tránh khỏi trong mọi tiến trình thiết kế.

Bản đặc tả cho giai đoạn tiếp theo là đầu ra của mỗi giai đoạn thiết kế. Đặc tả này có thể là một đặc tả hình thức, trừu tượng được tạo ra để làm sáng tỏ các yêu cầu, hoặc nó có thể là một đặc tả về những phần nào đó của hệ thống được nhận thức rõ. Vì tiến trình thiết kế liên tục, các đặc tả này trở nên chi tiết dần lên. Kết quả cuối cùng của tiến trình là các đặc tả chính xác về các giải thuật và các cấu trúc dữ liệu được cài đặt.

Đây là mô hình chung của tiến trình thiết kế và các tiến trình thực tế, thực tế có thể thích nghi nó theo các cách khác nhau. Các thích nghi có thể là:

1. Hai giai đoạn sau của thiết kế - thiết kế cấu trúc dữ liệu và giải thuật – có thể được trì hoãn đến tận tiến trình cài đặt
2. Nếu cách tiếp cận thăm dò để thiết kế được sử dụng, các giao diện hệ thống có thể được thiết kế sau khi các cấu trúc dữ liệu đã được đặc tả.

3. Giai đoạn đặc tả trùu tượng có thể bị nhảy qua, mặc dù nó thường là phần bản chất của thiết kế các hệ thống then chốt.

Ở đâu các phương pháp phát triển nhanh được sử dụng (xem chương 17), các đầu ra của tiến trình thiết kế sẽ không là các tài liệu đặc tả riêng rẽ mà sẽ được biểu diễn trong mã của chương trình. Sau khi kiến trúc hệ thống được thiết kế, các giai đoạn sau của thiết kế là tăng dần. Mỗi lần tăng được biểu diễn bởi mã chương trình hơn là một mô hình thiết kế.

Cách tiếp cận tương phản được thực hiện bởi các phương pháp có cấu trúc cho thiết kế dựa trên việc tạo các mô hình đồ họa của hệ thống (xem chương 8), và trong một số trường hợp, tự động phát sinh mã từ các mô hình này. Các phương pháp có cấu trúc được phát minh trong những năm 1970 để hỗ trợ thiết kế hướng chức năng (Constantine and Yourdon, 1979; Gane and Sarson, 1979). Nhiều phương pháp đang cạnh tranh để hỗ trợ thiết kế hướng đối tượng được đề xuất (Robinson, 1992; Booch, 1994) và các phương pháp này được hợp nhất trong những năm 1990 để tạo ngôn ngữ mô hình hóa hợp nhất – UML và các tiến trình thiết kế hợp nhất được kết hợp chặt chẽ (Rumbaugh, et al., 1991; Booch, et al., 1999; Rumbaugh, et al., 1999a; Rumbaugh, et al., 1999b). Tại thời điểm viết sách, chủ điểm với UML (UML 2.0) đang được xem xét thực hiện.

Một phương pháp có cấu trúc bao gồm một mô hình tiến trình thiết kế, các ký hiệu biểu diễn thiết kế, các định dạng báo cáo, các luật, và các hướng dẫn thiết kế. Các phương pháp có cấu trúc có thể hỗ trợ một số hoặc tất cả các mô hình sau đây của một hệ thống:

1. *Mô hình đối tượng* chỉ ra các lớp đối tượng được sử dụng trong hệ thống và các phụ thuộc của chúng.
2. *Mô hình tuần tự* chỉ ra các đối tượng trong hệ thống tương tác như thế nào khi hệ thống thực hiện.
3. *Mô hình chuyển đổi trạng thái* chỉ ra các trạng thái hệ thống và các nút bấm/sự kiện cho các dịch chuyển từ trạng thái này sang trạng thái khác.
4. *Mô hình cấu trúc* ở đó các thành phần hệ thống và các kết tập được tư liệu hóa.
5. *Mô hình luồng dữ liệu* ở đó hệ thống được mô hình hóa sử dụng các chuyển đổi dữ liệu xảy ra khi nó được xử lý. Mô hình này thường không được sử dụng trong các phương pháp hướng đối tượng nhưng nó vẫn thường xuyên được sử dụng trong việc thiết kế hệ thống nghiệp vụ và hệ thống thời gian thực.

Trong thực tế, các phương pháp có cấu trúc thực sự là các hiện thân về sự chuẩn hóa tốt. Theo sau các phương pháp này và việc áp dụng các hướng dẫn có thể dẫn đến một thiết kế hợp lý. Tính sáng tạo trong thiết kế vẫn được yêu cầu để quyết định về việc phân tích hệ thống và

để đảm bảo rằng thiết kế thỏa mãn đặc tả hệ thống. Các nghiên cứu thực nghiệm về các bản thiết kế (Bansler và Bodker, 1993) chỉ ra rằng chúng hiếm khi tuân theo các phương pháp một cách mù quáng. Chúng nhặt và chọn từ các hướng dẫn phù hợp với các hoàn cảnh riêng.

b. Lập trình/mã hóa và gỡ rối

Lập trình 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.

Lập trình là một hoạt động cá nhân và không có tiến trình chung để tuân theo. Một số lập trình viên bắt đầu với các thành phần mà họ hiểu, phát triển chúng, sau đó mới đến các thành phần ít hiểu biết hơn. Những người khác sử dụng cách tiếp cận ngược lại, tạm thời để lại các thành phần đã quen lại sau vì họ biết cách để phát triển chúng. Một số người phát triển thích xác định dữ liệu trước theo trình tự sau đó sử dụng tiến trình này để điều khiển việc phát triển chương trình; những người khác cho phép dữ liệu cụ thể chưa được xác định lâu đến chừng nào có thể.

Thông thường, các lập trình viên thực hiện một số kiểm thử về mã mà họ đã phát triển. Kiểm thử này thường bộc lộ các khuyết điểm mà phải được loại bỏ khỏi chương trình. Việc này gọi là gỡ rối. Kiểm thử và gỡ rối là các tiến trình khác nhau. Kiểm thử xác định các khuyết điểm tồn tại. Gỡ rối liên quan đến định vị và sửa các khuyết điểm được phát hiện này.

Hình 4.8 minh họa các giai đoạn gỡ rối. Các khuyết điểm phát hiện trong mã phải được định vị và chương trình được sửa đổi để thỏa mãn các yêu cầu của nó. Sau đó việc kiểm thử phải được lặp lại để đảm bảo rằng các thay đổi đã được tạo một cách đúng đắn. Do đó, tiến trình gỡ rối là một phần của cả hai hoạt động phát triển phần mềm và kiểm thử phần mềm.

Figure 4.8 The debugging process



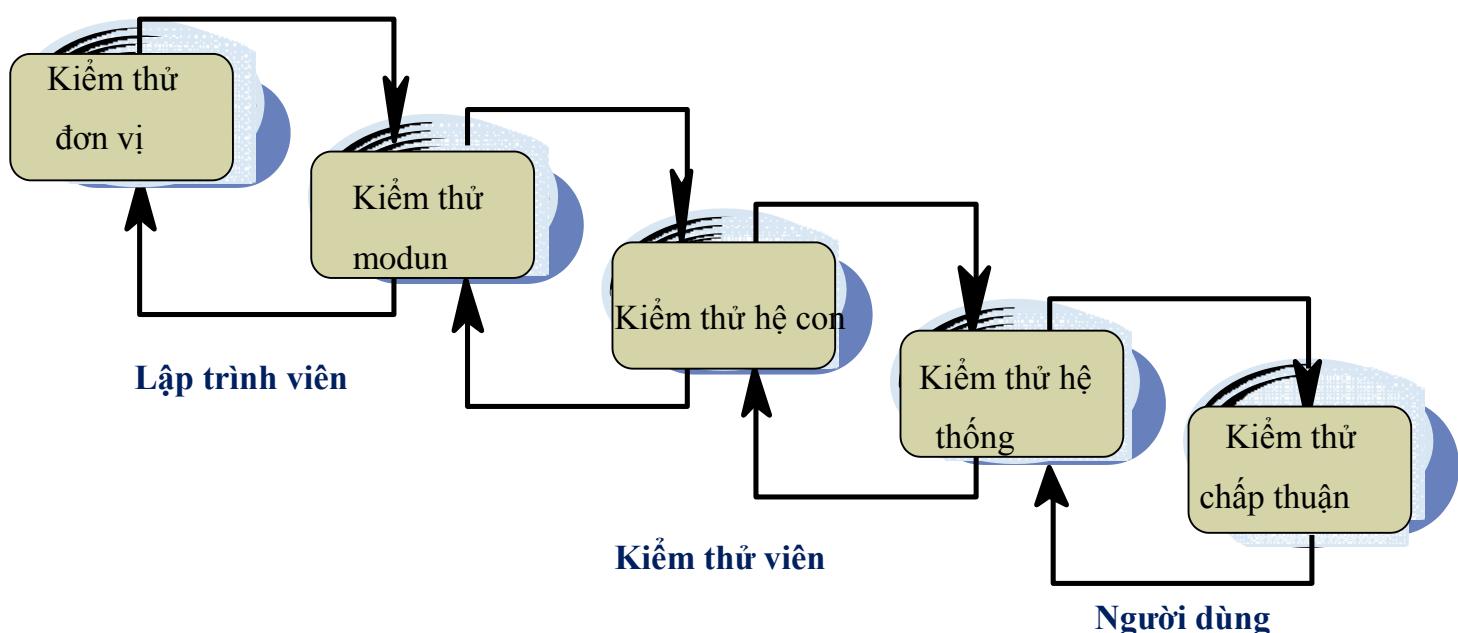
Khi gỡ rối, bạn sinh ra các giả thiết về các hành vi có thể quan sát của chương trình sau đó kiểm tra các giả thiết này với hy vọng tìm được các khuyết điểm mà dẫn đến lỗi thường. Việc kiểm thử các giả thiết có thể gồm việc theo vết mã chương trình được điều khiển bằng tay. Bạn có thể viết các trường hợp kiểm thử mới để khoanh vùng có vấn đề. Sự tác động của các công cụ gỡ rối chỉ ra ngay lập tức giá trị của các biến chương trình và vết của các mệnh lệnh đã thực hiện có thể được sử dụng để trợ giúp tiến trình gỡ rối 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 thỏa 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 được chỉ ra trong hình....



Trong đó:

- ✓ **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 con/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.

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.

Các giai đoạn sau của kiểm thử gồm tích hợp công việc từ một số lập trình viên và phải được lập kế hoạch cấp cao. Một đội kiểm thử độc lập nên làm việc bắt đầu từ các kế hoạch kiểm thử được lập từ trước được phát triển từ khi đặc tả và thiết kế hệ thống. Hình 4.10 minh họa các kế hoạch kiểm thử liên kết các hoạt động kiểm thử và phát triển như thế nào.

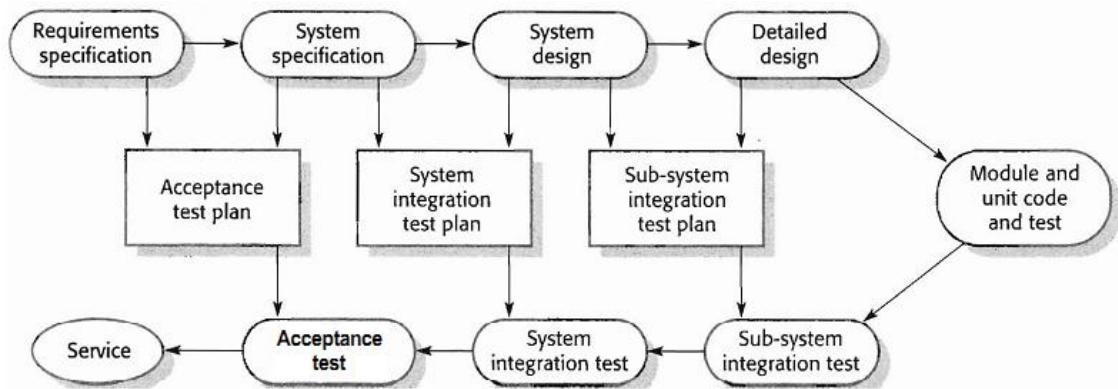


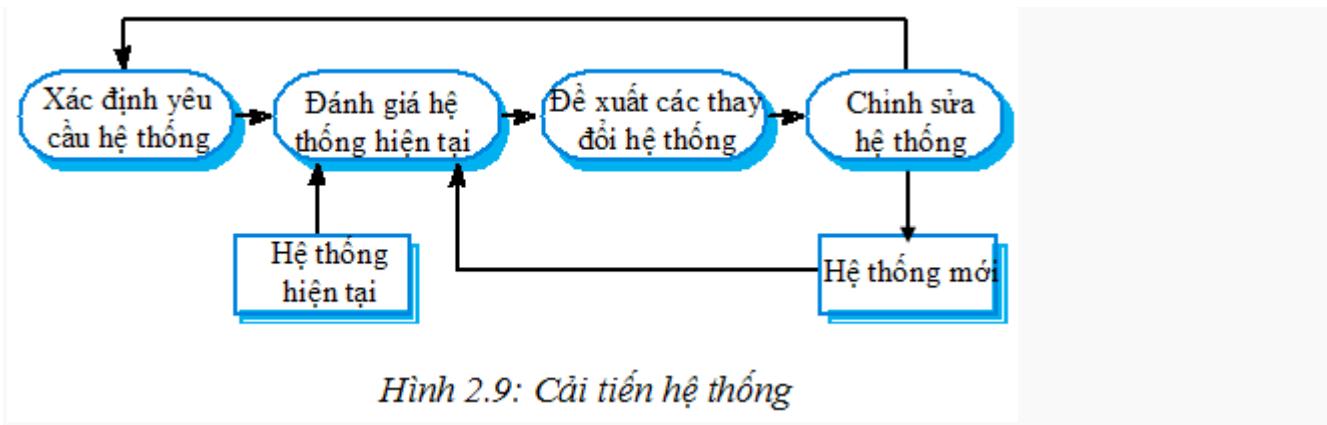
Figure 4.10 Testing phases in the software process

Kiểm thử chấp thuận đôi khi được gọi là kiểm thử Alpha. Các hệ thống đặt hàng được phát triển cho khách hàng đơn lẻ. Tiến trình kiểm thử alpha tiếp tục đến tận khi người phát triển hệ thống và khách hàng đồng ý rằng hệ thống được phát hành là một bản cài đặt các yêu cầu hệ thống có thể chấp nhận.

Khi hệ thống được bán ra thị trường như một sản phẩm phần mềm, tiến trình kiểm thử được gọi là kiểm thử Beta thường được sử dụng. Kiểm thử Beta gồm việc phát hành hệ thống đến một số khách hàng có khả năng, những người mà đồng ý sử dụng phần mềm đó. Họ báo cáo các vấn đề cho những người phát triển hệ thống. Những báo cáo này cho biết tình hình sản phẩm khi sử dụng thực tế và các phát hiện lỗi có thể không được dự đoán trước bởi những người xây dựng hệ thống. Sau những phản hồi này, hệ thống được sửa và phát hành lại để thực hiện kiểm thử Beta tiếp theo hoặc là bán ra thị trường.

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 là điều tất yếu. Quá trình tiến hóa phần mềm được chỉ ra trong hình Thông thường chi phí để bảo trì và cải tiến thường đắt hơn nhiều so với chi phí xây dựng phần mềm.



2.3 Công nghệ CASE

Các kỹ thuật CASE đã góp phần đáng kể để hoàn thiện tiến trình phần mềm cả về trình tự, tiến độ và chất lượng như tự động hóa một phần hoạt động mô hình hóa, hỗ trợ quản lý... Các kỹ thuật CASE đã có những cải tiến đáng kể trong tiến trình phần mềm. Tuy nhiên chưa có những cải tiến trọng yếu vì:

- ✓ Kỹ nghệ phần mềm yêu cầu những suy nghĩ sáng tạo. Những suy nghĩ này chưa sẵn sàng cho hoạt động tự động hóa.
- ✓ Kỹ nghệ phần mềm là một hoạt động nhóm. Với các dự án lớn, thường chiếm rất nhiều thời gian dành cho các tương tác nhóm.

Những hoạt động mà các kỹ thuật CASE chưa thể tự động hóa:

- ✓ Sự suy nghĩ sáng tạo trong SE
- ✓ Lựa chọn giải pháp công nghệ
- ✓ Giao tiếp khi làm việc nhóm
- ✓ Thực hiện việc quản lý dự án.

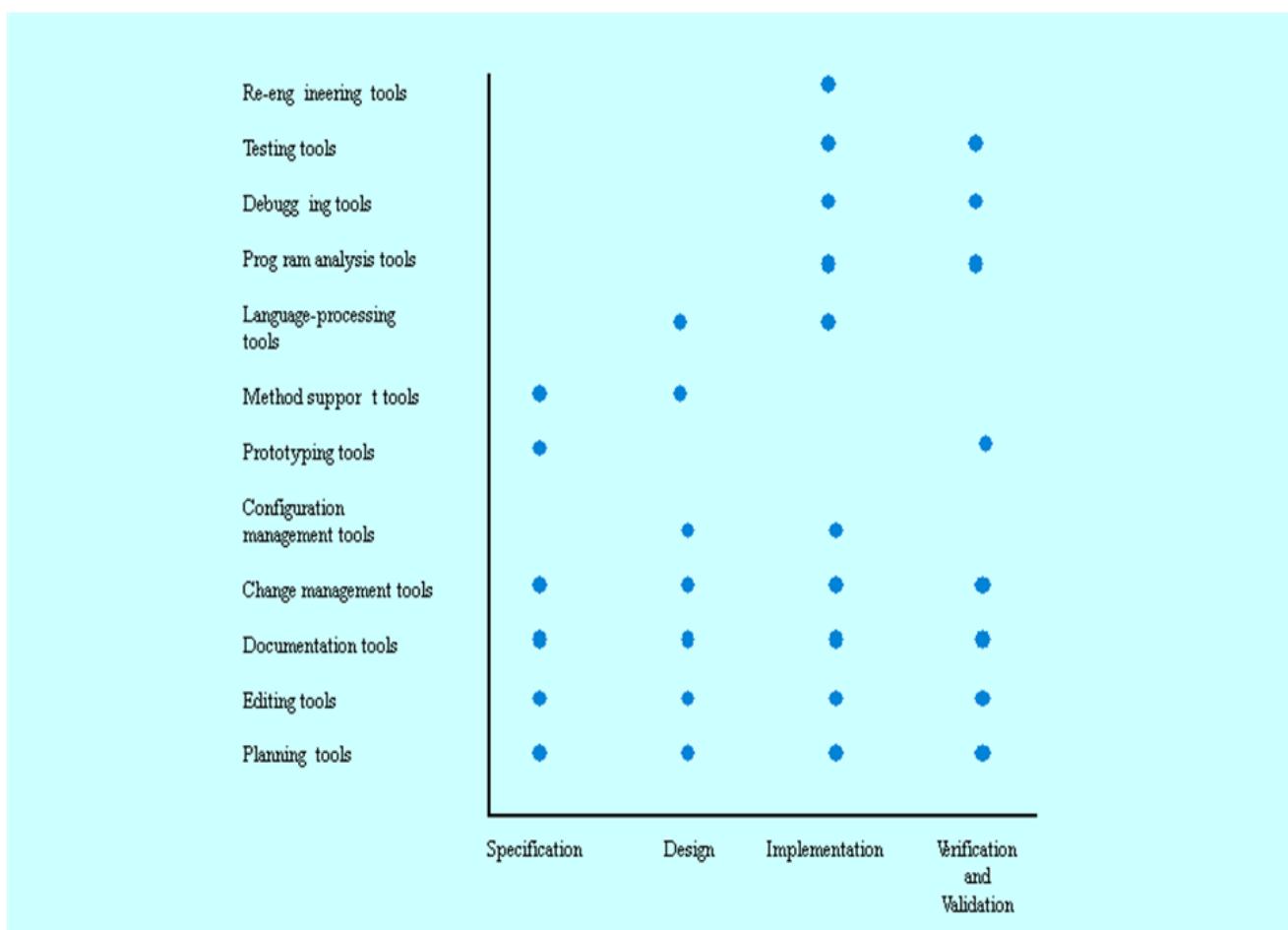
Phân loại CASE

Việc phân loại CASE theo những khía cạnh khác nhau giúp chúng ta thực sự hiểu được các kiểu công cụ CASE và sự hỗ trợ của chúng cho các hoạt động kỹ nghệ phần mềm. Các kỹ thuật CASE có thể được phân loại theo khía cạnh chức năng, khía cạnh tiến trình hay khía cạnh tích hợp. Ta lần lượt xét các kỹ thuật CASE được phân loại theo từng khía cạnh này.

Khía cạnh chức năng: Các công cụ CASE được phân loại theo chức năng đặc tả chúng.

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

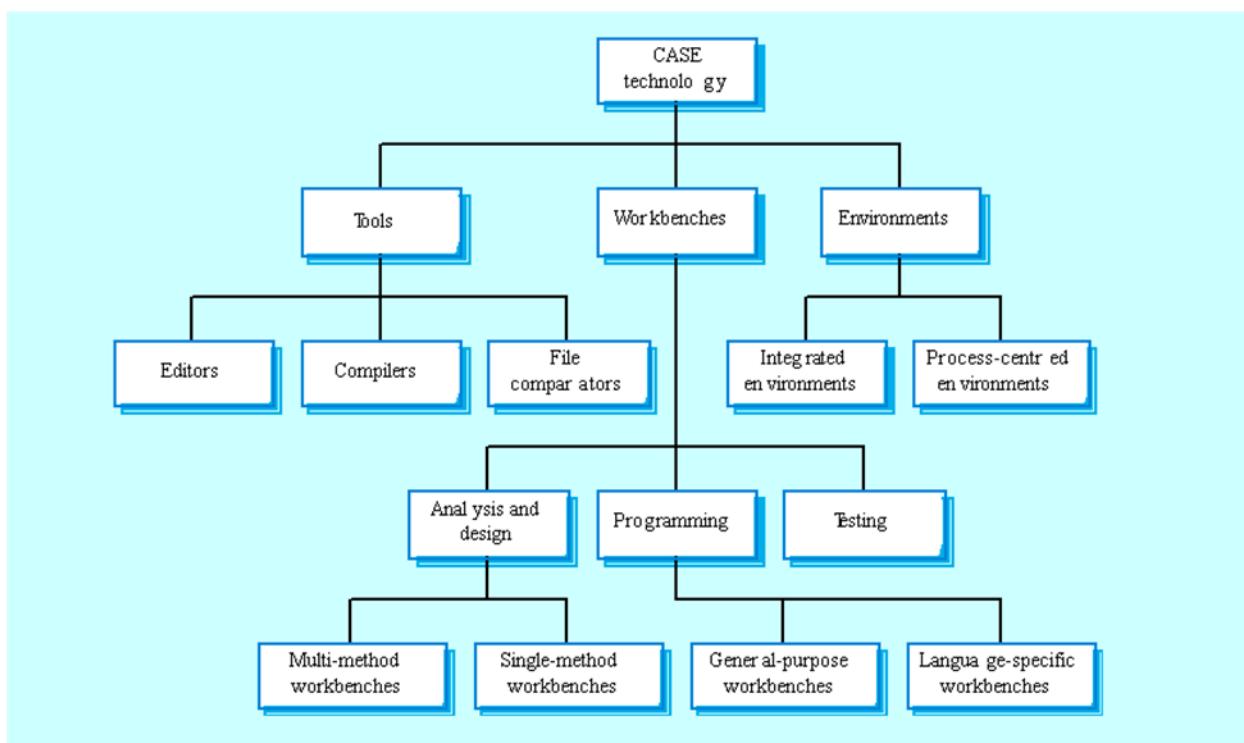
Khía cạnh tiến trình: Phân loại theo các hoạt động tiến trình mà chúng hỗ trợ.



Khía cạnh tích hợp: Phân loại theo sự tổ chức của chúng trong các đơn vị tích hợp bao gồm:

- ✓ **Các công cụ (Tools):** Hỗ trợ các nhiệm vụ đơn lẻ trong tiến trình như công cụ kiểm tra sự phù hợp của thiết kế, bộ soạn thảo văn bản, ...
- ✓ **Các nhóm công cụ (Workbenches):** Hỗ trợ một giai đoạn trong tiến trình như thiết kế, đặc tả, ... chúng thường chứa một số công cụ được tích hợp lại.
- ✓ **Các môi trường (Environments):** Hỗ trợ toàn bộ hoặc một phần đáng kể của tiến trình phần mềm tổng thể. Chúng thường chứa một số workbench được tích hợp lại.

Hình sau mô tả các kỹ thuật CASE được phân loại theo khía cạnh tích hợp:



2.4 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.

2.5 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 thực quản lý tiến trình của tổ chức cao (CMM)

Mô hình thuần thực khả năng (Capability Maturity Model) chỉ ra trong năm mức độ thuần thực cho hoạt động phát triển phần mềm:

(Chụp mô hình, mối quan hệ giữa tiến trình và sản phẩm – Tham khảo cuốn phát triển phần mềm thành công)

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 2

Chương 3 PHÂN TÍCH VÀ ĐẶC TẢ CÁC YÊU CẦU PHẦN MỀM

3.1 Tổng quan về phân tích và đặc tả yêu cầu

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 vận hành và phát triển hệ thống. Các yêu cầu tự chúng mô tả về các dịch vụ và các ràng buộc này. Chúng được phát sinh trong suốt tiến trình kỹ nghệ yêu cầu.

Xác định và đặc tả các yêu cầu là hoạt động cần sự phối hợp của nhà phát triển phần mềm và khách hàng. Nó quyết định chất lượng của phần mềm cần đạt được với chi phí dự kiến và thời hạn cho trước.

3.2 Các yêu cầu phần mềm và mục tiêu

3.2.1 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 mô tả/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 mòi thầu – do đó nó phải dễ hiểu (cần giải thích, cần phát biểu ở mức trừu tượng cao). 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 mâu 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:

- 1 Các yêu cầu người dùng (User requirements)

Là các phát biểu đơn giản, dễ hiểu bằng ngôn ngữ tự nhiên kết hợp các biểu đồ về các dịch vụ mà hệ thống cung cấp và các ràng buộc vận hành nó. Chúng được viết cho các khách hàng.

Các yêu cầu người dùng nên mô tả các yêu cầu chức năng và các yêu cầu phi chức năng theo cách mà người dùng có thể hiểu mà không cần có các kiến thức về kỹ thuật. Chúng được định nghĩa sử dụng ngôn ngữ tự nhiên (NL), các bảng và các biểu đồ sao cho chúng có thể được hiểu bởi người dùng.

2. Các yêu cầu hệ thống (System requirements)

Là các mô tả đủ chi tiết về các dịch vụ mà hệ thống cần cung cấp, các đặc trưng mà hệ thống cần có. Chúng là cơ sở cho thiết kế hệ thống. Tài liệu đặc tả các yêu cầu này có thể được đính kèm với bản hợp đồng về hệ thống giữa người khách hàng và nhà thầu.

Các yêu cầu hệ thống có thể được định nghĩa hoặc minh họa sử dụng các mô hình hệ thống chỉ ra ở chương 4.

Một số ví dụ về các yêu cầu người dùng và yêu cầu hệ thống:

1. Định nghĩa yêu cầu người dùng:

REQ2: “Phần mềm phải cung cấp cơ hội để biểu diễn và truy cập các file bên ngoài mà được tạo bởi các công cụ khác.”

2. Đặc tả yêu cầu hệ thống

REQ2.1: “Người dùng nên được cung cấp các tiện nghi để định nghĩa kiểu file bên ngoài”

REQ2.2: “Mỗi kiểu file bên ngoài có thể được kết hợp với một công cụ mà có thể áp dụng cho file đó”.

REQ2.3: “Mỗi kiểu file bên ngoài có thể được biểu diễn như một biểu tượng trên màn hình hiển thị của người dùng”.

REQ2.4: “Các tiện nghi nên được cung cấp cho biểu tượng biểu diễn kiểu file bên ngoài mà được định nghĩa bởi người dùng”.

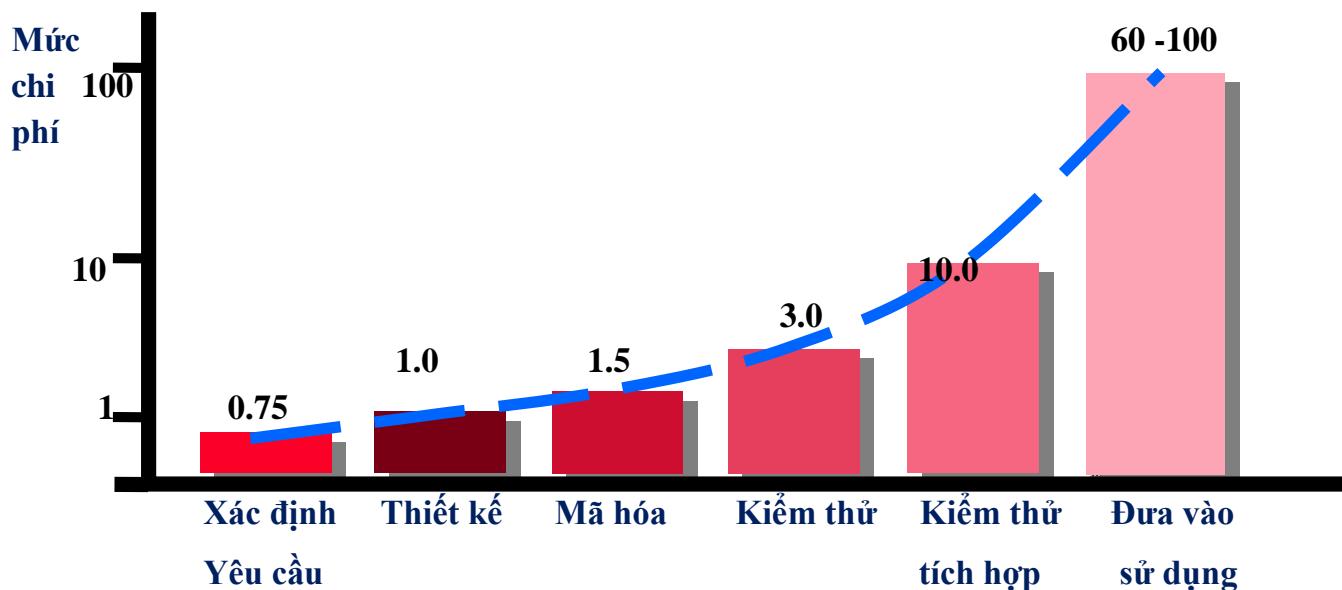
REQ2.5: “Khi người dùng lựa chọn biểu tượng biểu diễn file bên ngoài. Hiệu ứng của việc lựa chọn đó tác động đến công cụ gắn kết với kiểu file đó”.

Các yêu cầu và thiết kế:

Về nguyên tắc, các yêu cầu nên phát biểu những gì hệ thống nên làm và thiết kế nên mô tả cách thực hiện các yêu cầu này. Trong thực tế, các yêu cầu và thiết kế là không thể tách rời nhau vì:

- ✓ Kiến trúc hệ thống có thể được thiết kế để cấu trúc các yêu cầu;
- ✓ Hệ thống có thể tương tác bên trong với các hệ thống khác mà sinh ra các yêu cầu thiết kế;
- ✓ Sử dụng một thiết kế cụ thể có thể là yêu cầu miền ứng dụng.

Các yêu cầu cần được xác định đầy đủ, chính xác, không mâu thuẫn trong pha phân tích và xác định yêu cầu vì giá phải trả cho việc tìm và sửa các lỗi liên quan đến yêu cầu trong trễ trong các giai đoạn phát triển các già rất cao. Hình... sau đây là một minh chứng.



Những người đọc các kiểu yêu cầu gồm:

Các yêu cầu người dùng

- ✓ Những nhà quản lý khách hàng
- ✓ Những người dùng cuối hệ thống
- ✓ Các kỹ sư của khách hàng
- ✓ Những nhà quản lý hợp đồng
- ✓ Kiến trúc sư hệ thống

Các yêu cầu hệ thống

- ✓ Những người dùng cuối hệ thống
- ✓ Các kỹ sư của khách hàng
- ✓ Những người phát triển phần mềm
- ✓ Kiến trúc sư hệ thống

Đặc tả thiết kế phần mềm

- ✓ Các kỹ sư của khách hàng
- ✓ Những người phát triển phần mềm
- ✓ Kiến trúc sư hệ thống

Các yêu cầu viết ra cần phù hợp với mọi đối tượng có thể đọc và hiểu chúng.

3.2.1 Phân loại các yêu cầu phần mềm.

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

-
- + Yêu cầu chức năng (functional Requirements);
 - + Yêu cầu phi chức năng (non function Requirements);
 - + Yêu cầu miền ứng dụng (Domain Requirements);

Tuy nhiên, trong thực tế chúng ta rất khó phân biệt được ba loại yêu cầu này một cách rõ ràng. Trong chương này, chúng ta sẽ tìm cách phân biệt, xác định và đặc tả chúng. Để lấy ví dụ minh họa cho các yêu cầu chúng ta xét phần mềm hệ thống thư viện. Hệ thống này được mô tả như sau:

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.

a. 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.

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(doc) tài liệu, download, in tài liệu,...”

REQ5: “Tất cả những hóa đơn mà người sử dụng đăng ký để in sao tài liệu có một mã duy nhất.”

b. Yêu cầu phi chức năng

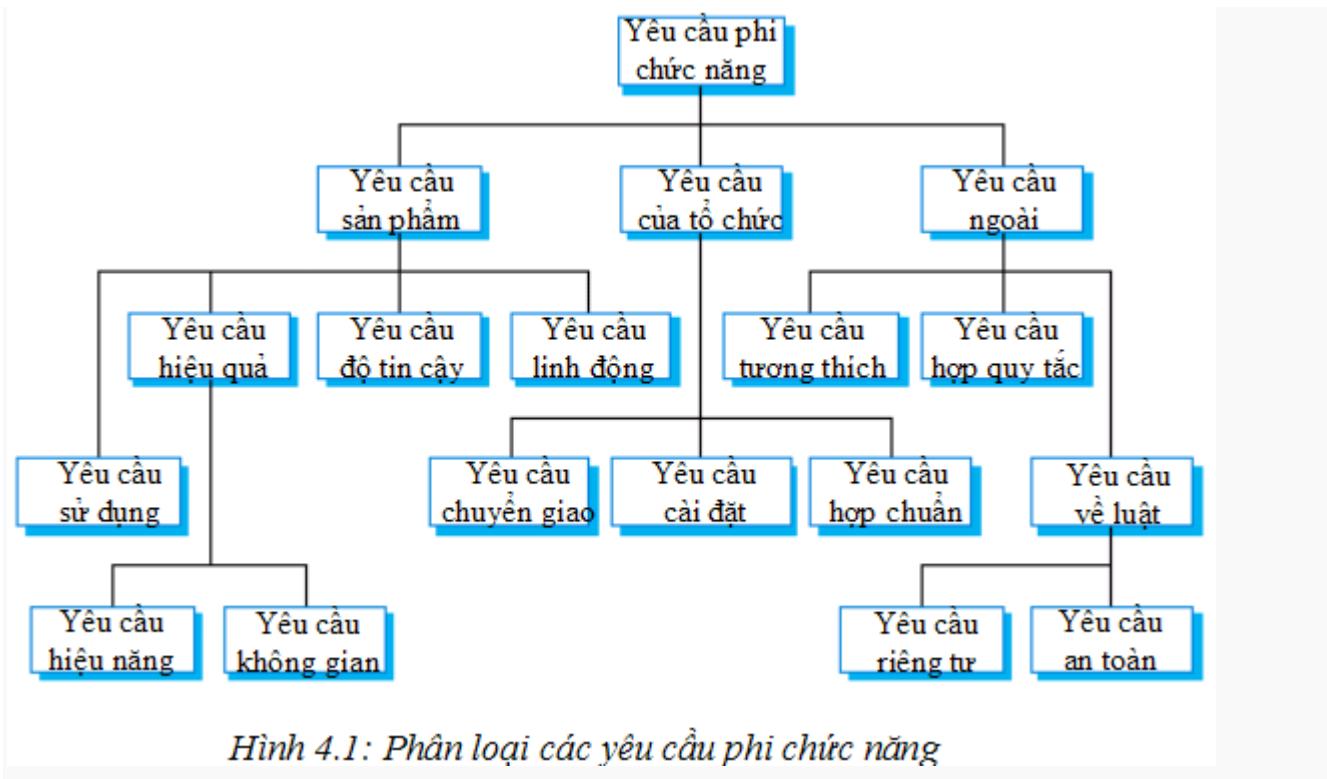
Các yêu cầu phi chức năng định nghĩa các ràng buộc lên các dịch vụ hoặc các chức năng được yêu cầu bởi hệ thống như các ràng buộc về mặt thời gian, các ràng buộc về tiến trình phát triển, các chuẩn, ...

Các yêu cầu phi chức năng không đề cập trực tiếp tới các chức năng/dịch vụ cụ thể của hệ thống. Chúng thường định nghĩa các thuộc tính của hệ thống như độ tin cậy, thời gian đáp ứng, các yêu cầu về lưu trữ ... của hệ thống phần mềm. Chúng cũng có thể định nghĩa các ràng buộc của hệ thống phần mềm như ràng buộc về thiết bị vào/ra, về giao diện ... của hệ thống. Một số yêu cầu phi chức năng còn có liên quan đến quy trình xây dựng hệ thống. Ví

dụ các chuẩn cần được sử dụng trong tiến trình phát triển/bảo trì phần mềm, các công cụ CASE cần sử dụng, ngôn ngữ lập trình nào sẽ được dùng, ...

Các yêu cầu phi chức năng xuất hiện là do yêu cầu của người sử dụng, các ràng buộc về ngân sách, các chính sách của tổ chức sử dụng hệ thống, các 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

Nhìn chung 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 có thể sẽ không sử dụng được hoặc không tiện dụng. Trong một số trường hợp, các yêu cầu phi chức năng còn quan trọng hơn các yêu cầu chức năng. Ví dụ, nếu xét một hệ thống lái máy bay rất lớn và phức tạp, cho dù được trang bị đầy đủ các dịch vụ nhưng nếu không thoả mãn yêu cầu về độ tin cậy và không được phê chuẩn là an toàn để vận hành thì hệ thống đó sẽ không được sử dụng trong thực tế. Sau đây là một số ví dụ về các yêu cầu phi chức năng theo từng loại của hệ thống LIBSYS.

1. Yêu cầu về sản phẩm

REQ6: “LIBSYS nên được cài đặt đơn giản bằng HTML mà không có frame hoặc Java applets.” - Yêu cầu về khả năng sử dụng

2. Yêu cầu về tổ chức

REQ7: “Quy trình phát triển hệ thống và các tài liệu phát hành phải phù hợp với tiến trình và các phát hành được định nghĩa trong chuẩn: XYZCo-SP-STAN-95.

3. Yêu cầu ngoài

REQ8: “Hệ thống không được để lộ các thông tin cá nhân của khách hàng”

Với các yêu cầu phi chức năng, thông thường chúng ta rất khó xác định chính xác và rất khó thẩm tra chúng. Vì người dùng thường quên hoặc phát biểu chúng một cách chung chung, mập mờ. Điều này gây khó khăn cho chúng ta trong việc xác định và định nghĩa chúng một cách chính xác, đầy đủ.

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 để từ đó ta có thể xác định được các yêu cầu phi chức năng. 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 \Rightarrow Đáp ứng yêu cầu. <50 giao dịch/giây \Rightarrow 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.”

Để chuyển các yêu cầu chức năng thành các phát biểu định lượng bạn đọc tham khảo thêm tài liệu [6].

c. Các yêu cầu miền ứng dụng

Các yêu cầu miền ứng dụng là các yêu cầu滋生 từ miền ứng dụng của hệ thống. Chúng phản ánh các thuộc tính và ràng buộc của miền ứng dụng. Nó có thể là yêu cầu chức năng hoặc phi chức năng. Yêu cầu miền ứng dụng đóng vai trò quan trọng, nếu các 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.

Ví dụ: xét hệ thống bảo vệ tàu thuyền:

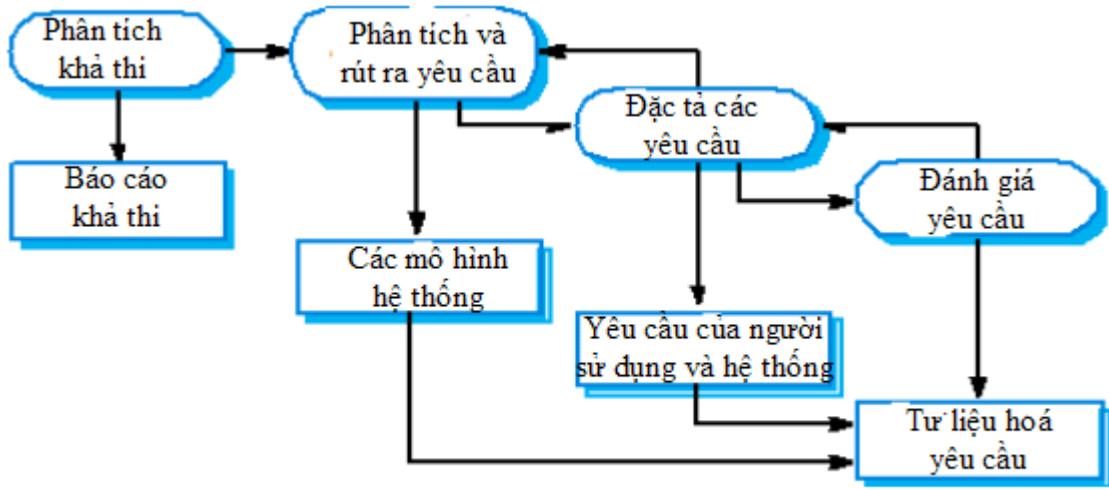
REQ12: “Sự giảm tốc độ của tàu được tính như sau:

$$D_{train} = D_{control} + D_{gradient}$$

*Trong đó: $D_{gradient}$ is $9.81ms^2 * compensated gradient/alpha$ và các giá trị $9.81ms^2/alpha$ là khác nhau với những loại tàu thuyền khác nhau.”*

3.3 Tiết trình kỹ nghệ yêu cầu

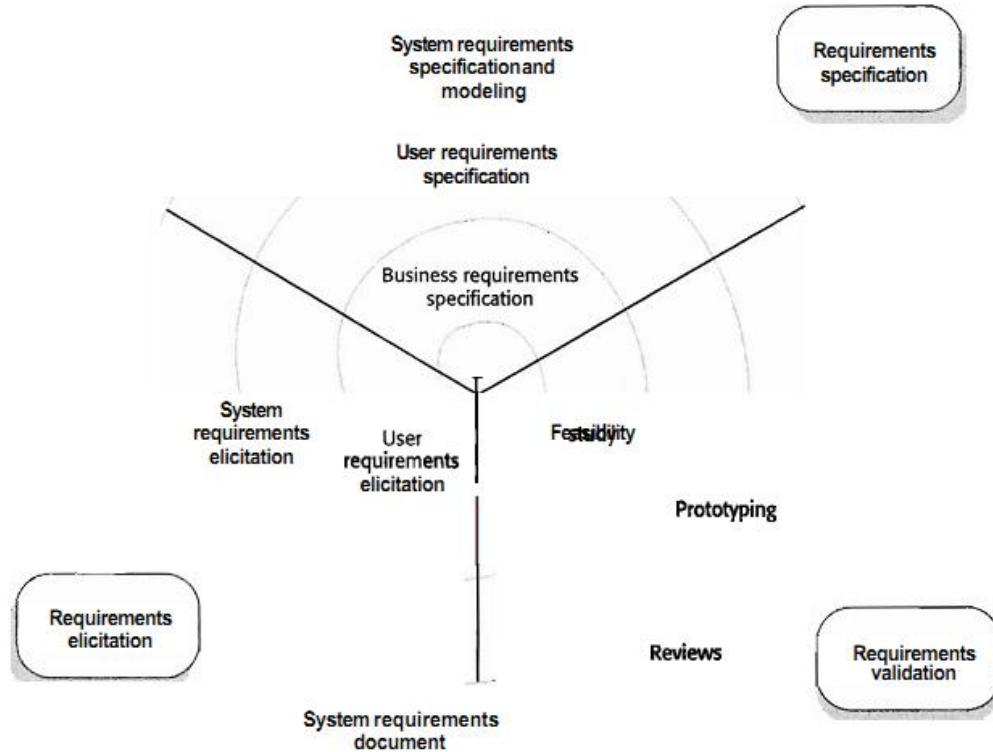
Tiết trình kỹ nghệ yêu cầu nhằm phân tích xác định và đặc tả mọi yêu cầu đặt ra đối với hệ thống một cách đầy đủ, chính xác. Tiết trình này được bao gồm các giai đoạn được mô tả như hình 2.6



Hình 2.6: Quy trình xác định yêu cầu

Trong tiến trình RE, ba hoạt động sau là ba hoạt động được lặp lại nhiều lần, mỗi lần lặp thêm dần tính hình thức/tính chi tiết và độ chính xác. Tiến trình này còn được biểu diễn dưới dạng mô hình xoắn ốc (hình dưới). Ở đó, các yêu cầu được phát triển với các mức chi tiết khác nhau ở những vòng lặp khác nhau. Số lượng vòng lặp xoắn ốc có thể thay đổi, vòng xoắn ốc dừng khi tất cả các yêu cầu của người dùng đều được chắt lọc đầy đủ.

Figure 7.2 Spiral model of requirements engineering processes



3.3.1 Phân tích tính khả thi của dự án

Khi phát triển mới các hệ thống phần mềm, tiến trình kỹ nghệ yêu cầu thường bắt đầu bằng việc phân tích khả thi của dự án. Mục đích của nghiên cứu tính khả thi là đi đến kết luận “có

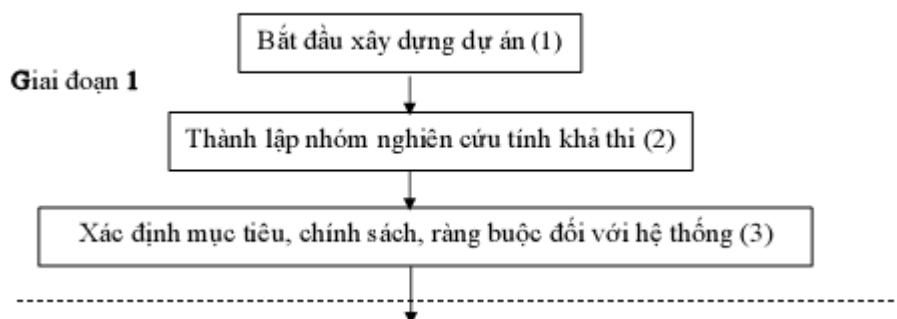
nên phát triển hệ thống hay không?”. Hoạt động này nếu xét về mặt công nghệ nhằm phân tích xác định xem liệu dự án có thể được triển khai với một ngân sách và công nghệ cho trước hay không, về mặt kinh tế nhằm chỉ ra rằng hệ thống cần phát triển có mang lại lợi nhuận cho người phát triển chúng không. Nội dung nghiên cứu tính khả thi của dự án tập trung trả lời các câu hỏi sau:

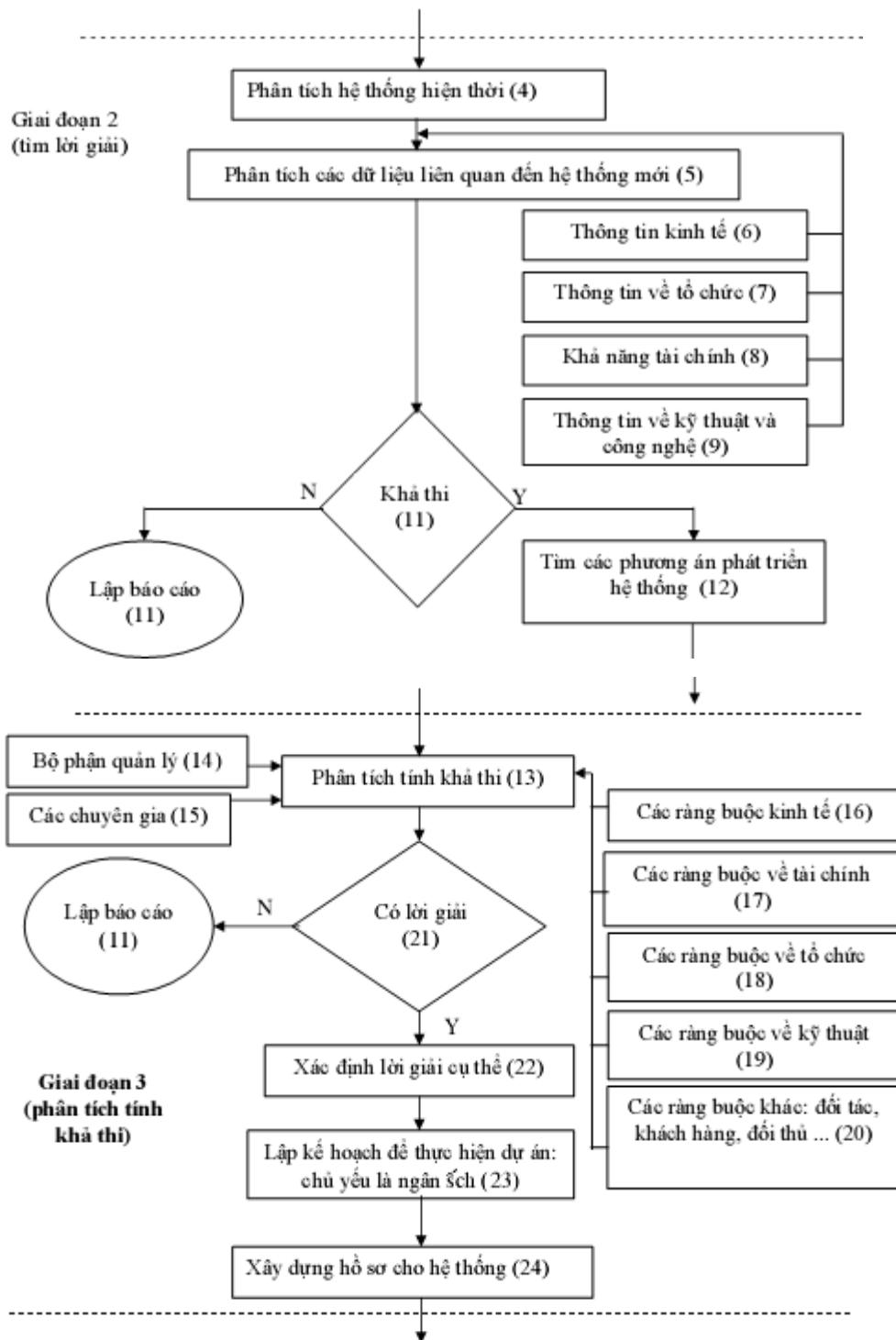
- ✓ Hệ thống được xây dựng sẽ giúp gì cho tổ chức?
- ✓ Hệ thống sử dụng công nghệ nào, kinh phí bao nhiêu, thời gian bao lâu?
- ✓ Hệ thống cần phải tích hợp với các hệ nào đang sử dụng?

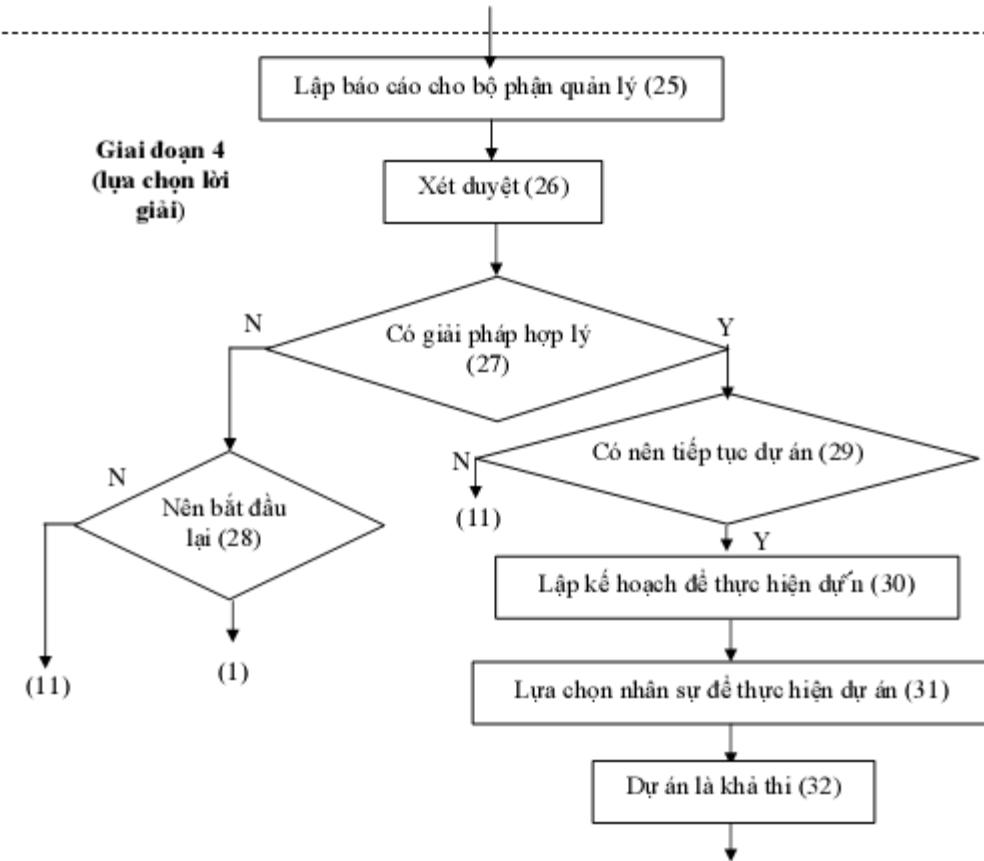
Thông tin đầu vào để phân tích khả thi là các yêu cầu nghiệp vụ, mô tả sơ bộ về hệ thống, cách thức hệ thống hỗ trợ các yêu cầu nghiệp vụ. Kết quả của việc phân tích khả thi là một báo cáo quyết định có nên phát triển hệ thống phần mềm được đề xuất hay không. Hoạt động này nên tiến hành một cách nhanh chóng và không quá tốn kém

Hoạt động phân tích tính khả thi có thể được thực hiện theo các cách khác nhau. Sau đây là thuật toán phân tích tính khả thi được sử dụng cho một số dự án tin học. Thuật toán gồm 4 giai đoạn như sau:

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







3.3.2 Phân tích và rút ra các yêu cầu

Phân tích và rút ra các yêu cầu còn được gọi là xác định các yêu cầu của hệ thống phần mềm cần xây dựng. Trong pha này, người dùng, kỹ sư, nhà quản lý, chuyên gia miền... cùng hợp tác để làm rõ:

- ✓ Phạm vi lĩnh vực ứng dụng
- ✓ Các dịch vụ mà hệ thống cần cung cấp
- ✓ Các ràng buộc áp đặt nên hoạt động của hệ thống

Bằng cách xây dựng các mô hình phân tích (mô hình nghiệp vụ của hệ thống) để làm rõ các yêu cầu trên. Nhiệm vụ của nhà phân tích là gợi mở, xác định đúng, đầy đủ, chính xác các yêu cầu của hệ thống.

Từ đây, xuất hiện một khái niệm mới mà chúng ta cần làm quen đó là Stakeholder. Stakeholder là những người tham dự vào dự án xây dựng hệ thống, những người này

có thể là người sử dụng cuối, người quản lý, kỹ sư bảo trì phần cứng & phần mềm, chuyên gia lĩnh vực, ...

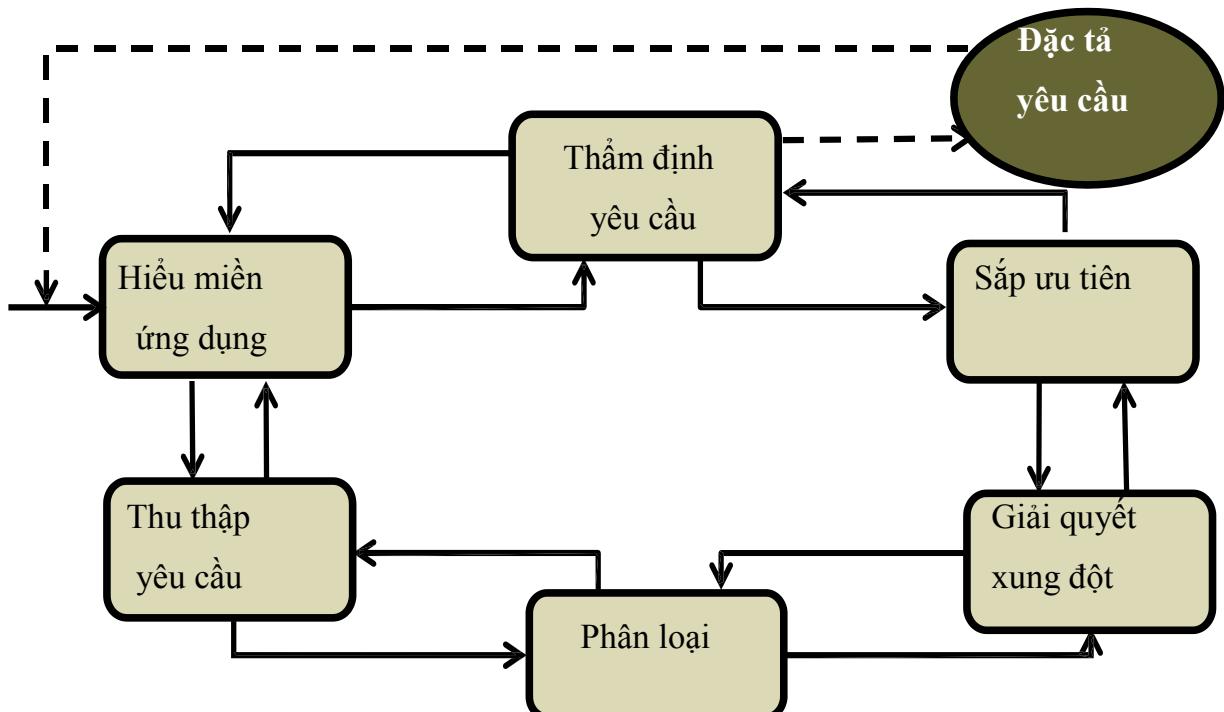
Ví dụ, trong hệ thống ATM gồm các Stakeholder sau: khách hàng của ngân hàng, đại diện của các ngân hàng khác, người quản lý ngân hàng, nhân viên ngân hàng, quản trị CSDL, quản lý bảo mật, phòng marketing, kỹ sư bảo trì phần cứng và phần mềm, người điều hành ngân hàng.

Thực tế cho thấy, việc phát hiện và tìm hiểu yêu cầu của stakeholder thường gây khó khăn cho chúng ta 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, để khắc phục các nguyên nhân này, hoạt động phân tích yêu cầu là hoạt động cần được thực hiện lặp lại nhiều lần và có sự tham gia đánh giá của các loại Stakeholder liên quan.

Tiến trình phân tích yêu cầu được biểu diễn như hình...



yêu cầu

a.Các nguyên lý cần quan tâm khi phân tích các yêu cầu hệ thống

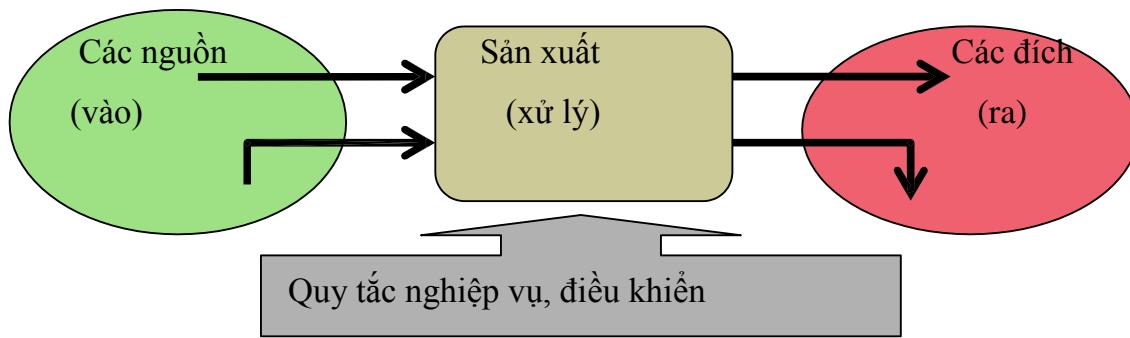
1. *Mô hình hóa miền thông tin*: Phải hiểu và biểu diễn được miền thông tin (problem domain):

- ✓ Xác định được các thực thể dữ liệu (các đối tượng)
- ✓ Các định được các thuộc tính của chúng
- ✓ Thiết lập được các mối quan hệ giữa các thực thể dữ liệu.

2. *Mô hình hóa chức năng*: Bản chất của phần mềm là biến đổi thông tin, do đó, ta cần:

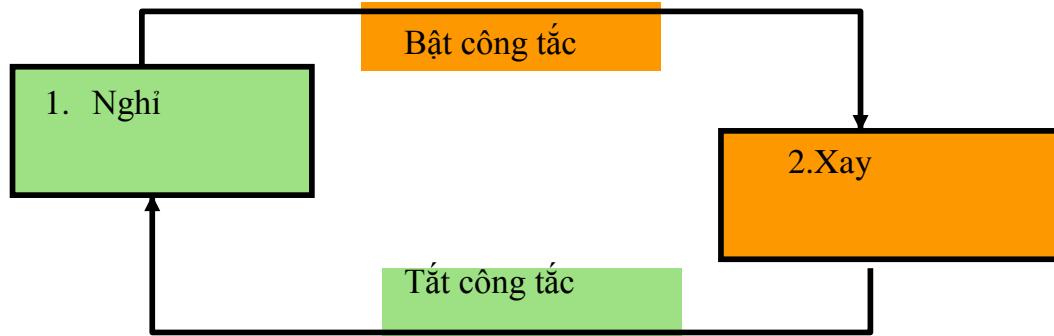
- ✓ Xác định được các chức năng (dùng để biến đổi thông tin),
- ✓ Xác định cách thức dữ liệu (thông tin) di chuyển trong hệ thống (luồng dữ liệu)
- ✓ Xác định được các tác nhân tạo dữ liệu (nguồn) và tác nhân tiếp nhận dữ liệu (đích)

Mô hình chung của mọi quá trình sản xuất được mô tả như sau:



3. *Mô hình hóa hành vi*: Phần mềm (hệ thống) có trạng thái (hành vi) do đó, chúng ta cần:

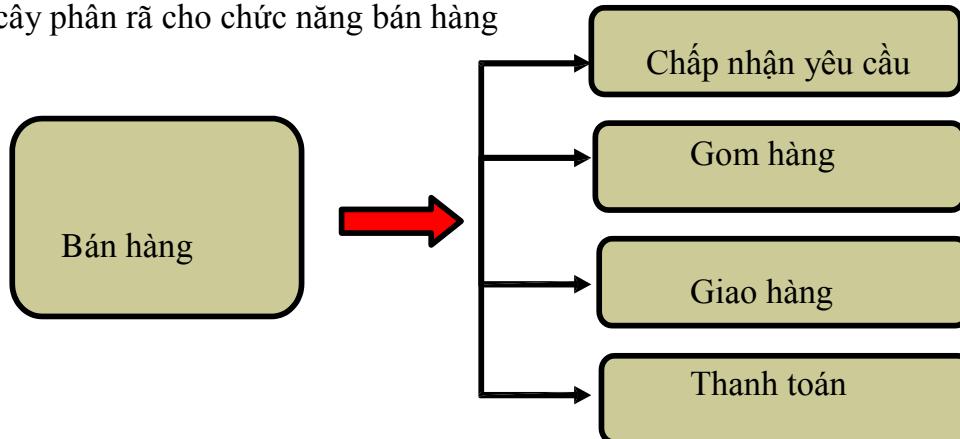
- ✓ Xác định các trạng thái của hệ thống, ví dụ máy xay cà phê có 2 trạng thái: nghỉ, làm việc.
- ✓ Xác định các sự kiện làm thay đổi hành vi hệ thống. Ví dụ các sự kiện làm thay đổi trạng thái của máy say cà phê gồm bật công tắc, tắt công tắc. Hành vi của máy được mô hình hóa như hình...



4. Phân hoạch, làm mịn: làm mịn, phân hoạch và biểu diễn các mô tả ở các mức chi tiết khác nhau:

- ✓ Làm mịn các mô hình dữ liệu
- ✓ Tạo cây (biểu đồ) phân rã chức năng.
- ✓ Biểu diễn hành vi ở các mức chi tiết khác nhau.

Ví dụ: Tạo cây phân rã cho chức năng bán hàng



Khi phân tích yêu cầu, chúng ta cần nhìn vấn đề từ bên ngoài và thấy được bản chất của vấn đề, tức là cần xác định rõ hệ thống làm gì (chức năng) dưới những điều kiện, ràng buộc gì mà không cần quan tâm đến cách thức cài đặt (là quá trình vật lý) như làm như thế nào, khi nào làm, làm ở đâu, ...

b. Các phương pháp thu thập yêu cầu:

1. Khung nhìn (Viewpoint)

Kỹ thuật này dựa trên cơ sở mỗi stakeholder liên quan đến dự án có những hiểu biết và nhu cầu hiểu biết nhất định về dự án. Do đó khi trình bày các vấn đề về hệ thống

nên phụ thuộc vào đối tượng trình bày. Khung nhìn là cách xây dựng các yêu cầu về một vấn đề 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, gặp gỡ một số lượng ít người, thông thường từ một đến 2 người tại một địa điểm cụ thể vào khoảng thời gian xác định để đạt được 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. Các yêu cầu sẽ được lấy ra từ những câu trả lời của stakeholder. Khi phỏng vấn người phỏng vấn thường sử dụng hai kiểu câu hỏi:

- ✓ *Câu hỏi đóng*: là câu hỏi đã được định nghĩa trước và có nhiều đáp án để Stakeholder lựa chọn trả lời. Ví dụ: “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 các câu hỏi 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 Stakeholder và cách họ tương tác với hệ thống như thế nào hay nói cách khác là đạt được các mục tiêu đặt ra của buổi phỏng vấn.

Để phỏng vấn thành công, người phỏng vấn nên tiến hành các bước 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
- ✓ Khi tiến hành phỏng vấn cần chuẩn bị trước và tiến hành như sau:
 - 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 một cách phù hợp nhất
 - 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
- Thông nhất lại ngày lấy lại bản hiệu chỉnh
- Xác nhận lại lịch làm việc

Phỏng vấn thường chia làm 2 loại phỏng vấn hình thức và phi hình thức. 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 vào 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.

3. Kịch bản

Phương pháp kịch bản này sinh từ cơ sở: Khi tiếp xúc với một vấn đề, chúng ta thường hiểu một vấn đề đó thông qua các ví dụ thực tế (thực hành) dễ dàng hơn là thông qua những mô tả trừu tượng về nó (lý thuyết).

Sử dụng kịch bản để phát hiện ra các yêu cầu hệ thống, nghĩa là ta tạm thời đóng vai trò là người sử dụng hệ thống/Stakeholder người dùng cuối để làm công việc của họ.

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. 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 giới hạn về thời gian có thể làm tối thiểu hóa vấn đề.Thêm vào đó, người kỹ sư phần mềm có thể có thiên kiến về quá trình xử lý công việc, các công việc cần 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 hoặc ghi băng, ghi hình các hoạt động, các bước xử lý công việc. Kết quả này sẽ được dùng để phân tích xác định các sự kiện, các hoạt động, các công việc, các lý do tiến hành công việc, 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 quy trình cũng như cách thức tiến hành các 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 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 ở đây muốn nói tới các cảm nang, quy định, các thao tác chuẩn mà tổ chức cung cấp để hướng dẫn quy trình và cách thức triển khai công việc 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. Các tài liệu đưa ra các thông tin khách quan, chính xác.

6. Xem xét phần mềm

Khi các phần mềm ứng dụng cũ phải được thay thế bởi 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, các ràng buộc trong thiết kế tương lai của phần mềm. Nhược điểm chính của việc chắt lọc thông tin từ việc quan sát phần mềm là các tài liệu của nó có thể không chính xác hoặc không kịp thời.Thêm vào đó, thời gian có thể bị lãng phí nếu ứng dụng đang bị xoá bỏ.

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 này.

3.3.3 Đặc tả yêu cầu

Đặc tả các yêu cầu phần mềm là hoạt động mô tả cụ thể các yêu cầu trong tài liệu đặc tả. Yêu cầu nên được mô tả ở các mức độ trừu tượng khác nhau, đầy đủ, chính xác dần để nhiều đối tượng người đọc có thể hiểu được chúng.

Tài liệu đặc tả yêu cầu chứa những yêu cầu chính thức về những gì cần phải thực hiện bởi đội phát triển hệ thống. Tài liệu này nên chứa tất cả các định nghĩa về yêu cầu của người sử dụng và đặc tả yêu cầu hệ thống. Nó 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 nghiêm 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.

Ví dụ về sự mập mờ trong việc sử dụng các thuật ngữ để phát biểu yêu cầu:

Xét thuật ngữ ‘appropriate viewers’:

- ✓ *Ý định của người dùng – Hiển thị với mục đích cụ thể cho mỗi kiểu tài liệu khác nhau;*
- ✓ *Người phát triển hiểu – Cung cấp một hiển thị văn bản mà chỉ ra các nội dung của tài liệu.*

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

a. 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 gấp 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 khâu trù.”

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

“Các tiện nghi của lưỡi: Trợ giúp trong việc đặt vị trí của các thực thể trong một biểu đồ, người dùng có thể bật lưỡi ở chế độ centimet hoặc inch, thông qua việc chọn trên panel điều khiển. Ban đầu, lưỡi ở chế độ tắt. Lưỡi 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 lưỡi sẽ cung cấp khung nhìn có thể vẫn nhỏ nhưng số dòng lưỡi sẽ bị ẩn đi để tránh lắp kín biểu đồ nhỏ hơn bởi các dòng lưỡi 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 lưỡi soạn thảo)
- Yêu cầu phi chức năng (các đơn vị lưỡi).
- Yêu cầu giao diện người dùng phi chức năng (sự bật, tắt lưỡi).
=> đây chính là sự lộn xộn giữa các loại yêu cầu

b. 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 lưỡi

“Bộ soạn thảo sẽ cung cấp tiện ích của lưỡi ở đó một ma trận gồm các dòng và cột cung cấp nên cho cửa sổ soạn thảo. Lưỡi sẽ là lưỡi bị động, ở đó sự cẩn chỉnh các thực thể là trách nhiệm của người dùng.

- ✓ **Lý do cơ bản:** Lưỡi giúp người dùng tạo các biểu đồ có trật tự/gọn gang với các thực thể có không gian tốt. Khi lưỡi được kích hoạt, ở đó các thực thể che đi các dòng lưỡi 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
- ✓ **Nguồn gốc:** Ray Wilson Glasgow Office. ”

* Một số hướng dẫn cho việc viết các yêu cầu:

- Đưa ra một form chuẩn và sử dụng nó cho mọi yêu cầu
- Sử dụng ngôn ngữ một cách thống nhất. Ví dụ: Sử dụng **shall** cho các yêu cầu bắt buộc, sử dụng **should** cho các yêu cầu mong muốn.
- Sử dụng văn bản nổi bật để xác định những phần yêu cầu chính.
- Tránh sử dụng những thuật ngữ máy tính.

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 hệ 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.

c. 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 bằng 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 lè 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 ($((r2-r1) \square (r1-r0))$)	CompDose=round (($r2-r1$)/4) If rounded result = 0 then CompDose=Minimum Dose

d. Các ngôn ngữ mô tả thiết kế

Các 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.

e. Các kỹ 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 dùng các mô tả Use Case và các biểu đồ trình tự.

Một số dạng biểu diễn:

Các mô hình đồ thị:

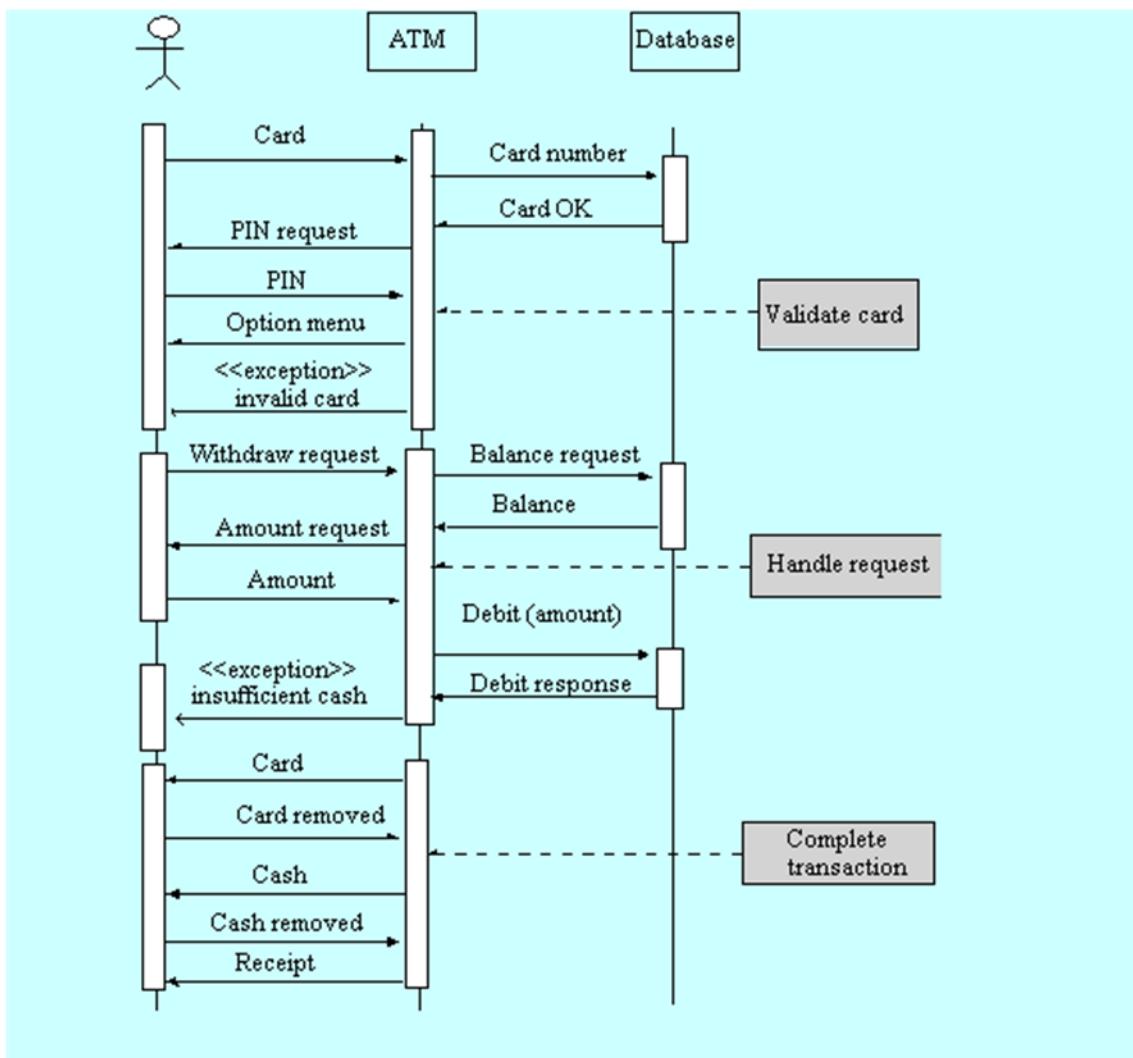
Các mô hình đồ thị hữu ích nhất khi bạn cần chỉ ra các thay đổi trạng thái như thế nào hoặc khi bạn cần mô tả một chuỗi các hoạt động. Các mô hình đồ thị được trình bày trong chương 4.

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

- Thảm định thẻ;
- Xử lý yêu cầu;
- Hoàn thành giao dịch.



f. 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.

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 tượng 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

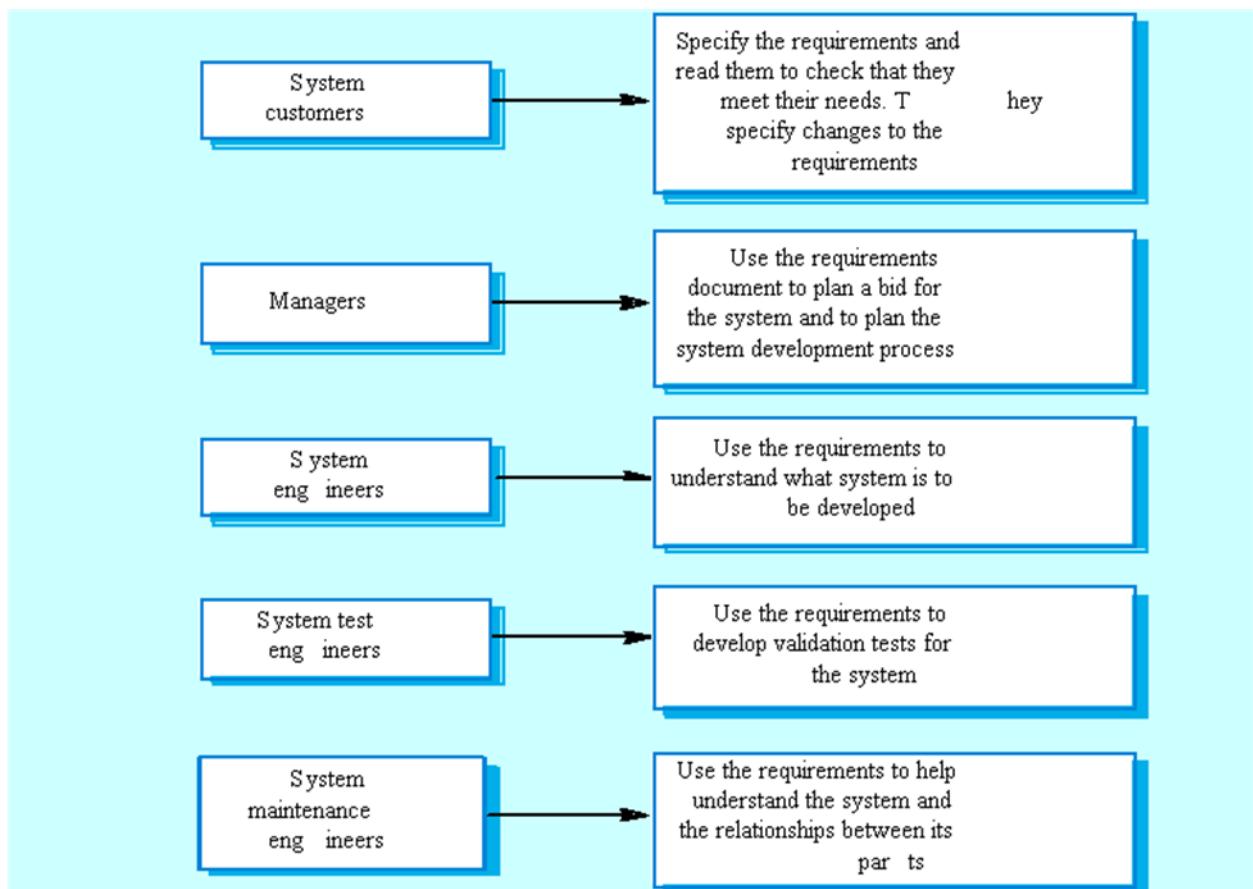
3.2.2 Các ràng buộc của hệ thống

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:



3.3.4. Đánh giá yêu cầu

Đánh giá yêu cầu là việc xét duyệt lại các yêu cầu đã được định nghĩa tài liệu đặc tả. 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 đóng vai trò rất quan trọng. Trong quá trình đánh giá yêu cầu, chúng ta phải kiểm tra các yêu cầu ở những khía cạnh sau:

- ✓ *Hợp lệ:* Nhằm trả lời câu hỏi: Các yêu cầu của người sử dụng có được hỗ trợ tốt nhất bởi các chức năng của hệ thống phần mềm hay không?
- ✓ *Nhất quán:* Nhằm trả lời câu hỏi: “Có yêu cầu nào xung đột nhau hay không?”.

- ✓ *Hoàn thiện*: Nhằm trả lời câu hỏi: “Tất cả các yêu cầu của khách hàng đã được xác định đầy đủ chưa?”
- ✓ *Hiện thực*: Nhằm trả lời câu hỏi: “Các yêu cầu có thể được cài đặt với một ngân sách và công nghệ cho trước hay không?”
- ✓ *Xác thực*: Nhằm trả lời câu hỏi: “Các yêu cầu có thể được thẩm định hay đánh giá hay không?”

Để đánh giá các yêu cầu, ta có thể dùng một số kỹ thuật đánh giá sau:

- ✓ *Xem xét lại các yêu cầu*: phân tích các yêu cầu một cách hệ thống, lấy ý kiến khách hàng và nên được tiến hành thường xuyên để phát hiện sớm các lỗi.
- ✓ *Dung mẫu thử*: Sử dụng các mô hình hệ thống để kiểm tra tính thực hiện được của các yêu cầu
- ✓ *Tạo ra các ca kiểm thử*: Nhằm kiểm tra tính kiểm thử được của các yêu cầu
- ✓ *Sử dụng CASE*: nhằm kiểm tra tính nhất quán của các yêu cầu

Các kỹ thuật này có thể sử dụng riêng lẻ hoặc kết hợp chúng lại để đánh giá các yêu cầu.

3.3.5. Quản lý các yêu cầu

Quản lý yêu cầu là quy trình quản lý các yêu cầu và những thay đổi của từng yêu cầu trong quá trình phát triển và bảo trì hệ thống phần mềm.

Thực tế cho thấy, các yêu cầu thường không đầy đủ và không đồng nhất 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 đang phát triển/bảo trì.
- ✓ 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 cũng thay đổi trong suốt quá trình phát triển/bảo trì hệ thống.
- ✓ Môi trường nghiệp vụ và môi trường kỹ thuật của hệ thống cũng thay đổi trong suốt quy trình phát triển/bảo trì hệ thống.

Quy trình lập kế hoạch quản lý yêu cầu:

- Xác định yêu cầu: Làm thế nào để xác định được từng yêu cầu của phần mềm
- Quản lý thay đổi: Xác định các hoạt động tiếp theo khi yêu cầu thay đổi. Các chính sách tìm vết thông tin về mối quan hệ giữa các yêu cầu. Những mối quan hệ này cần phải được lưu giữ.
- Xác định CASE tool hỗ trợ: 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.

Để quản lý được các thay đổi đối với mỗi yêu cầu, ta cần tiến hành như sau:

- ✓ Phân tích các vấn đề liên quan đến yêu cầu và đặc tả thay đổi: thảo luận về các vấn đề nảy sinh từ 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 lên 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.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 3

Đọc thêm cuốn *hiểu thấu về CMMI* để bổ sung thêm vào bài giảng CNPM @

Chương 4 CÁC MÔ HÌNH HỆ THỐNG

4.1 Giới thiệu

Các yêu cầu của người sử dụng thường được viết bằng ngôn ngữ tự nhiên để những người không có kiến thức về mặt kỹ thuật có thể hiểu được nó. Tuy nhiên, những yêu cầu hệ thống chi tiết phải được mô hình hóa. Mô hình hóa hệ thống giúp cho người phân tích hiểu rõ các chức năng của hệ thống. Ta có thể sử dụng các mô hình khác nhau để biểu diễn hệ thống từ nhiều khía cạnh khác nhau. Trong chương này, chúng ta sẽ tìm hiểu về một số mô hình hóa hệ thống. Mục tiêu của chương này:

- ✓ Giải thích tại sao ngữ cảnh của hệ thống nên được mô hình hóa như một phần của tiến trình RE.
- ✓ Mô tả mô hình hóa hành vi, mô hình hóa dữ liệu và mô hình hóa đối tượng.
- ✓ Giới thiệu một số ký pháp được sử dụng trong Unified Modeling Language (UML)
- ✓ Chỉ ra cách thức CASE workbenches hỗ trợ việc mô hình hóa hệ thống.

4.2 Mô hình hóa hệ thống

Mô hình hóa hệ thống giúp người phân tích hiểu chức năng của hệ thống và các mô hình được sử dụng để giao tiếp với khách hàng. Các mô hình khác nhau biểu diễn hệ thống ở các khía cạnh khác nhau:

- ✓ Khía cạnh bên ngoài chỉ ra ngữ cảnh của hệ thống và môi trường của nó.
- ✓ Khía cạnh hành vi, chỉ ra hành vi của hệ thống;
- ✓ Khía cạnh cấu trúc chỉ ra kiến trúc hệ thống hoặc kiến trúc ứng dụng.

Các kiểu mô hình:

- ✓ **Mô hình xử lý dữ liệu:** chỉ ra dữ liệu được xử lý ở các giai đoạn khác nhau như thế nào.
- ✓ **Mô hình kết cấu:** chỉ ra cách thức các thực thể được cấu tạo từ các thực thể khác như thế nào.
- ✓ **Mô hình kiến trúc:** chỉ ra các hệ thống con.

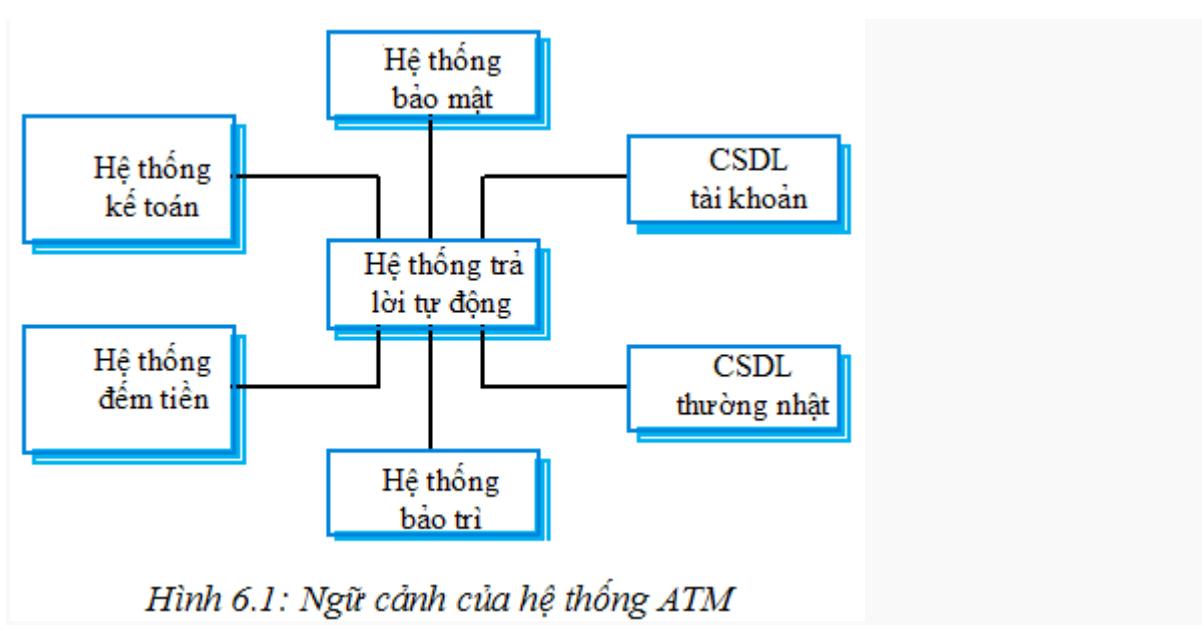
- ✓ **Mô hình phân cấp lớp:** chỉ ra các thực thể có các đặc trưng chung như thế nào.
- ✓ **Mô hình phản ứng/trạng thái:** chỉ ra phản ứng của hệ thống với các sự kiện xảy ra như thế nào.

4.2.1. 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 là gì.

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.

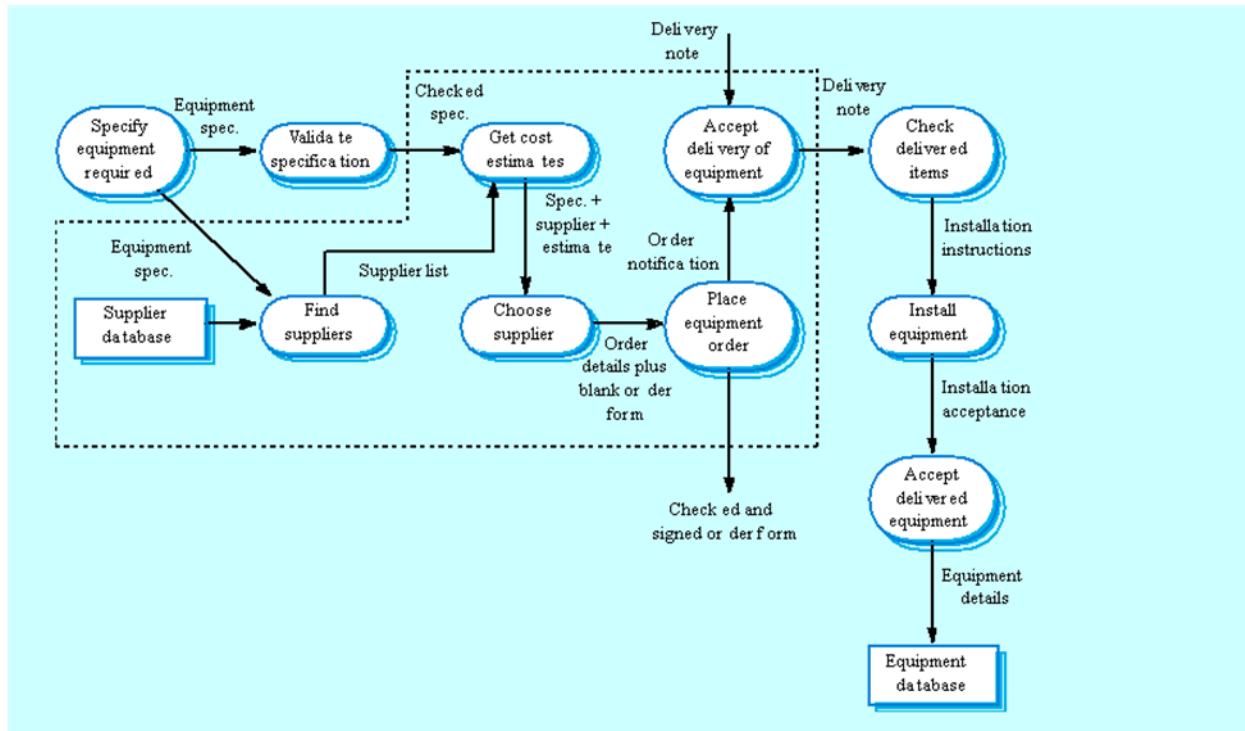
Ví dụ: Mô hình ngữ cảnh của hệ thống ATM



Mô hình ngữ cảnh 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 hỗ trợ cho nó.

Các mô hình tiến trình chỉ ra tiến trình tổng thể và các tiến trình được hệ thống hỗ trợ là gì. 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.

Ví dụ: Xét tiến trình đặt mua trang thiết bị phần mềm



4.2. 2 Mô hình ứng xử

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

- ✓ Mô hình luồng dữ liệu: biểu diễn tiến trình xử lý dữ liệu trong hệ thống
- ✓ Mô hình máy trạng thái: biểu diễn các trạng thái của hệ thống biến đổi khi một sự kiện xảy ra.

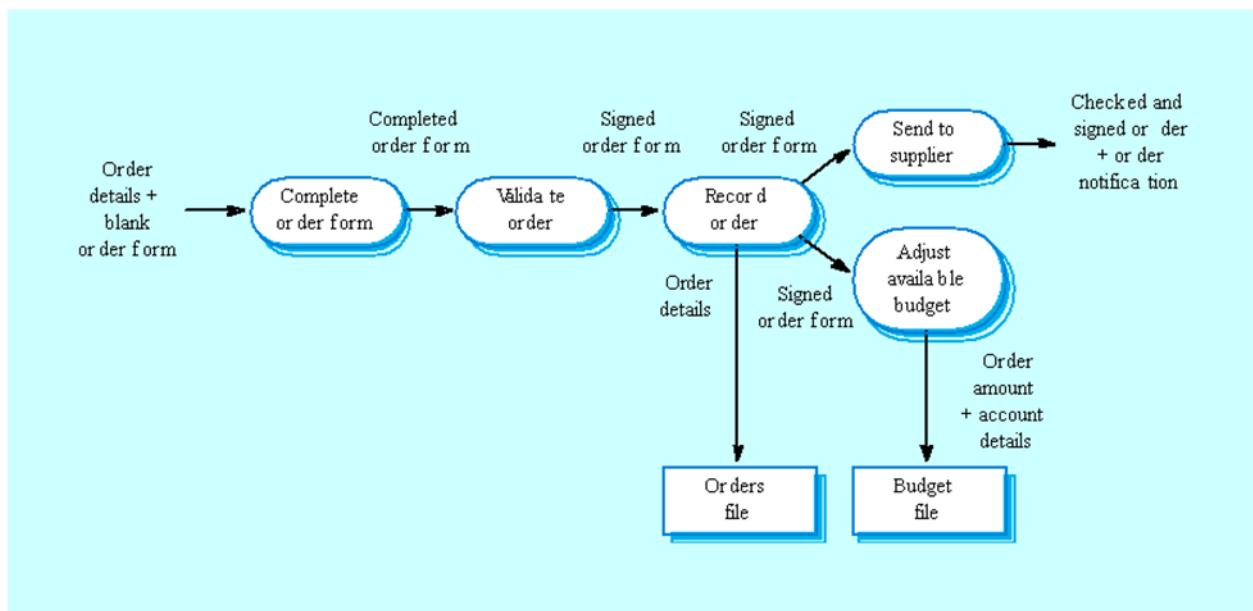
Hai mô hình này biểu diễn những góc nhìn khác nhau, nhưng cả hai đều cần thiết để mô tả hành vi của hệ thống.

4.2.1. Mô hình luồng dữ liệu (Data Flow Diagrams - DFDs)

Mô hình luồng dữ liệu được sử dụng để mô hình hóa quy trình xử lý dữ liệu của hệ thống. Nó chỉ ra các bước xử lý khi dữ liệu chảy qua hệ thống từ điểm đầu tới điểm

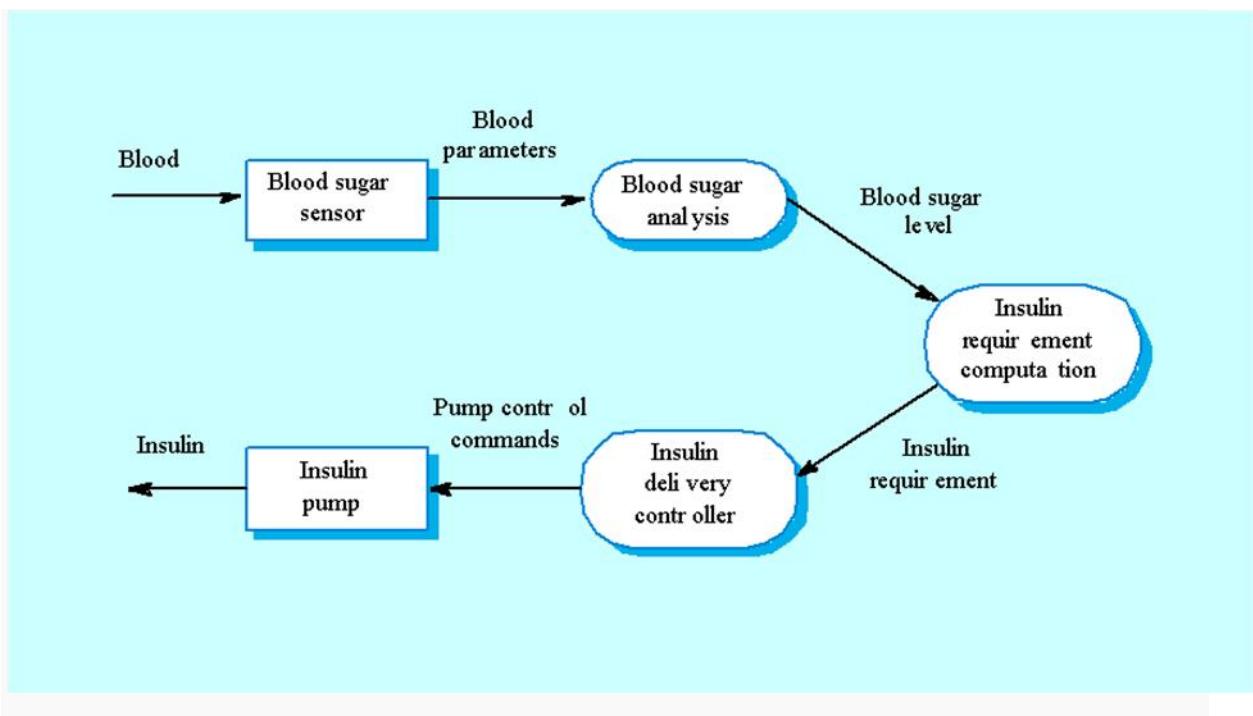
cuối. Mô hình luồng dữ liệu mô hình hoá hệ thống từ góc độ chức năng. Việc tìm vết và tư liệu hoá quan hệ giữa dữ liệu với một quy trình rất có ích đối với việc tìm hiểu toàn bộ hệ thống. Mô hình luồng dữ liệu là phần cốt lõi của rất nhiều phương pháp phân tích. Nó chứa các ký pháp rất dễ hiểu đối với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

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



Các biểu đồ luồng dữ liệu cũng có thể hữu ích trong việc chỉ ra dữ liệu gì được trao đổi giữa hệ thống này các hệ thống khác trong môi trường của nó.

Ví dụ: Xét mô hình luồng dữ liệu của chiếc bơm Insulin



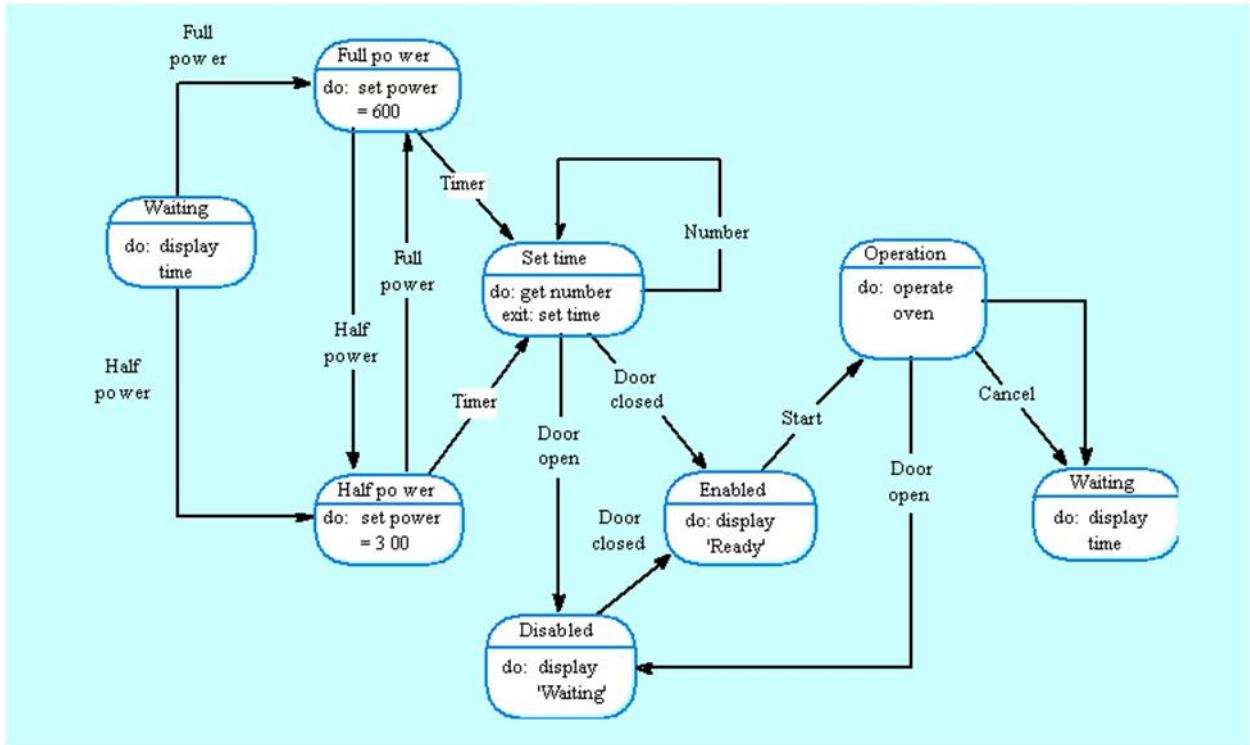
4.2.2. Mô hình máy trạng thái

Các mô hình máy trạng thái mô tả các đáp ứng của hệ thống với các sự kiện xảy ra bên trong và bên ngoài của nó. Chúng thường được sử dụng cho việc mô hình hóa các hệ thống thời gian thực.

Các mô hình máy trạng thái biểu diễn các trạng thái của hệ thống và các sự kiện gây ra sự dịch chuyển trạng thái của hệ thống. Mỗi trạng thái của hệ thống là một nút và một sự kiện tương ứng với cung nối giữa hai nút xác định. Khi có một sự kiện xảy ra, hệ thống sẽ dịch chuyển từ trạng thái này sang trạng thái khác.

Các biểu đồ trạng thái là một phần của UML và chúng được sử dụng để biểu diễn các mô hình máy trạng thái. Biểu đồ trạng thái cho phép phân tích một trạng thái thành nhiều trạng thái 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 (hình)



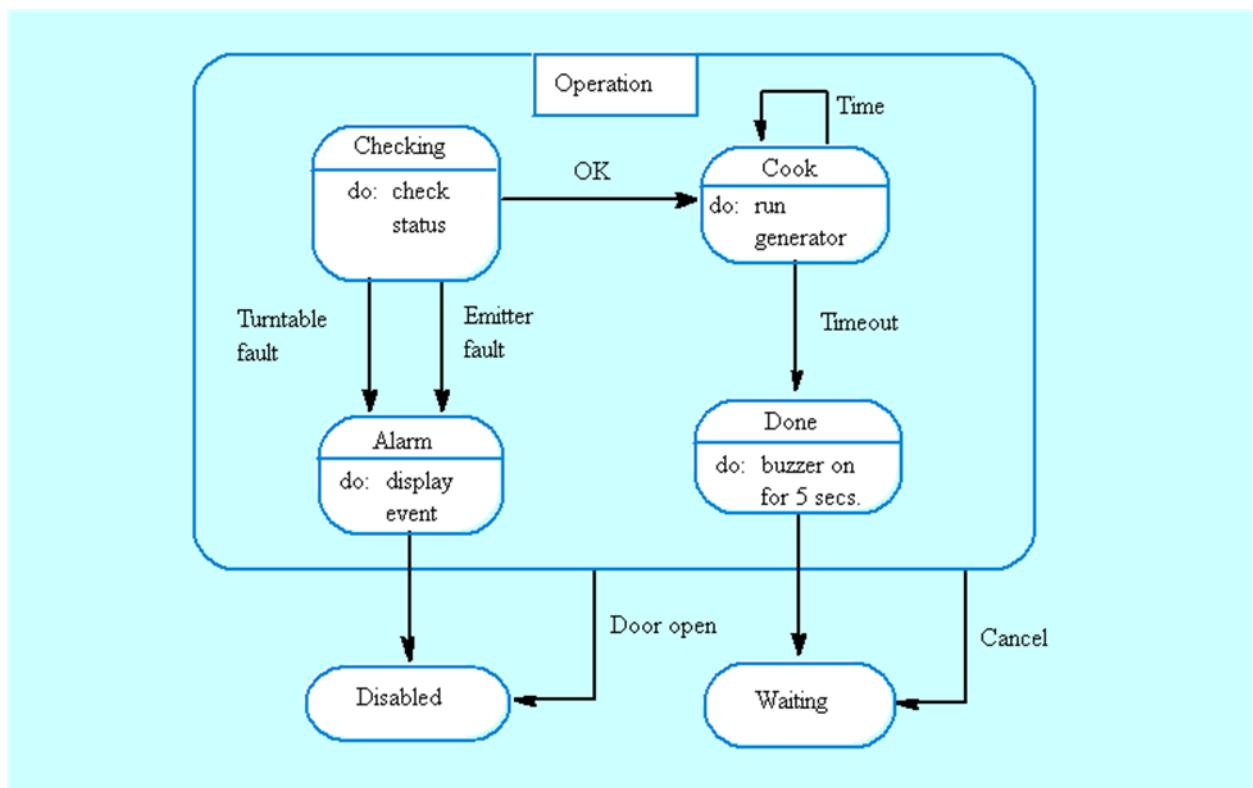
Mô tả các trạng thái của lò vi sóng:

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 "Half power"
Full power	The oven power is set to 600 watts. The display shows "Full power"
Set time	The cooking time is set to the user's 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 "Not ready"
Enabled	Oven operation is enabled. Interior oven light is off. Display shows "Ready to cook"
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 "Cooking complete" 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

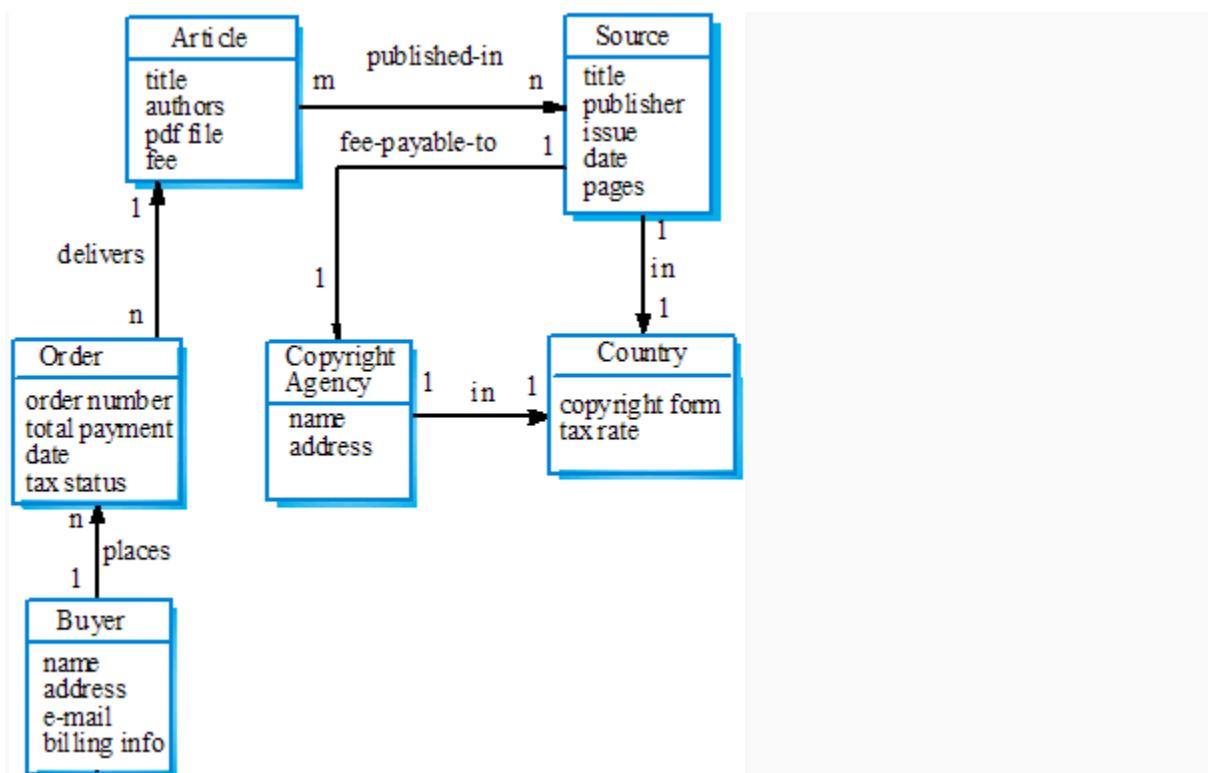
Trạng thái hoạt động/vận hành của lò vi sóng:



4.2.3. Các 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



Các mô hình dữ liệu thường không chi tiết, do đó chúng ta có thể sử dụng từ điển dữ liệu làm công cụ hỗ trợ. Từ điển dữ liệu là danh sách tất cả các tên gọi được sử dụng trong các mô hình hệ thống. Chúng có thể là các thực thể, các quan hệ và các thuộc tính ... Từ điển dữ liệu giúp hỗ trợ quản lý các tên gọi và tránh được sự trùng lặp các tên, lưu trữ kiến thức một cách có tổ chức để kết nối các pha phân tích, thiết kế và cài đặt. Có khá nhiều các CASE workbenches hỗ trợ tạo từ điển dữ liệu cho hệ thống phần mềm.

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. Các 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, hay thực thể trong thế giới thực.

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

Một hệ thống có thể cần tạo nhiều mô hình đối tượng như:

- ✓ *Các mô hình thừa kế;*
- ✓ *Các mô hình kết tập;*
- ✓ *Các mô hình tương tác.*

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. Việc phát hiện các lớp đối tượng là một quy trình rất khó khăn đòi hỏi ta phải tìm hiểu sâu về lĩnh vực của ứng dụng. Các lớp đối tượng thường phản ánh 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.

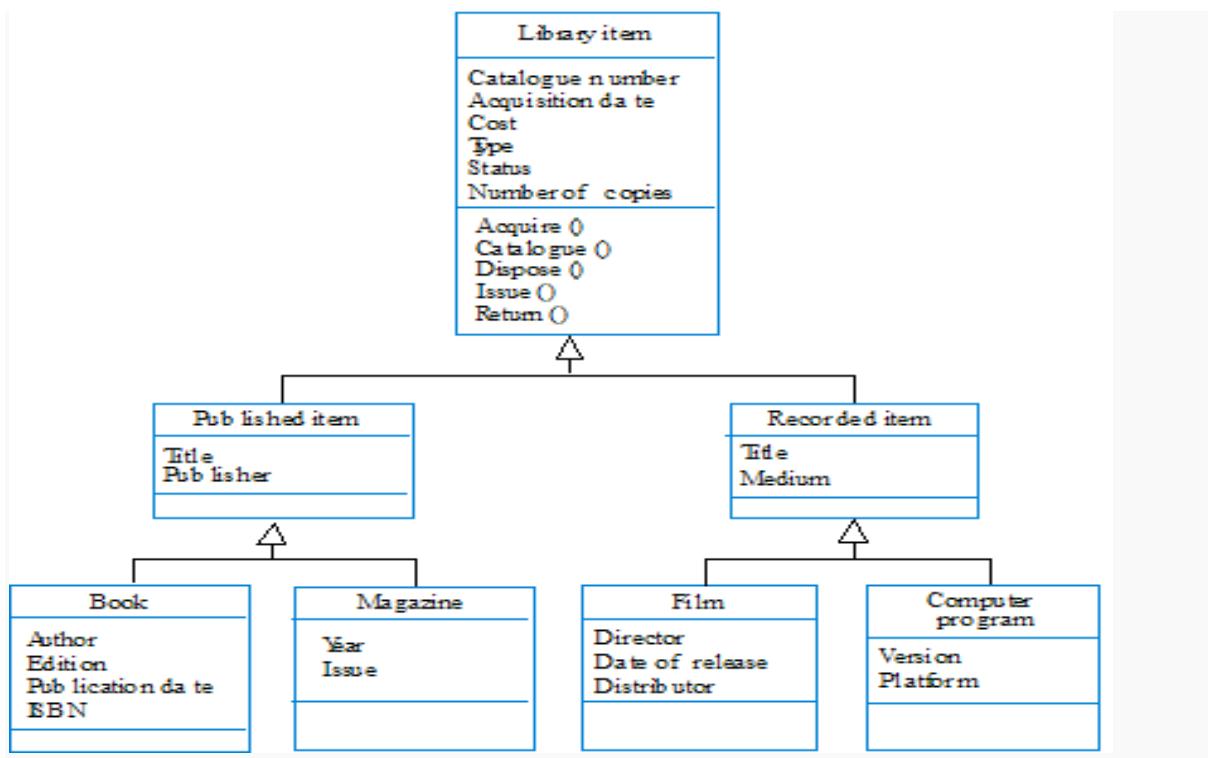
4.4.1. Mô hình thừa kế

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

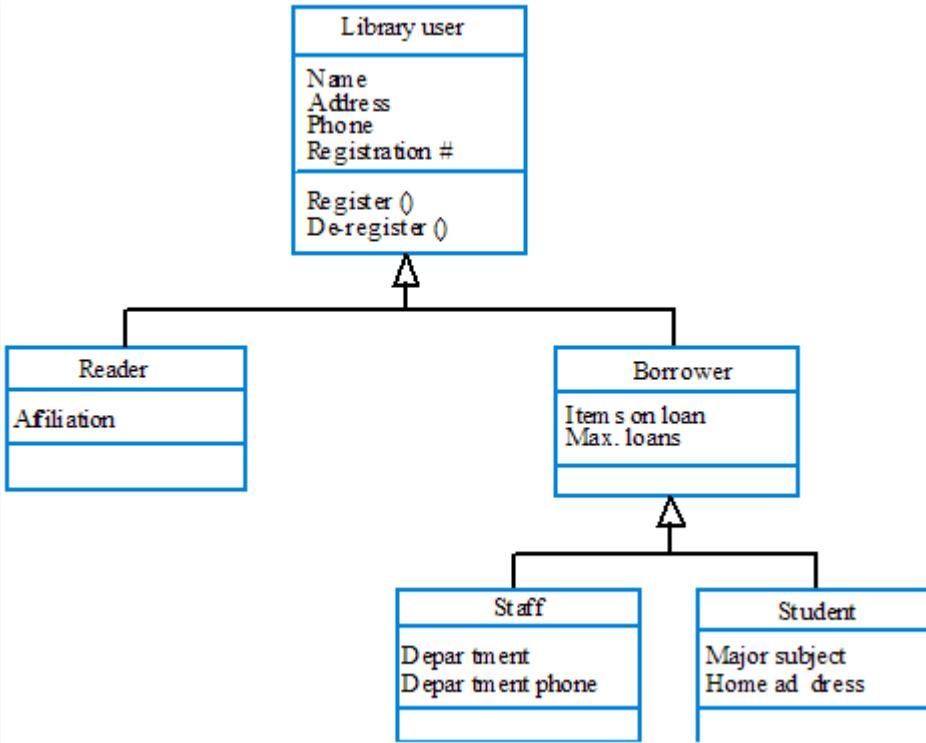
tương thừa kế những thuộc tính và phương thức của các lớp cha của nó có thể bổ sung những đặc điểm của riêng nó.

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 giữa các lớp của LIBSYS

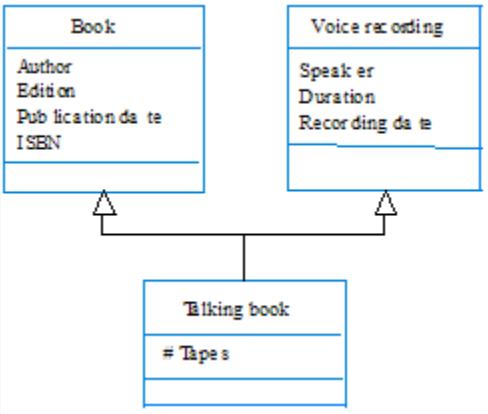


Ví dụ 2: cấu trúc phân cấp của lớp User trong LIBSYS



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

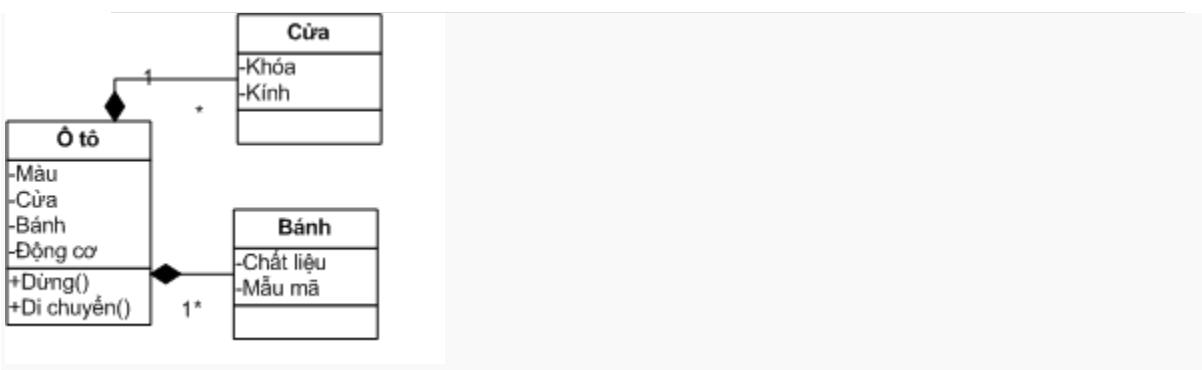


4.4.2. Mô hình kết hợp

Mô hình kết hợp biểu diễn cách cấu tạo của một lớp từ các lớp khác. Mô hình kết hợp giống như các mô hình dữ liệu biểu diễn mối quan hệ hợp - thành (part-of).

Ví dụ 1: Mô hình kết hợp

Đối tượng ô tô được tạo thành từ nhiều đối tượng khác như: cửa, bánh xe ...

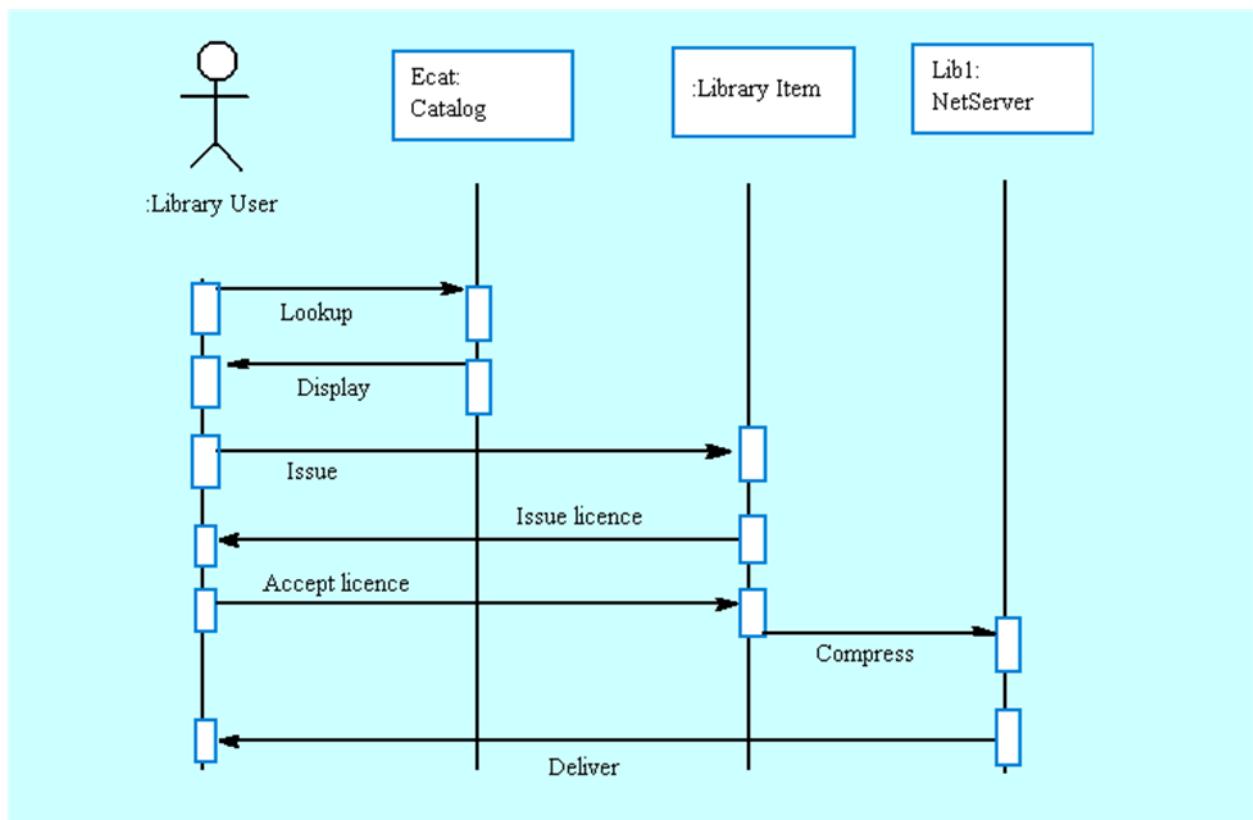


4.4.3. Mô hình ứng xử

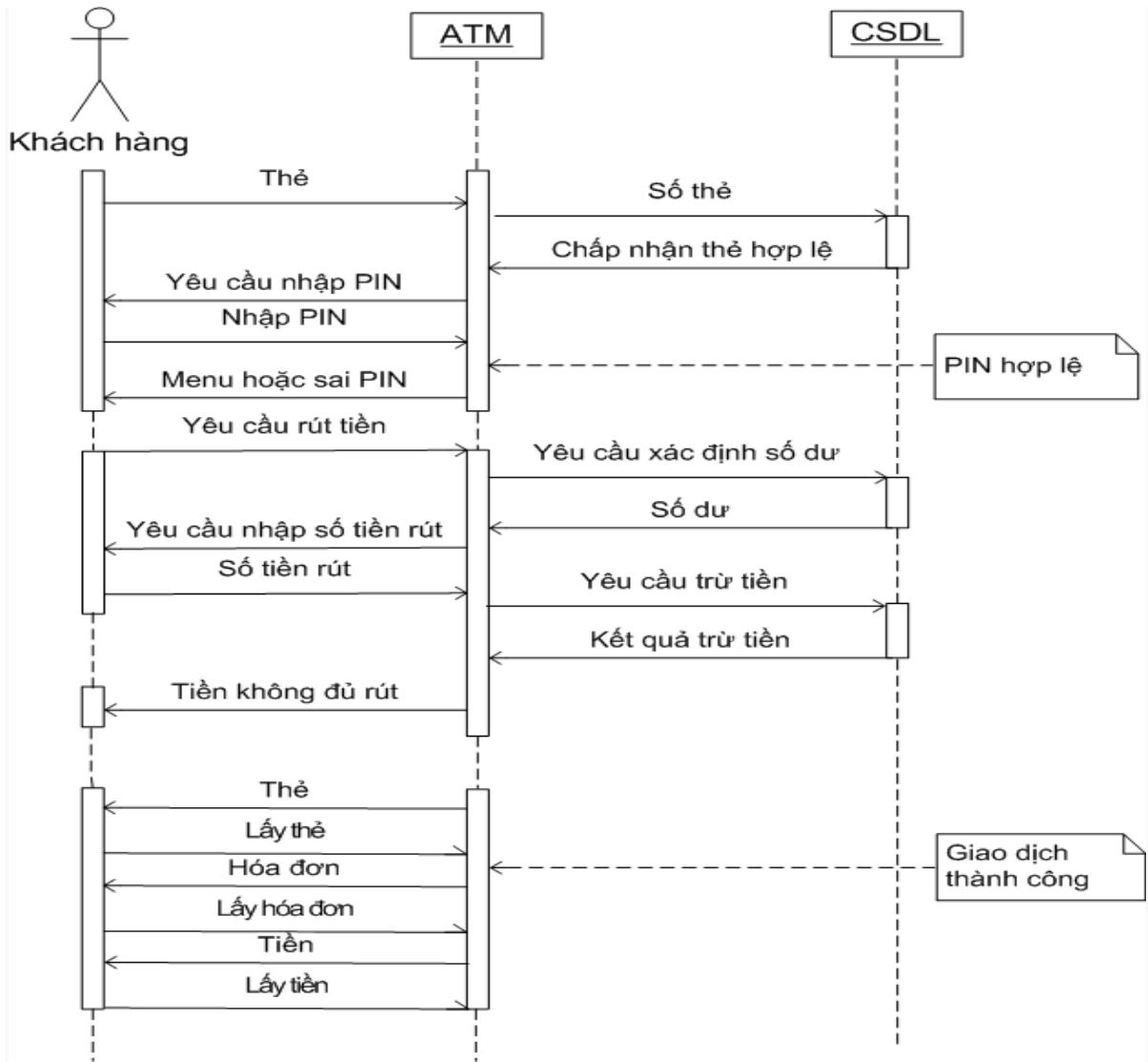
Mô hình ứng xử mô tả tương tác giữa các đối tượng nhằm tạo ra một số ứng xử cụ thể của hệ thống mà đã được xác định như là một ca sử dụng.

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

Ví dụ 1: Biểu đồ trình tự của ca phát hành tài liệu điện tử đến độc giả của hệ thống LIBSYS



Ví dụ 2: Mô tả ca sử dụng Rút tiền của hệ thống ATM.



4.5. Phương pháp hướng cấu trúc

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:

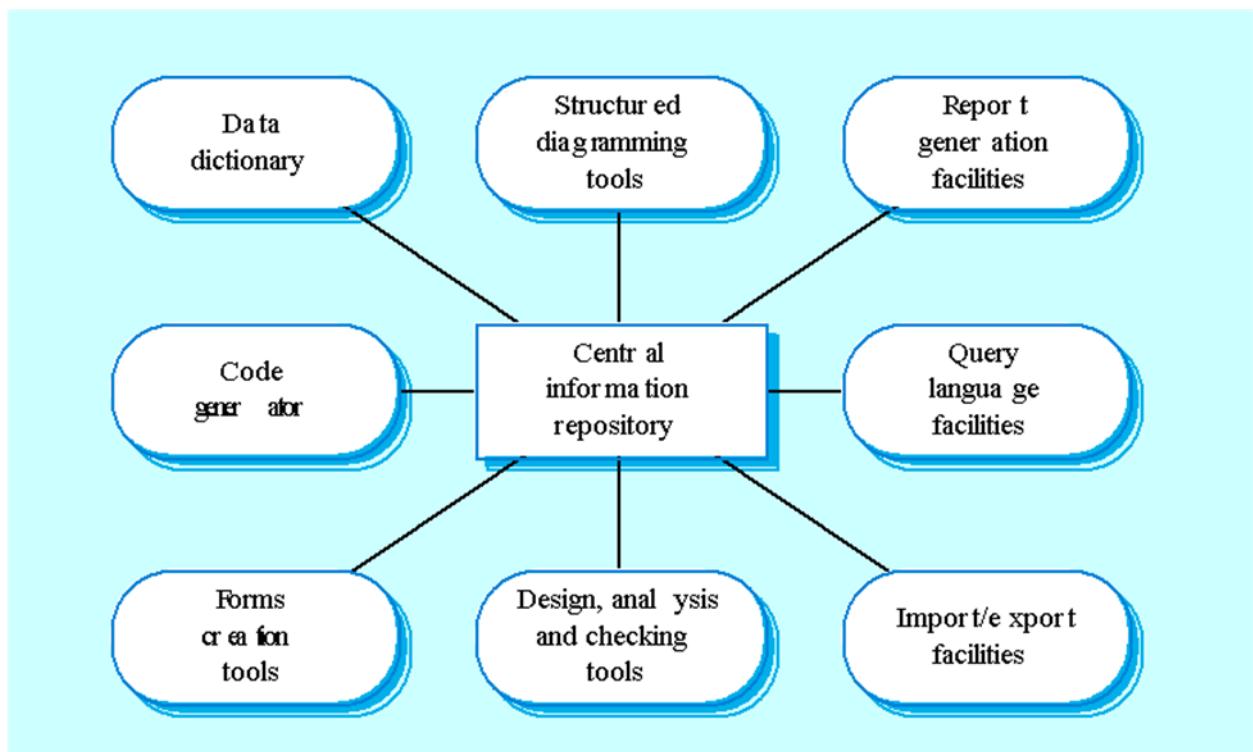
- ✓ Không mô hình hoá được các yêu cầu hệ thống phi chức năng
- ✓ Không chứa những thông tin để xác định liệu phương pháp có thích hợp với một vấn đề đưa ra hay không.

- ✓ Tạo ra quá nhiều tài liệu
- ✓ Mô hình hóa 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 để 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 hoạt động tiền trình phần mềm có liên quan như phân tích, thiết kế hoặc kiểm thử..
- ✓ Workbenches phân tích và thiết kế hỗ trợ việc mô hình hóa hệ thống trong tiến trình RE và tiến trình thiết kế hệ thống. Các workbenches này có thể hỗ trợ phương pháp thiết kế cụ thể hoặc cung cấp sự hỗ trợ cho việc tạo một số kiểu mô hình hệ thống khác nhau.
- ✓ Workbench phân tích và thiết kế:



- ✓ Các thành phần workbench phân tích:
 - Các bộ soạn thảo
 - Các công cụ kiểm tra và phân tích mô hình
 - Kho chứa và ngôn ngữ truy vấn kết hợp
 - Từ điển dữ liệu

- Các công cụ phát sinh và định nghĩa báo cáo
- Các công cụ định nghĩa Forms
- Các bộ chuyển đổi Import/export .
- Các công cụ phát sinh mã.

Các CASE tool hỗ trợ mô hình hóa hệ thống là một công cụ quan trọng của phương pháp hướng cấu trúc.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 4

Chương 5 THIẾT KẾ VÀ CÀI ĐẶT HỆ THỐNG

5.1 Thiết kế phần mềm

5.1.1 Thiết kế phần mềm là gì

Thiết kế phần mềm là hoạt động chuyên 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.

5.1.2 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
- ✓ Là 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.

5.1.3 Các khái niệm thiết kế cơ sở

Trùu tượng hóa	<i>Trùu tượng hóa dữ liệu, thủ tục, điều khiển</i>
Làm mịn	<i>Chi tiết hóa các trùu tượng theo ý đồ</i>

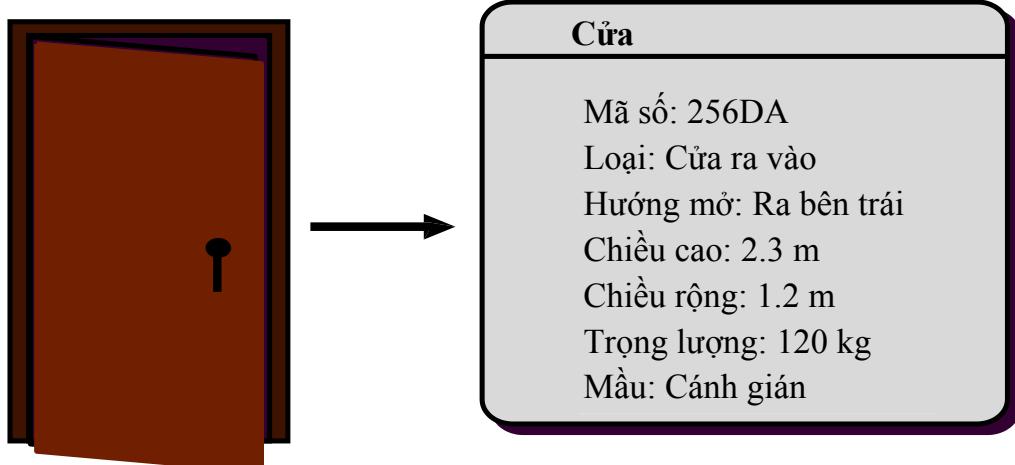
Tính mô đun	<i>Phân chia dữ liệu và chức năng</i>
Kiến trúc	<i>Cấu trúc tổng thể của phần mềm</i>
Thủ tục	<i>Thuật toán để thực hiện</i>
Cơ chế	<i>Điều khiển bằng giao diện</i>

a. Trùu tượng hóa

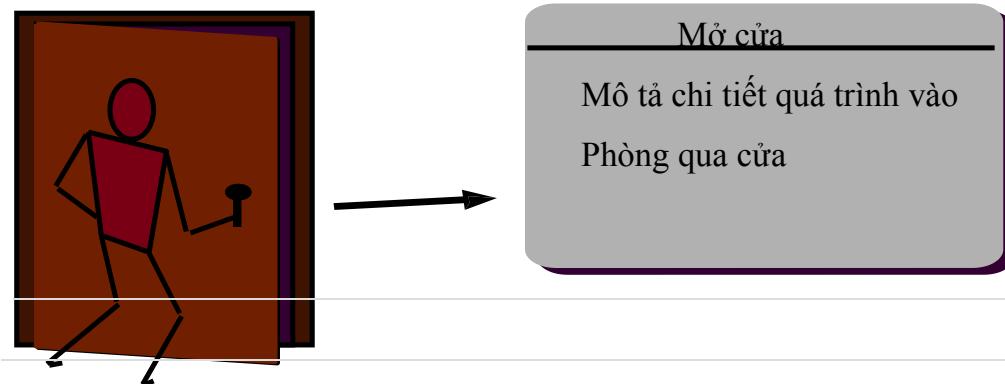
Trùu tượng hóa là khái niệm cơ sở trong tư duy của con người. Nó là quá trình ánh xạ một sự vật, hiện tượng trong thế giới thực thành một khái niệm logic.

Có nhiều mức độ trừu tượng khác nhau, cho phép con người tập trung (tư duy) vào việc giải quyết vấn đề mà không cần bận tâm đến các chi tiết khác, biểu diễn vấn đề bằng một cấu trúc tự nhiên.

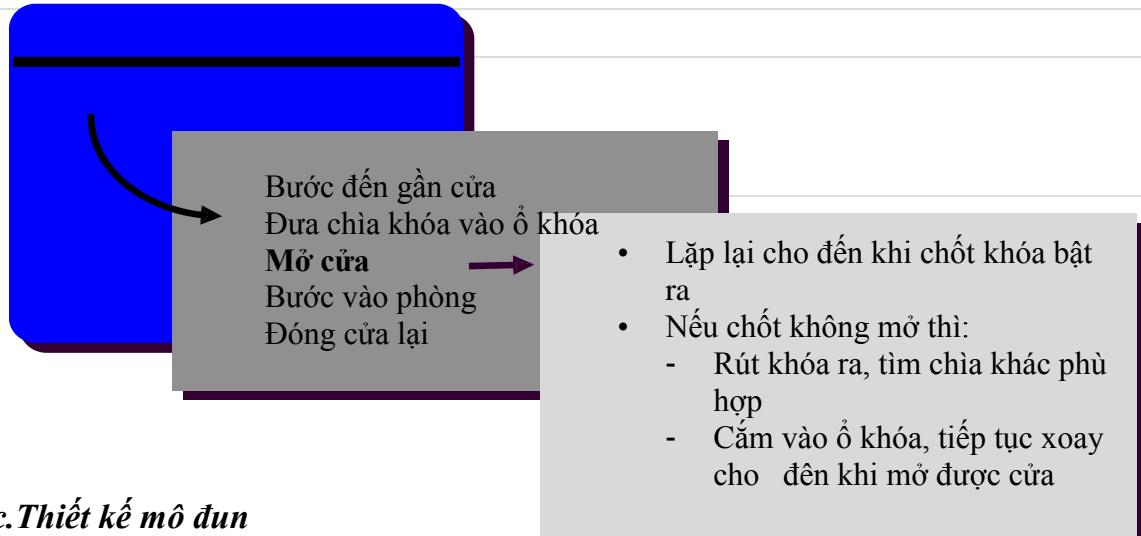
Ví dụ 1: trừu tượng hóa dữ liệu



Ví dụ 2: Trùu tượng hóa thủ tục



b. Làm mịn từng bước



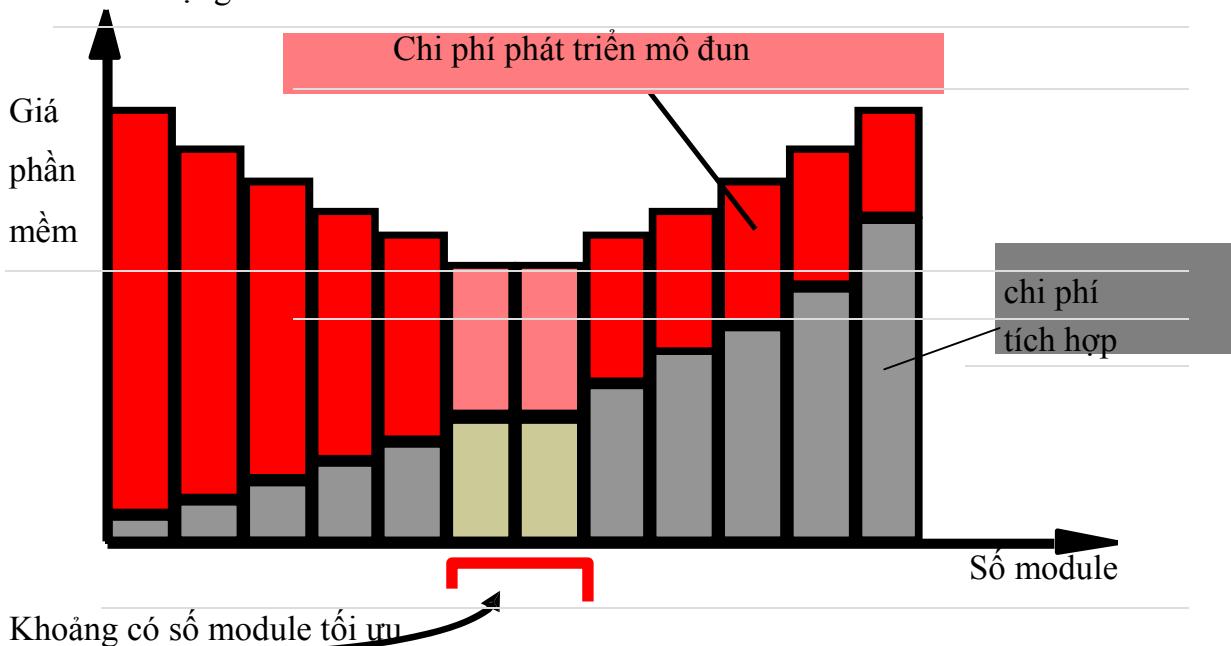
c. Thiết kế mô đun

Thiết kế mô đun dựa trên quan điểm “chia để trị”. Xét 2 mô đun P1 và P2. Gọi C là độ phức tạp của mô đun, và E và công sức thực hiện module. Ta có

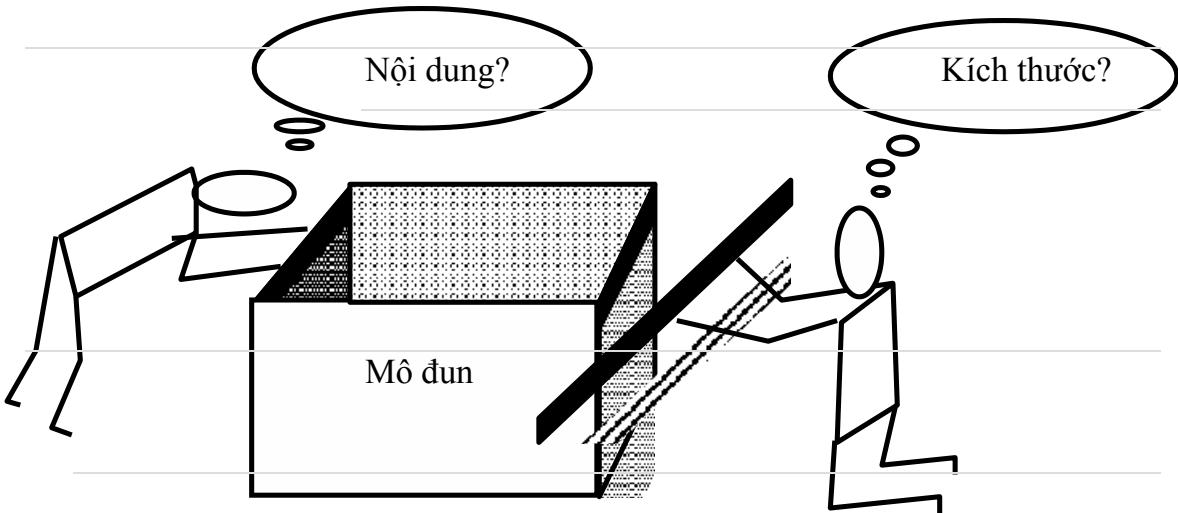
$$C(P1+P2) > C(P1)+C(P2); \quad E(P1+P2) > E(P1)+E(P2)$$

Do đó, chúng ta cần thiết kế các mô đun đảm bảo:

- ✓ Giảm độ phức tạp, có khả năng phát triển song song.
- ✓ Cục bộ, dễ bảo trì, dễ hiểu, dễ tái sử dụng.
- ✓ Số lượng module là tối ưu



- ✓ Kích cỡ module cũng cần thiết kế phù hợp: Kích cỡ module được quyết định dựa trên khái niệm độc lập chức năng: Mỗi module nên thực hiện một công việc, điều này đảm bảo tính dễ hiểu, dễ sửa đổi, dễ tái sử dụng.



c. *Che dấu thông tin*

Để che dấu thông tin, ta nên sử dụng mô đun thông qua giao diện. Người dùng chỉ cần đưa vào các tham số và nhận được các giá trị trả về. Khi sử dụng mô đun ta không cần biết cách thức cài đặt của nó như thuật toán, cấu trúc dữ liệu, các giao diện ngoại lai (mô đun thứ cấp và thiết bị vào/ra), tài nguyên hệ thống....của mô đun. Lý do cần che dấu thông tin: Che dấu thông tin giúp tạo ra phần mềm với chất lượng tốt hơn, cụ thể giúp:

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

d. *Chất lượng thiết kế*

Ba đặc trưng sau đây được xem như là một chỉ dẫn cho một thiết kế tốt (McMlaughli[MCG91]):

Thiết kế phải *triển khai được tất cả các yêu cầu* trong mô hình phân tích và các yêu cầu tiềm ẩn mà khách hàng đòi hỏi.

Thiết kế cần là *bản hướng dẫn đọc, dễ hiểu* cho người viết chương trình, người kiểm thử và người bảo trì.

Thiết kế cần cung cấp *một bức tranh đầy đủ về phần mềm trên quan điểm triển khai* hướng đến các mặt dữ liệu, chức năng và hành vi của hệ thống.

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

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

Understandability: Tính hiểu được của bản thiết kế,

Adaptability: Tính thích nghi được.

1. Coupling (Tính ghép nối)

Độ đo sự liên kết (trao đổi dữ liệu) giữa các mô đun. Nếu ghép nối chặt chẽ thì sẽ dẫn đến khó hiểu, khó sửa đổi do phải tính đến các liên kết có thể, dễ gây lỗi lan truyền. Do đó, mức độ quan hệ giữa các module là tốt nếu các mô đun ghép nối lỏng lẻo với nhau. Mức độ lỏng lẻo này được thể hiện qua loại hình ghép nối (xem hình ...)

Ghép nối thường		Loose and best
Ghép nối dữ liệu		Still very good
Ghép nối nhãn		Ok
Ghép nối điều khiển		Ok
Ghép nối chung		Very bad
Ghép nối nội dung		Tight and Worst

Trong đó:

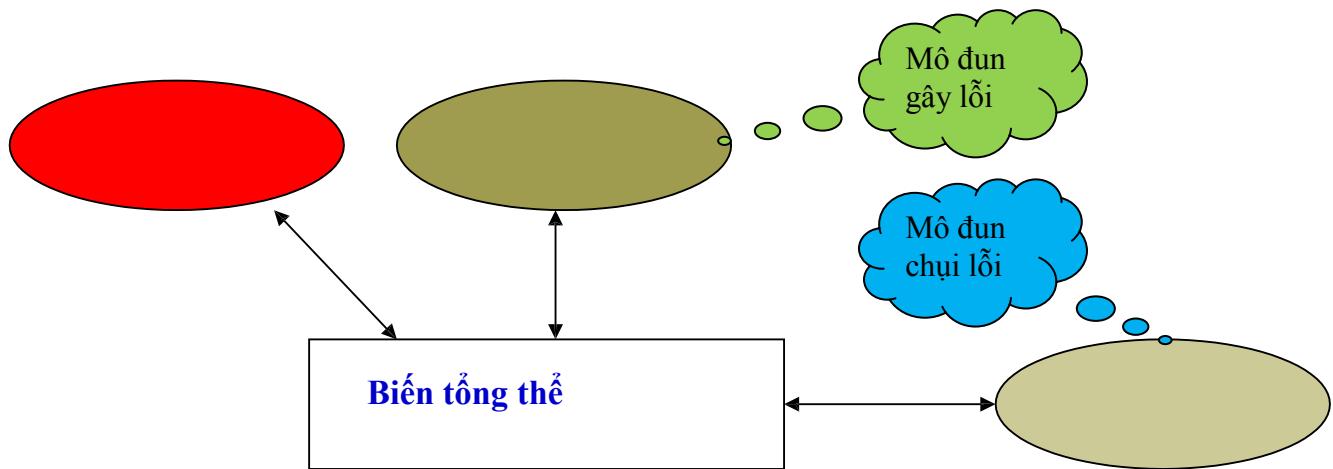
- ❖ **Ghép nối nội dung:** Các mô đun dùng dữ liệu hay thông tin điều khiển được duy trì trong một mô đun khác. Đây là trường hợp xấu nhất.

Ví dụ:

- Các ngôn ngữ bậc thấp chỉ dùng biến chung,
- Làm dụng lệnh goto trong một chu trình

Mô đun 1:	Mô đun 2
10 $k=1$	
20 <i>gosub 100</i>	$100 \quad Y=3*k*k+7*k-3$
$>120 \ goto 60$	<i>110 return</i>
$K+1$	
50 <i>goto 20</i>	

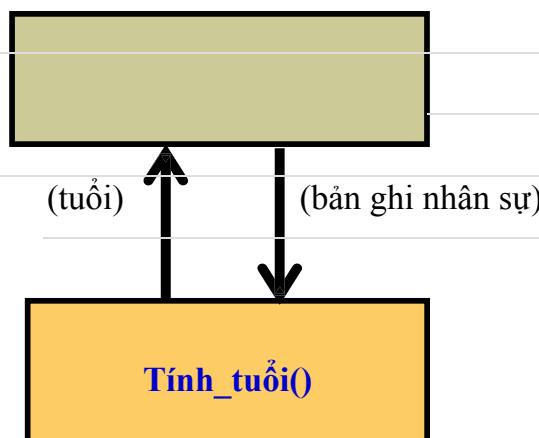
- ❖ **Ghép nối chung:** Các mô đun trao đổi dữ liệu thông qua biến tổng thể. Do đó, lỗi của mô đun này có thể ảnh hưởng đến hoạt động của mô đun khác, khó sử dụng lại các mô đun



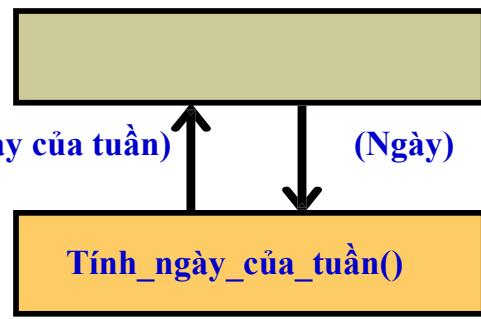
- ❖ **Ghép nối điều khiển:** Các mô đun trao đổi thông tin điều khiển, điều này làm cho thiết kế khó hiểu, khó sửa đổi và dễ nhầm lẫn.

<pre>Procedure PrintRec is begin <i>Display Name (name, sex);</i> end PrintRec;</pre>	<pre>procedure DisplayName (in :name,sex) is begin <i>if sex = m then</i> <i>print Mr.</i> <i>else</i> <i>print Ms</i> <i>print name</i> end DisplayName;</pre>
---	--

- ❖ **Ghép nối nhãn:** Các mô đun trao đổi thừa thông tin. Một mô đun có thể thực hiện chức năng ngoài ý muốn, điều này làm giảm tính thích nghi của mô đun.



- ❖ **Ghép nối dữ liệu:** Truyền dữ liệu qua tham số, nhận kết quả qua tham số và giá trị trả lại



❖ **Ghép nối thường:** Các mô đun là những công việc hoàn toàn độc lập:

Ngủ_dậy(); Đánh_răng(); Rửa_mặt(); Ăn_sáng(); Đi_làm();

2. Cohension (*Tính kết dính*)

Đo sự phụ thuộc lẫn nhau giữa các thành phần trong một module. Nếu tính kết dính cao thì tính cục bộ sẽ cao (đảm bảo độc lập chức năng), mô đun sẽ dễ hiểu, dễ sửa đổi.

Do đó tiêu chuẩn thiết kế tốt là sự kết dính chặt và sự ghép nối lỏng giữa các module.

Sự kết dính thể hiện ở chỗ mỗi mô đun chỉ nên thực hiện một chức năng, mọi thành phần của mô đun phải tham gia thực hiện chức năng đó. Hình ... mô tả các mức độ kết dính:

Chức năng	Hight and best
Tuần tự	Ok
Truyền thông	Still ok
Thủ tục	Not bad at all
Thời điểm	Still not bad at all
Logic	Still not bad at all

Gom gốp

lowest and worst by far

Trong đó:

- ❖ **Kết dính gom gốp (coincidental cohesion):** Gom các thành phần không liên quan với nhau vào nhau
- ❖ **Kết dính logic (logical cohesion):** Gồm các thành phần làm chức năng logic tương tự. Ví dụ Hàm xử lý lỗi chung
- ❖ **Kết dính thời điểm/tạm thời (temporal cohesion):** Gồm các thành phần hoạt động cùng thời điểm. Ví dụ hàm khởi tạo, đọc dữ liệu, cấp phát bộ nhớ
- ❖ **Kết dính thủ tục (procedural cohesion):** Các thành phần thực hiện theo một thứ tự xác định. Ví dụ: Tính lương cơ bản, tính phụ cấp, tính bảo hiểm.
- ❖ **Kết dính truyền thông (communicational cohesion):** Các thành phần truy cập đến cùng tập dữ liệu. Ví dụ tính toán thống kê (tính max, tính min, mean, variation, ..)
- ❖ **Kết dính tuần tự (Sequential cohesion):** Đầu ra của một thành phần là đầu vào của thành phần tiếp theo. Ví dụ từ ảnh màu -> ảnh đen trắng -> ảnh nén.
- ❖ **Kết dính chức năng (Functional cohesion):** Các thành phần cùng góp phần thực hiện một chức năng. Ví dụ các thao tác sắp xếp

3. *Understandability (Tính hiểu được)*

Tính hiểu được là kết quả tổng hợp từ nhiều thuộc tính như:

- ✓ Cấu trúc rõ ràng, tốt
- ✓ Ghép nối lỏng lẻo
- ✓ Kết dính cao
- ✓ Được lập tài liệu
- ✓ Thuật toán, cấu trúc dễ hiểu.

4. *Adaptability (Khả năng thích nghi)*

Khả năng thích nghi là kết quả tổng hợp từ các thuộc tính sau:

-
- ✓ *Tính hiểu được*: Giúp sửa đổi được, tái sử dụng được
 - ✓ *Tự chừa*: Không sử dụng thư viện ngoài, thuộc tính này mâu thuẫn với xu hướng tái sử dụng.
-

Thiết kế hướng đối tượng và chất lượng

Thiết kế hướng đối tượng hướng tới chất lượng thiết kế tốt:

-
- ✓ Đóng gói, che dấu thông tin (đảm bảo độc lập dữ liệu)
 - ✓ Các thực thể hoạt động độc lập (đảm bảo tính cục bộ, có thể dùng lại)
 - ✓ Trao đổi dữ liệu qua truyền thông (liên kết yếu)
 - ✓ Có khả năng kế thừa (đảm bảo khả năng dùng lại)
 - ✓ Cục bộ, dễ hiểu, dễ tái sử dụng
-

5.1.4 Thiết kế thiết kế

Phần mềm là tập các mô đun tương tác lẫn nhau. Mô đun hóa là chìa khóa cho một phần mềm tốt. Mục tiêu của thiết kế là xác định:

-
- ✓ Các mô đun chức năng
 - ✓ Cách thức cài đặt mô đun
 - ✓ Sự tương tác giữa các modun.
-

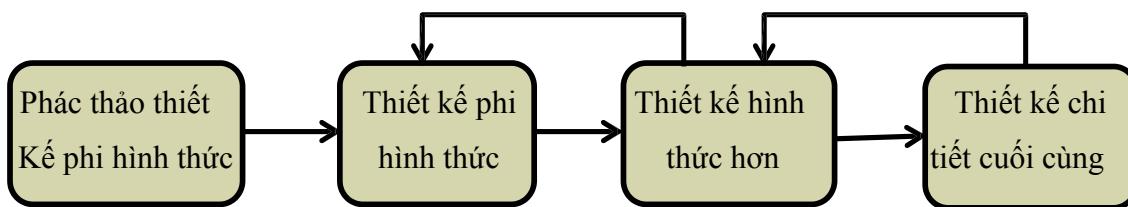
5.1.5 Nguyên lý thiết kế

1. Thiết kế không bị bó buộc vào một cách nhìn hạn chế nào. Nó cần được lựa chọn từ các giải pháp có thể.
 2. Thiết kế phải cho phép lật ngược lại mô hình phân tích. Các mô đun và các yêu cầu không nhất thiết phải tương ứng 1-1, nhưng phải kiểm tra được sự thỏa mãn các yêu cầu.
 3. Không nên tạo lại các thiết kế (giải pháp) đã có, mà cần tái sử dụng tối đa chúng.
 4. Mô hình thiết kế (giải pháp) nên tiến gần đến mô hình thế giới thực (bài toán).
-

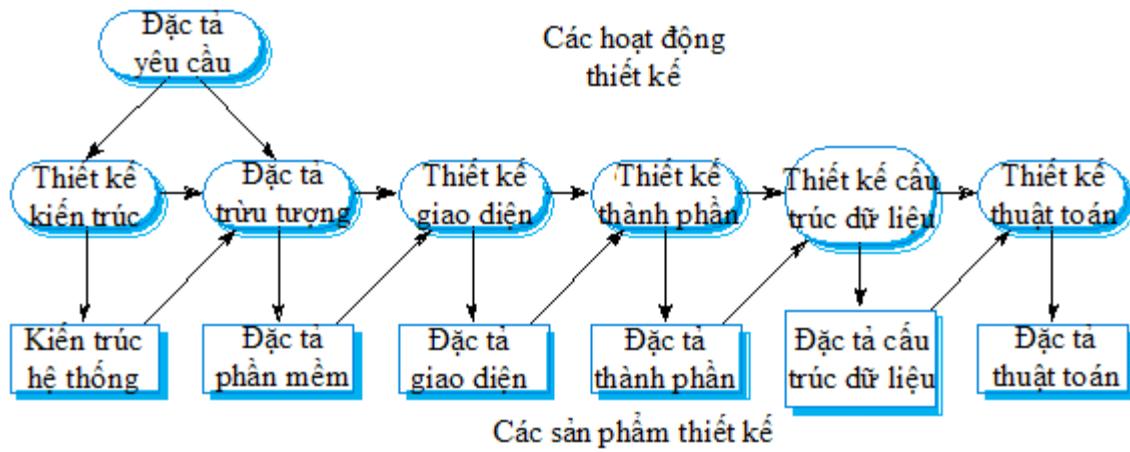
5. Biểu diễn thiết kế phải nhất quán và có tính tích hợp. Thiết kế do nhiều người tiến hành //, do đó phải thống nhất cách biểu diễn, thống nhất giao diện.
 6. Thiết kế cần có cấu trúc dễ hiểu, dễ thay đổi tức là phải được mô đun hóa và phân cấp.
 7. Thiết kế không phải là mã hóa. Kết quả thiết kế luôn có mức trừu tượng hơn mã hóa, phải đảm bảo dễ hiểu, dễ thay đổi.
 8. Thiết kế cần được đánh giá chất lượng ngay trong khi được tạo ra. Chất lượng thiết kế được đánh giá qua tính kết dính, tính ghép nối, hiệu quả thuật toán
 9. Thiết kế cần được thẩm định để tránh các lỗi mang tính hệ thống. Các lỗi có thể là thiếu chức năng, chức năng không rõ, mâu thuẫn,
-

5.1.6 Tiến trình thiết kế

Bản thiết kế phần mềm là một tài liệu mô tả về cấu trúc của phần mềm được cài đặt. Thông thường, người thiết kế khi thiết kế không đi tới một thiết kế hoàn thiện ngay lập tức mà phát triển thiết kế lặp qua một số phiên bản. Mỗi lần lặp tiến trình thiết kế thêm dần tính hình thức và tính chi tiết. Thiết kế được phát triển theo vết liên tục để sửa các thiết kế sóm hơn. Tiến trình thiết kế có thể bao gồm việc phát triển một số mô hình hệ thống với *các mức độ trừu tượng khác nhau*. Như vậy tiến trình thiết kế là quá trình tăng cường tính hình thức hóa, tính chính xác và luôn quay lại các bản thiết kế trước đó để kiểm tra và hoàn thiện dần các bản thiết kế sau.



Mô hình tổng quát của tiến trình thiết kế



Hình 2.7: Mô hình chung của quy trình thiết kế

Trên đây là mô hình thiết kế hệ thống nói chung. Tuy nhiên ta có thể tùy biến nó theo các cách tiếp cận khác nhau để phù hợp với ứng dụng cụ thể đang cần xây dựng. Ví dụ như chúng ta có thể thích nghi nó theo các cách sau:

- ✓ Hai giai đoạn sau của thiết kế có thể được trì hoãn tới khi bắt đầu tiến trình cài đặt.
- ✓ Nếu thiết kế sử dụng cách tiếp cận thăm dò, các giao diện hệ thống có thể được thiết kế sau khi cấu trúc dữ liệu được đặc tả.
- ✓ Khi sử dụng các phương pháp phát triển nhanh:
 - Các đầu ra của tiến trình thiết kế không là các tài liệu riêng lẻ mà là mã chương trình.
 - Mỗi lần tăng thiết kế, bản thiết kế được biểu diễn bởi mã chương trình hơn là một mô tả thiết kế

Hình 5.1 là một tiến trình thiết kế trong đó các hoạt động được tiến hành tuần tự. Trong thực tế các hoạt động này là đan xen nhau. Các phản hồi từ giai đoạn này đến giai đoạn khác dẫn đến phải làm lại thiết kế là không thể tránh khỏi trong mọi tiến trình thiết kế.

Đầu vào của quy trình thiết kế là tài liệu đặc tả yêu cầu phần mềm. Đầu ra của tiến trình thiết kế là các bản đặc tả thiết kế. Mỗi tài liệu này là đầu ra của từng giai đoạn thiết kế tương ứng. Vì thiết kế là liên tục, các bản đặc tả này trở nên chi

tiết dàn, hình thức dàn. Kết quả cuối cùng của tiến trình thiết kế là bản đặc tả chính xác về các giải thuật và các cấu trúc dữ liệu được đặc tả.

5.2. Thiết kế kiến trúc

5.2.1 Tổng quan về thiết kế kiến trúc

a. Khái niệm kiến trúc

Kiến trúc phần mềm chỉ cấu trúc tổng thể của một phần mềm và cách thức tổ chức qua đó cho ta một sự tích hợp về mặt khái niệm của một hệ thống [SHA95a]

Kiến trúc thông thường được thể hiện bằng một biểu đồ phân cấp của các thành phần và mối quan hệ giữa chúng. Một kiến trúc đầy đủ nếu nó thể hiện hệ thống theo nhiều góc nhìn khác nhau như tĩnh, động, dữ liệu, triển khai..

b. Vai trò của kiến trúc phần mềm

Kiến trúc phần mềm không phải là mô hình hoạt động, mà nó là mô hình phân hoạch theo những cách nhìn khác nhau (chức năng, dữ liệu, tiến trình, tĩnh hay động, ..). Bản kiến trúc hệ thống giúp các kỹ sư hệ thống:

- ✓ Phân tích tính hiệu quả của kết quả thiết kế đáp ứng được yêu cầu của phần mềm
 - ✓ Tìm các giải pháp thay thế cấu trúc ở giai đoạn sớm
 - ✓ Tìm các rủi ro liên quan đến cấu trúc.
-

c. Khái niệm về thiết kế kiến trúc

Thiết kế kiến trúc là quá trình xác định các hệ con lập thành hệ thống và khung làm việc để điều khiển và giao tiếp giữa các hệ con với nhau (sự truyền thông giữa chúng)

Thiết kế kiến trúc là hệ thống là giai đoạn sớm nhất trong quy trình thiết kế hệ thống, nó được tiến hành song song cùng với một số hoạt động đặc tả. Khi thiết kế kiến trúc, chúng ta sử dụng các biểu đồ cấu trúc (structure chart) để mô tả cái nhìn tổng thể về hệ thống, mối quan hệ giữa các mô đun và giao diện giữa các mô đun mà không cần chỉ ra thứ tự thực hiện, số lần thực hiện và chi tiết thiết kế là gì.

Đầu ra của hoạt động thiết kế kiến trúc là tài liệu mô tả về kiến trúc phần mềm. Bản thiết kế kiến trúc biểu diễn sự liên kết giữa các tiến trình thiết kế và đặc tả.

5.2.2 Các lợi ích của kiến trúc hệ thống rõ ràng

Bản thiết kế kiến trúc hệ thống rõ ràng sẽ có tác dụng giúp:

- ✓ **Giao tiếp với Stakeholder:** Kiến trúc có thể được sử dụng để thảo luận bởi các Stakeholders hệ thống.
 - ✓ **Phân tích hệ thống:** Có thể dùng kiến trúc để phân tích xem hệ thống có thỏa mãn các yêu cầu phi chức năng của nó hay không.
 - ✓ **Tái sử dụng Large-scale:** Bản kiến trúc tốt có thể được sử dụng lại cho một loạt các hệ thống.
-

5.2.3 Kiến trúc và các đặc 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
-

5.2.4 Các xung đột kiến trúc

- ✓ Sử dụng các thành phần large-grain cải thiện tính thực thi nhưng giảm khả năng bảo trì.
-

- ✓ Dựa vào các thành phần dư thừa cải thiện tính sẵn dùng nhưng làm cho tính an ninh khó khăn hơn.
 - ✓ Khoanh vùng các đặc trưng liên quan đến tính an toàn thường có nghĩa sự giao tiếp sẽ nhiều hơn do đó giảm khả năng thực thi.
-

5.2.5 Các quyết định thiết kế kiến trúc

Thiết kế kiến trúc hệ thống là một tiến trình sáng tạo, do đó cần các tiến trình thiết kế khác nhau cho các kiểu hệ thống khác nhau. Trong tiến trình thiết kế luôn cần phải tạo ra các quyết định. Sau đây là một số quyết định được tạo chung cho mọi tiến trình thiết kế, thông qua việc trả lời các câu hỏi sau:

- ✓ Có kiến trúc ứng dụng tổng thể để sử dụng cho hệ thống đang triển khai không?
 - ✓ Hệ thống sẽ được phân tán như thế nào?
 - ✓ Những kiểu kiến trúc nào là phù hợp?
 - ✓ Sử dụng cách tiếp cận gì để tổ chức hệ thống?
 - ✓ Hệ thống sẽ được phân chia thành các modules như thế nào?
 - ✓ Sử dụng chiến lược điều khiển gì?
 - ✓ Đánh giá thiết kế kiến trúc như thế nào?
 - ✓ Kiến trúc nên được tư liệu hóa như thế nào?
-

5.2.6 Sử dụng lại kiến trúc

Các bản thiết kế kiến trúc cũng nên được sử dụng lại vì:

- ✓ Các hệ thống cho cùng một miền ứng dụng: Thường có các kiến trúc giống nhau phản ánh các khái niệm của miền.
- ✓ Các dòng sản phẩm: Được xây dựng quanh một kiến trúc lõi, đồng thời bổ sung thêm một số biến đổi để thỏa mãn các yêu cầu khách hàng riêng lẻ.

Các kiến trúc ứng dụng trình bày trong chương 13 và các dòng sản phẩm trong chương 18 [6]

5.2.7 Các mô hình kiến trúc

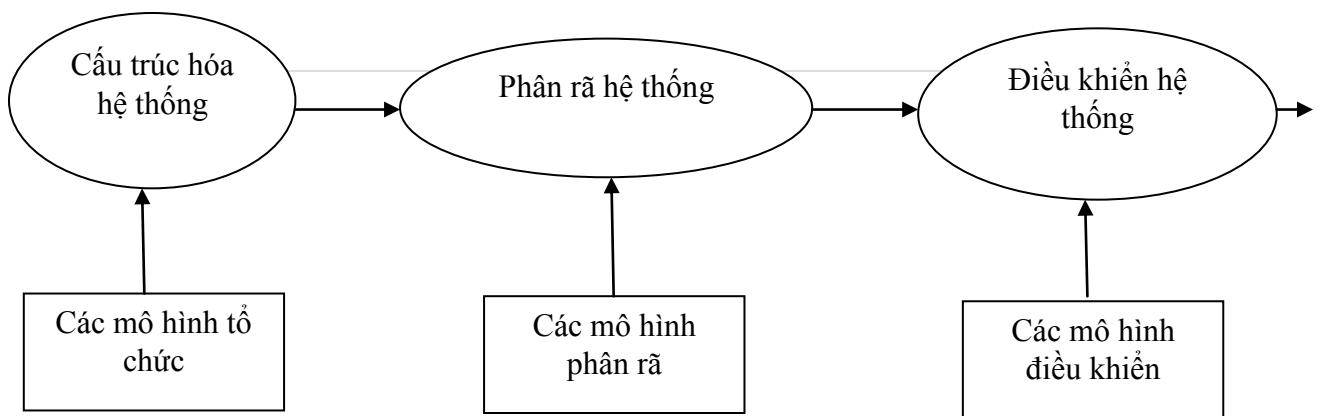
Mô hình kiến trúc của một hệ thống cụ thể có thể tuân theo kiểu kiến trúc tổng thể. Hiểu biết về các kiểu kiến trúc tổng thể này giúp ta dễ dàng hơn trong việc định nghĩa kiến trúc hệ thống. Tuy nhiên, hầu hết các hệ thống lớn là hỗn tạp, không đồng nhất và không tuân theo một kiểu kiến trúc đơn lẻ nào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

5.2.8 Tiến trình thiết kế kiến trúc

Tiến trình thiết kế kiến trúc thường bao gồm các hoạt động được biểu diễn như hình ... sau:



Hình:

a. Cấu trúc hóa hệ thống

Cấu trúc hóa hệ thống là hoạt động phân chia hệ thống thành các hệ con (sub-system) độc lập và xác định sự trao đổi thông tin giữa các hệ con, xác định các giao diện của chúng.

Các mô hình kiến trúc

Khi thiết kế kiến trúc các mô hình thiết kế khác nhau được tạo ra. Mỗi mô hình biểu diễn một cách nhìn cụ thể của kiến trúc. Một số mô hình kiến trúc thông dụng của hệ thống:

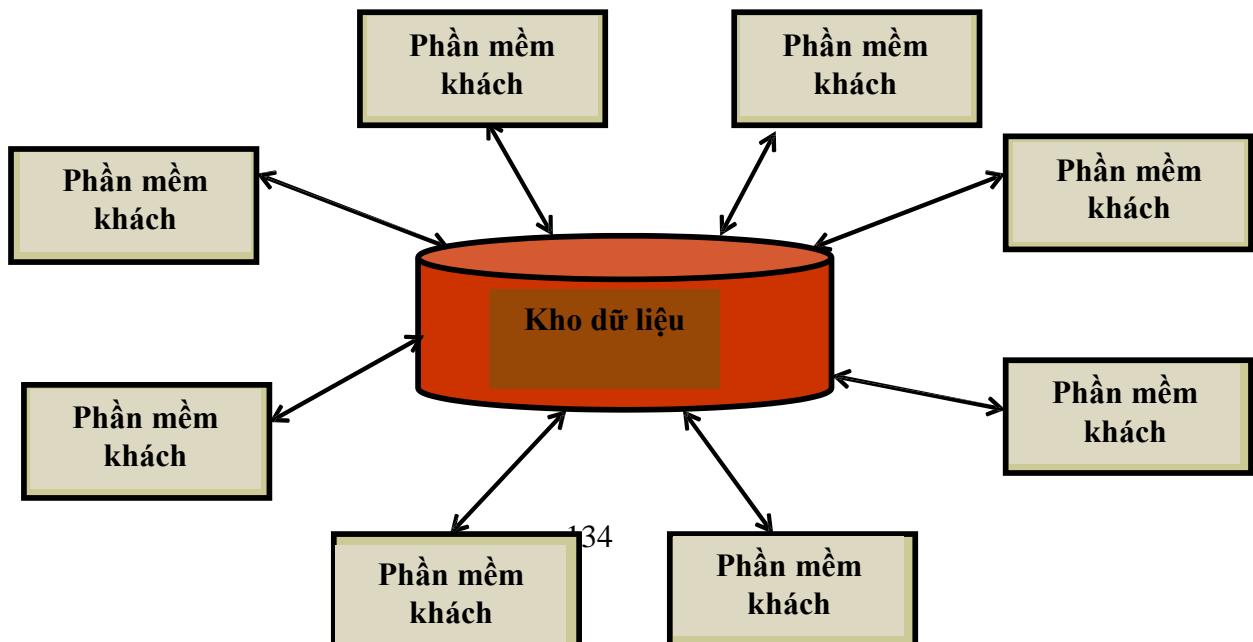
1. Kiến trúc dữ liệu tập trung (Data centered Architectures)
2. Kiến trúc khách-dịch vụ (Client-server Architectures)
3. Kiến trúc phân tầng (Layered Architecture)

1. Kiến trúc dữ liệu tập trung

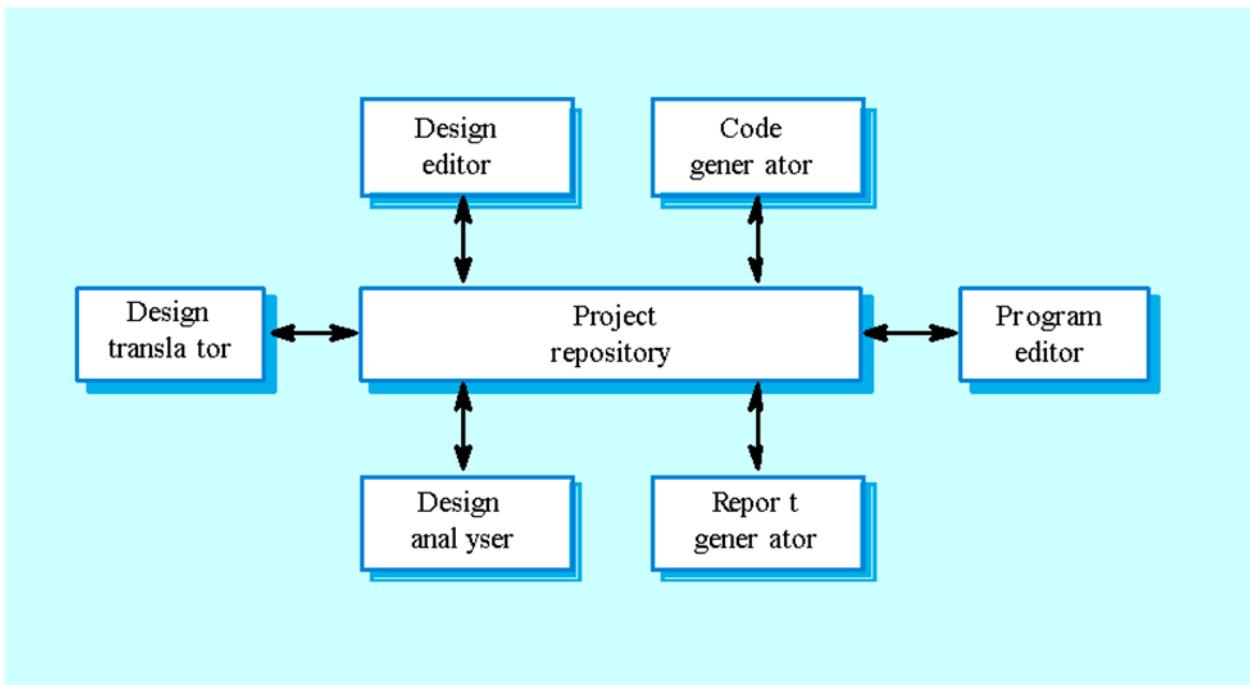
Khi các hệ thống con cần trao đổi thông tin với nhau. Việc trao đổi dữ liệu được thực hiện theo hai cách:

- ✓ Dữ liệu được chia sẻ chung trong một CSDL trung tâm/kho dữ liệu trung tâm và được truy cập bởi các hệ thống con. Đây còn gọi là kiểu kho dữ liệu dùng chung.
- ✓ Mỗi hệ thống con duy trì một CSDL riêng của nó và truyền dữ liệu rõ ràng đến các hệ thống khác

Khi hệ thống có một khối lượng lớn dữ liệu được chia sẻ thì mô hình kho dữ liệu kho dữ liệu chia sẻ là tốt nhất. Mô hình kiến trúc dữ liệu tập trung chỉ ra tại hình



Ví dụ: Kiến trúc của bộ công cụ CASE



Các đặc tính của mô hình kho dữ liệu chia sẻ

✓ Các lợi thế

- Tiện lợi để chia sẻ một số lượng dữ liệu lớn;
- Các phân hệ xử lý không cần quan tâm đến cách thức dữ liệu được được quản lý ở bộ quản lý trung tâm như thế nào, ví dụ: hoạt động sao lưu phục hồi, an ninh, bảo mật, vv...
- Mô hình chia sẻ được công khai như giản đồ kho chúa.

✓ Các bất lợi

- Các phân hệ xử lý phải chấp nhận mô hình kho chứa dữ liệu. Không tránh khỏi sự thỏa hiệp;
- Cài tiến dữ liệu là khó và đắt đỏ;
- Không phát huy các chính sách quản lý dữ liệu cụ thể cho từng phân hệ;

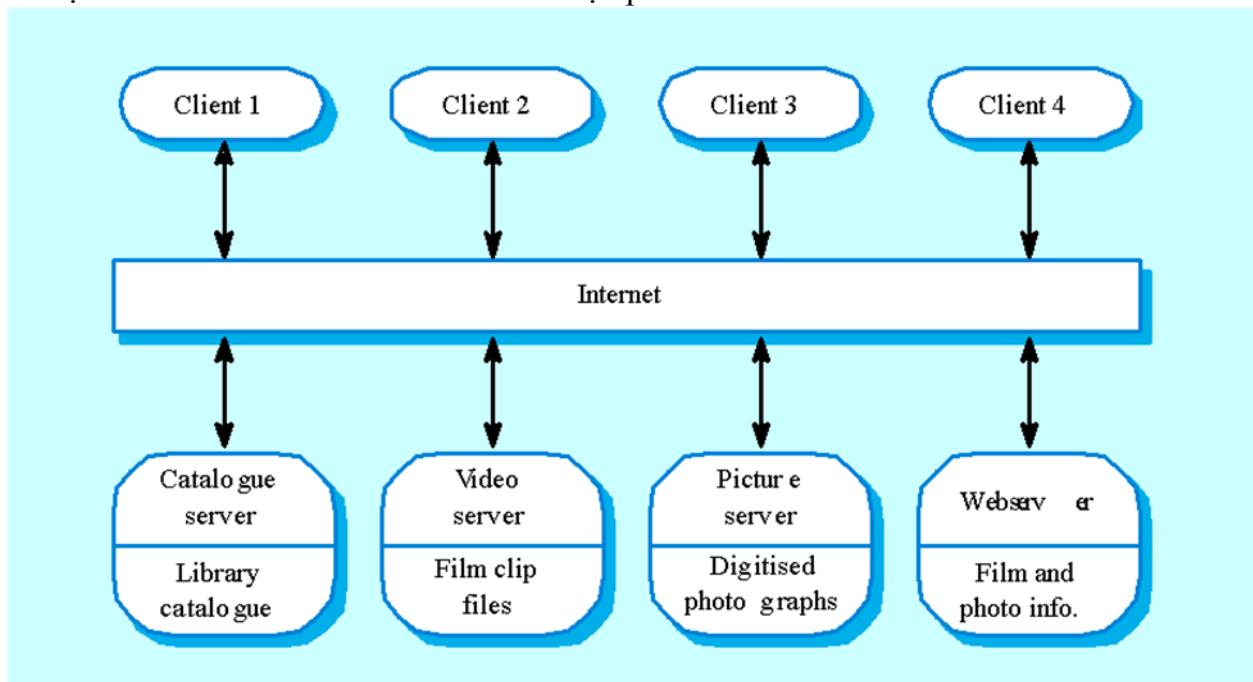
- Khó phân tán dữ liệu hiệu quả.

2. Kiến trúc Khách – Phục vụ

Kiến trúc Khách – Chỉ là mô hình hệ thống phân tán. Nó chỉ ra cách thức dữ liệu và các xử lý được phân tán trên một loạt các thành phần. Đây là mô hình kiến trúc đang được sử dụng phổ biến nhất. Mô hình này bao gồm ba thành phần chính:

- ✓ Một tập hợp các servers độc lập cung cấp các dịch vụ cụ thể: ví dụ in ấn, quản lý dữ liệu, vv...
- ✓ Một tập các Client gọi các dịch vụ này từ các servers.
- ✓ Mạng kết nối cho phép các Client truy cập đến các servers

Ví dụ: Mô hình Client – Server của thư viện phim và ảnh



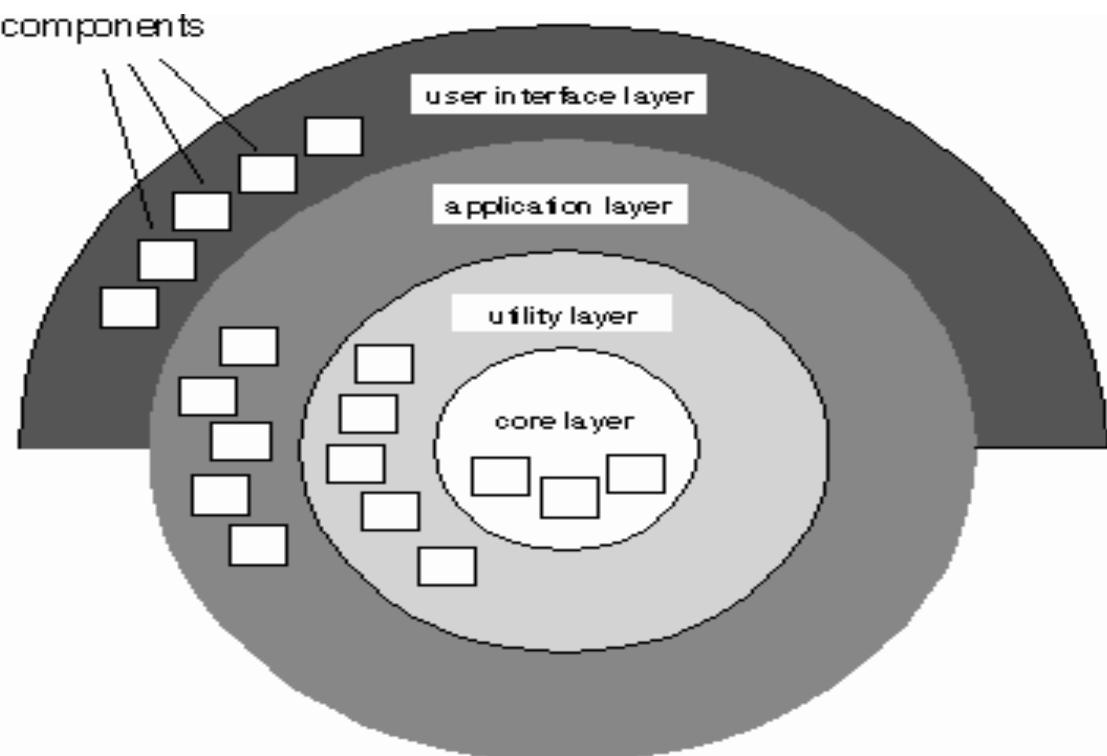
Các đặc tính của mô hình Client – Server

- ✓ *Lợi ích*
 - Phân tán dữ liệu trực tiếp
 - Sử dụng hiệu quả mạng, chi phí thiết bị rẻ hơn;
 - Dễ dàng mở rộng, nâng cấp, thêm dịch vụ
- ✓ *Bất lợi*

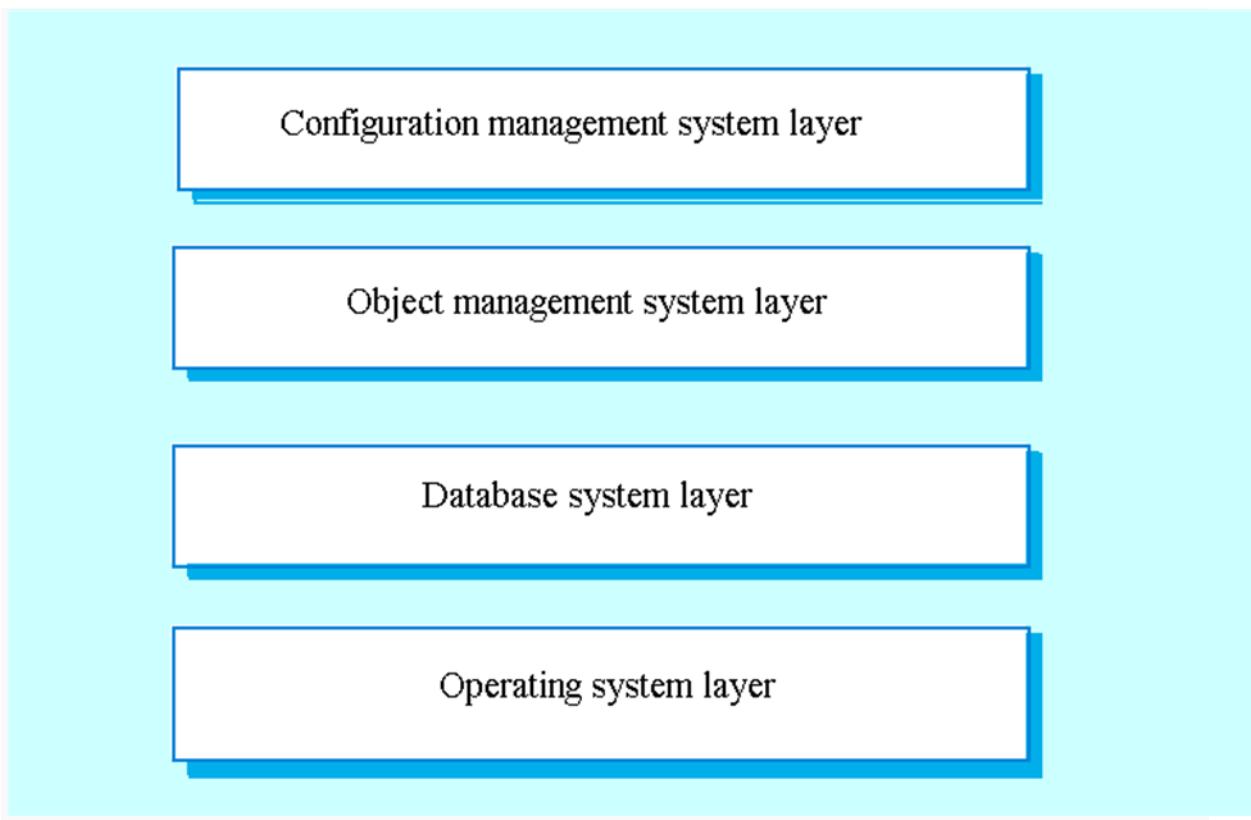
- Các hệ con dùng các Cấu trúc dữ liệu khác nhau, không chia sẻ được, việc trao đổi dữ liệu có thể không hiệu quả.
- Quản lý dữ liệu dư thừa trên mỗi server;
- Không lưu trữ chung tên và các dịch vụ - Có thể khó tìm các dịch vụ và kiểm tra dịch vụ có rồi không.

3.Kiến trúc phân tầng

Kiến trúc phân tầng dùng để mô hình hóa giao diện của các phân hệ. Nó tổ chức hệ thống thành các tầng (hoặc các máy trùu tượng). Mỗi tầng cung cấp một tập các dịch vụ. Mô hình kiến trúc này hỗ trợ sự phát triển tăng trưởng của các tầng. Khi giao diện của một tầng thay đổi, chỉ những tầng kế nó bị ảnh hưởng. Tuy nhiên, thường không tự nhiên để cấu trúc các hệ thống theo cách này, không phải hệ thống nào cũng dễ dàng phân chia theo kiến trúc này. Kiến trúc phân tầng được chỉ ra ở hình



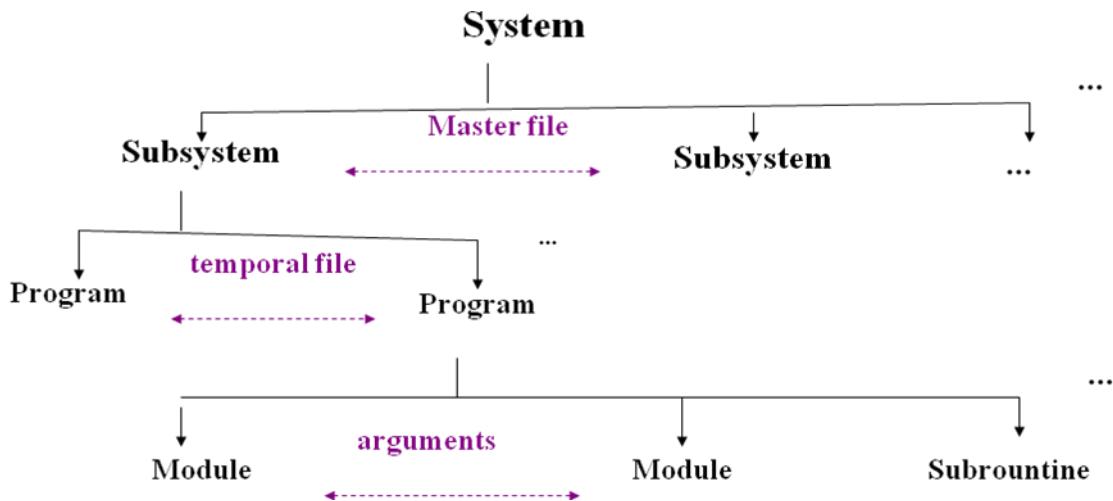
Ví dụ: Mô hình phân tầng của hệ thống quản lý phiên bản:



b. Phân rã hệ thống

Phân rã hệ thống thành các hệ con. Ở đây ta cần phân biệt sự khác nhau giữa hệ con và mô đun.

Kiến trúc hệ thống nhìn từ cấu trúc phân cấp có dạng như hìnhsau:



Ở đây ta cần phân biệt giữa hệ thống con và module:

- ✓ *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.

Các kiểu phân chia module:

Kiểu phân chia module là kiểu phân chia các hệ thống con thành các modules. Thông thường không có sự phân biệt cứng nhắc giữa tổ chức hệ thống và phân chia module. Hai kiểu mô hình phân chia module được sử dụng thông dụng là:

-
- ✓ Mô hình đối tượng:
 - Hệ thống được phân chia thành các đối tượng tương tác;
 - ✓ Mô hình đường ống hoặc mô hình dữ liệu:
 - Hệ thống được phân chia thành các module chức năng để chuyển đổi các đầu vào thành cách đầu ra.
-

c. Điều khiển hệ thống

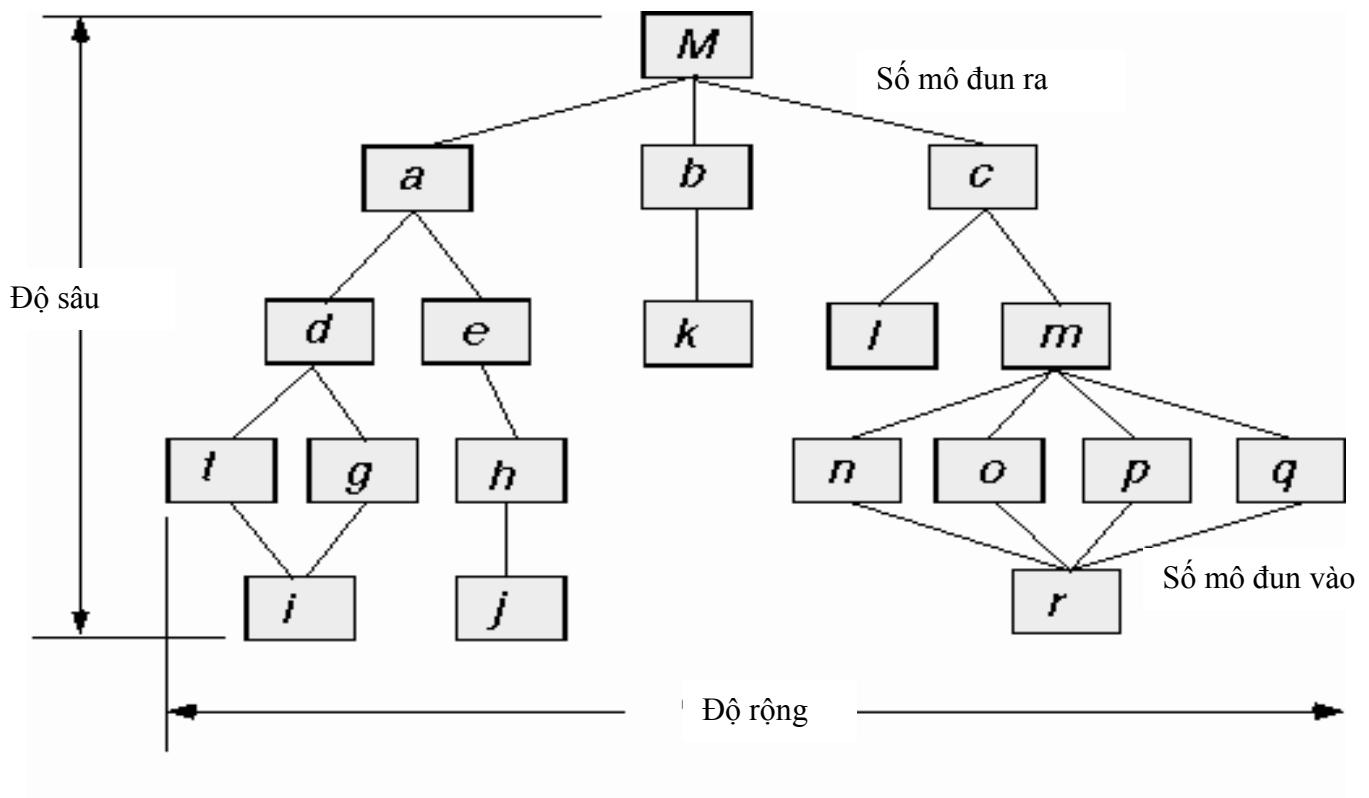
Điều khiển hệ thống là hoạt động xác lập mô hình điều khiển giữa các phần khác nhau của hệ thống đã được xác định. Điều khiển hệ thống liên quan đến luồng điều khiển giữa các hệ thống con. Có hai kiểu điều khiển:

- ✓ **Điều khiển tập trung:** Một hệ thống chịu trách nhiệm điều khiển, khởi tạo và dừng sự thực hiện của các hệ thống con khác.
- ✓ **Điều khiển dựa trên sự kiện:** Mỗi hệ thống con có thể phản ứng lại với các sự kiện phát sinh từ các hệ thống con khác hoặc từ môi trường của hệ thống.

Điều khiển tập trung

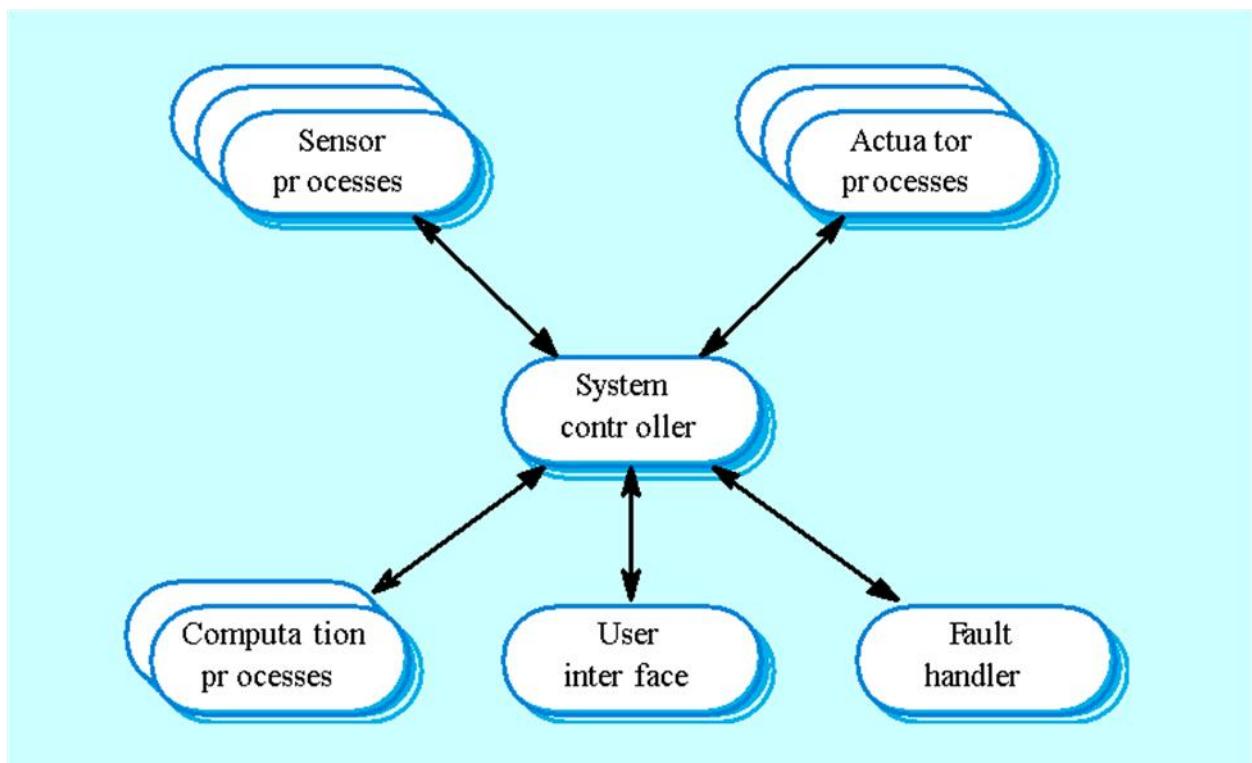
Hệ thống con điều khiển chịu trách nhiệm quản lý việc thực hiện của các hệ thống con khác. Hai mô hình kiến trúc điều khiển tập trung được sử dụng khá phổ biến:

- ✓ *Mô hình kiến trúc gọi - trả lời (call-return architecture):* Là mô hình thủ tục top – down: Việc điều khiển bắt đầu ở đỉnh của cây thủ tục và di chuyển xuống. Mô hình này có thể áp dụng có các hệ thống tuần tự. Mô hình này được biểu diễn như hình



- ✓ *Mô hình quản lý*: Một thành phần hệ thống được thiết kế như một hệ quản trị để điều khiển việc dừng, khởi động và phối hợp các xử lý của hệ thống khác. Có thể cài đặt các hệ thống tuần tự như một lệnh CASE. Mô hình này có thể áp dụng cho các hệ thống song song.

Ví dụ: Mô hình quản lý của hệ thống điều khiển thời gian thực



Điều khiển hướng sự kiện

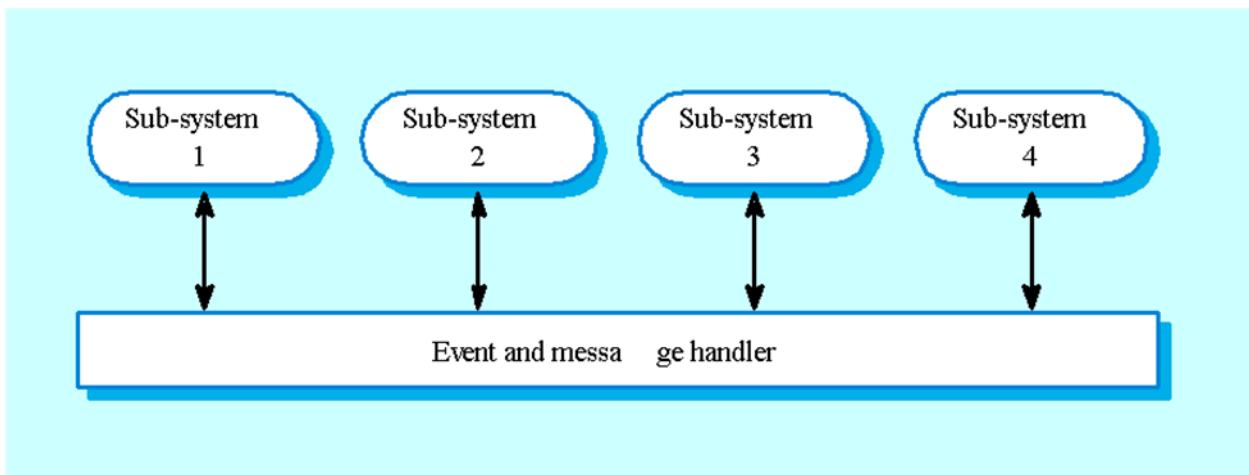
Các hệ thống điều khiển dựa trên sự kiện được điều khiển bởi các sự kiện phát sinh từ bên ngoài hệ thống. Các hệ thống này thường sử dụng hai mô hình điều khiển chính:

- ✓ **Các mô hình thông báo:**

- Khi một sự kiện được thông báo đến tất cả các hệ thống con. Hệ thống con nào điều khiển sự kiện sẽ thực hiện.
- Mô hình này hiệu quả trong việc tích hợp các hệ thống con trên các máy tính khác nhau trong một mạng. Các hệ thống con đăng ký sự kiện cụ thể mình quan tâm. Khi các sự kiện này xảy ra, điều khiển được chuyển đến hệ thống con mà có thể điều khiển sự kiện. Chính sách điều khiển không được gắn với bộ điều khiển thông điệp và sự kiện. Các hệ

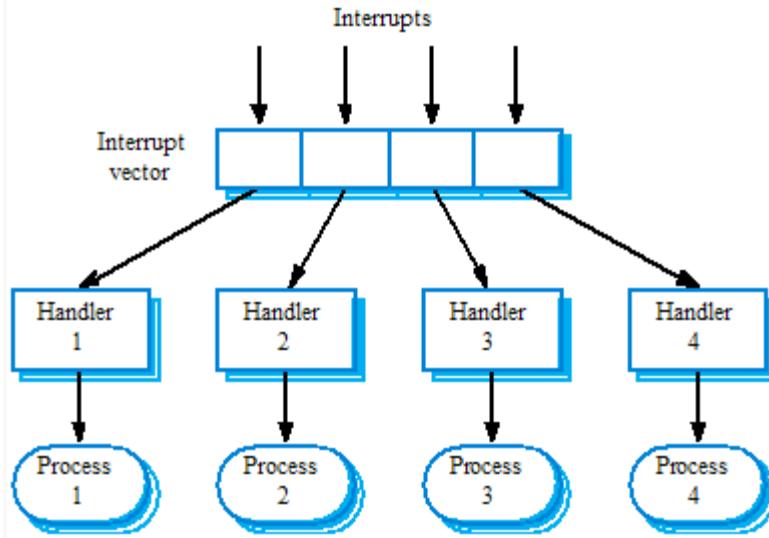
thống con tự quyết định có điều khiển các sự kiện mà chúng quan tâm hay không. Tuy nhiên, mô hình này có hạn chế là các hệ thống phát sự kiện không biết liệu sự kiện đó có được xử lý hay không bởi các hệ thống khác.

- Mô hình thông báo được biểu diễn:



✓ **Các mô hình điều khiển ngắn:**

- Được sử dụng trong các hệ thống thời gian thực. Ở đó các ngắt được phát hiện bởi bộ điều khiển ngắn và truyền đến một số thành phần khác để xử lý. Mô hình này gồm các kiểu ngắn với một bộ điều khiển được định nghĩa cho mỗi kiểu. Mỗi kiểu ngắn được kết hợp với một vị trí bộ nhớ và công tắc cứng làm chuyển đổi bộ điều khiển của nó. Mô hình này cho phép phản ứng nhanh như lập trình phức tạp và khó thám định.
- Mô hình này được biểu diễn như sau:



Hình 7.6: Mô hình điều khiển lan truyền

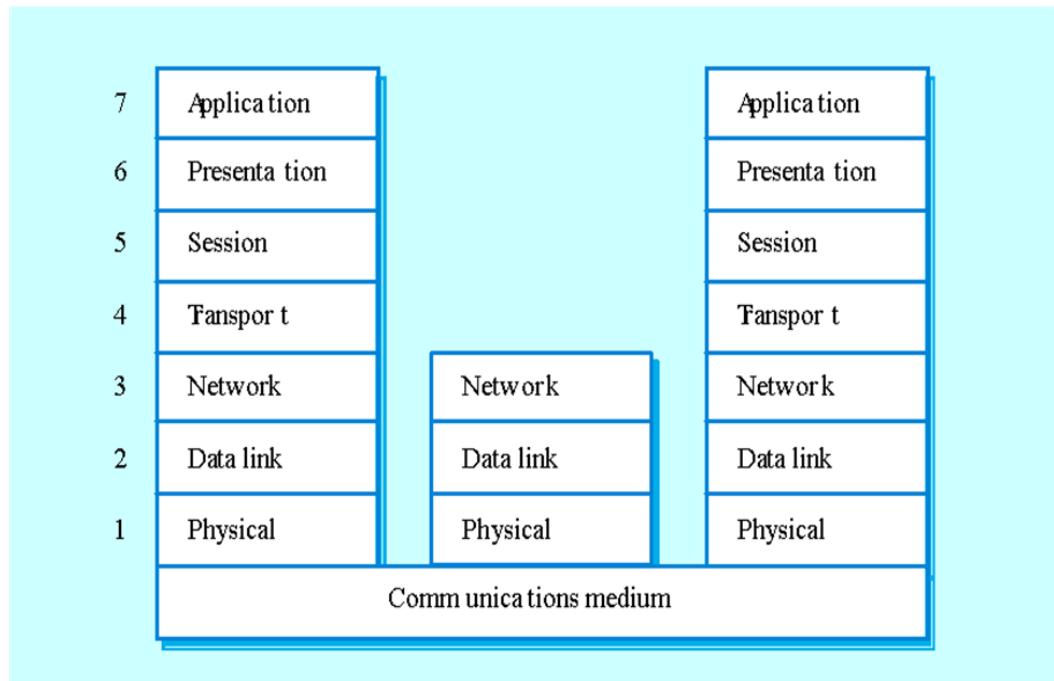
c. Các kiến trúc tham chiếu

Mặc dù các hệ thống có cùng một miền ứng dụng có khác nhau về chi tiết nhưng kiến trúc chung của nó khá giống nhau. Do đó, khi xây dựng một hệ thống mới, ta có thể sử dụng lại kiến trúc của hệ thống tương tự đã tồn tại.

Một mô hình kiến trúc được đặc tả cho kiến trúc chung cho các hệ thống thuộc cùng một miền ứng dụng. Hai kiểu mô hình kiến trúc đặc tả miền khá thông dụng:

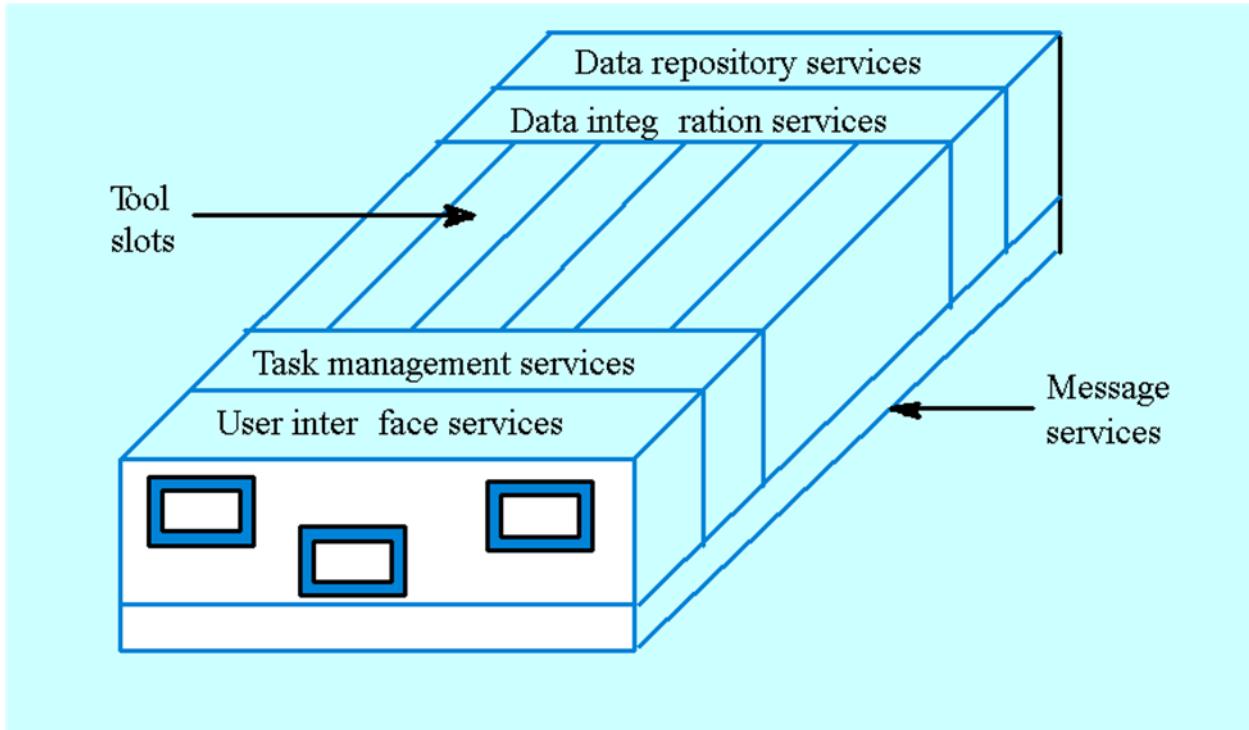
- ✓ **Mô hình chung:** là sự trừu tượng hoá từ một số các hệ thống thực và bao hàm các thuộc tính quan trọng của các hệ thống này. Mô hình chung là mô hình bottom-up.
- ✓ **Mô hình tham chiếu:**

- 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.
- Ví dụ 1: Mô hình OSI là một hình phân tầng cho sự giao tiếp các hệ thống – là một kiến trúc tham chiếu.



Ví dụ 2: Mô hình tham chiếu CASE. Mô hình này gồm:

- ✓ Các dịch vụ kho chứa dữ liệu: Lưu trữ và quản lý các mục dữ liệu.
- ✓ Các dịch vụ tích hợp dữ liệu: Quản lý các nhóm thực thể.
- ✓ Các dịch vụ quản lý nhiệm vụ: Định nghĩa và ban hành các mô hình tiến trình
- ✓ Các dịch vụ thông điệp: Giao tiếp giữa tool-tool và tool-environment
- ✓ Các dịch vụ giao diện người dùng: Phát triển giao diện người dùng

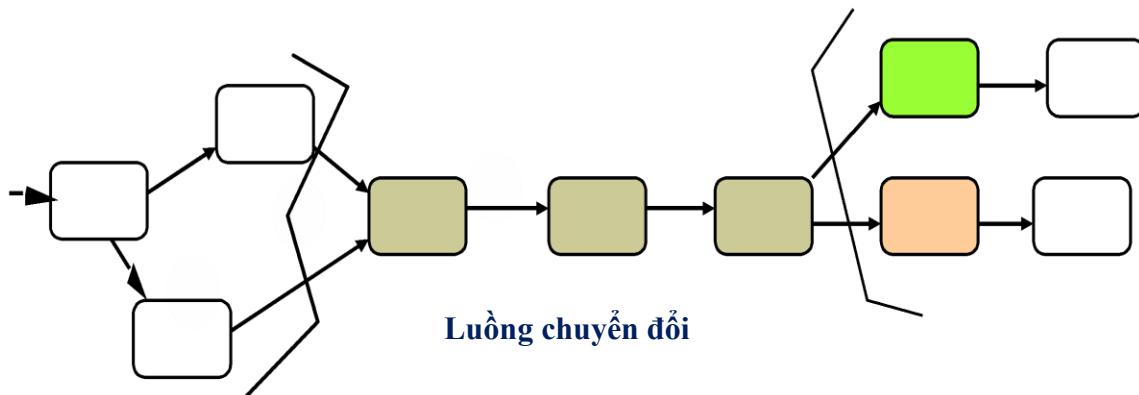


5.2.9 Xây dựng kiến trúc chương trình từ các biểu đồ luồng dữ liệu

a. Thiết kế cấu trúc chương trình

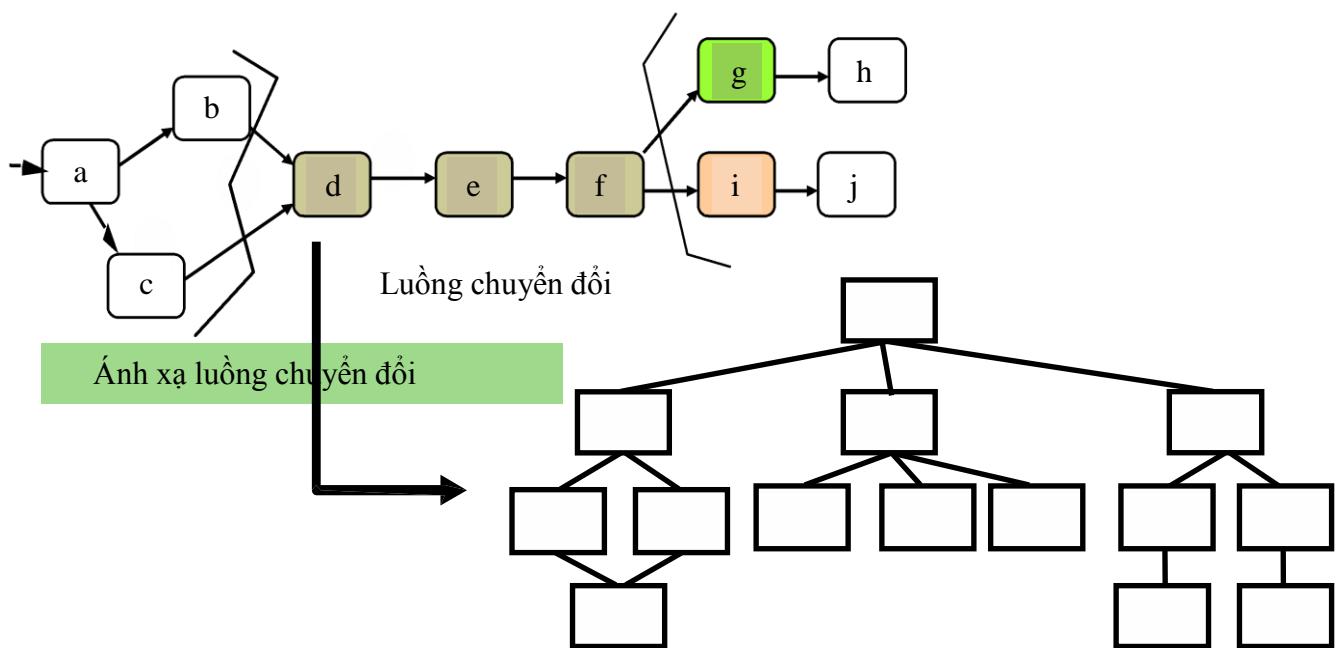
Thiết kế cấu trúc chương trình với mục tiêu tạo ra module có kiến trúc tốt. Mô đun có kiến trúc tốt nếu nó được phân hoạch hợp lý, và các module được liên kết qua điều khiển. Để thiết kế cấu trúc chương trình, ta thường sử dụng cách tiếp cận chuyển đổi (ánh xạ/mapping) biểu đồ luồng dữ liệu (DFD) thành kiến trúc chương trình. Các luồng dữ liệu được chia làm hai loại đó là luồng chuyển đổi (xử lý tập trung) và luồng giao dịch (định tuyến phân phối).

Luồng chuyển đổi: Xử lý tập trung

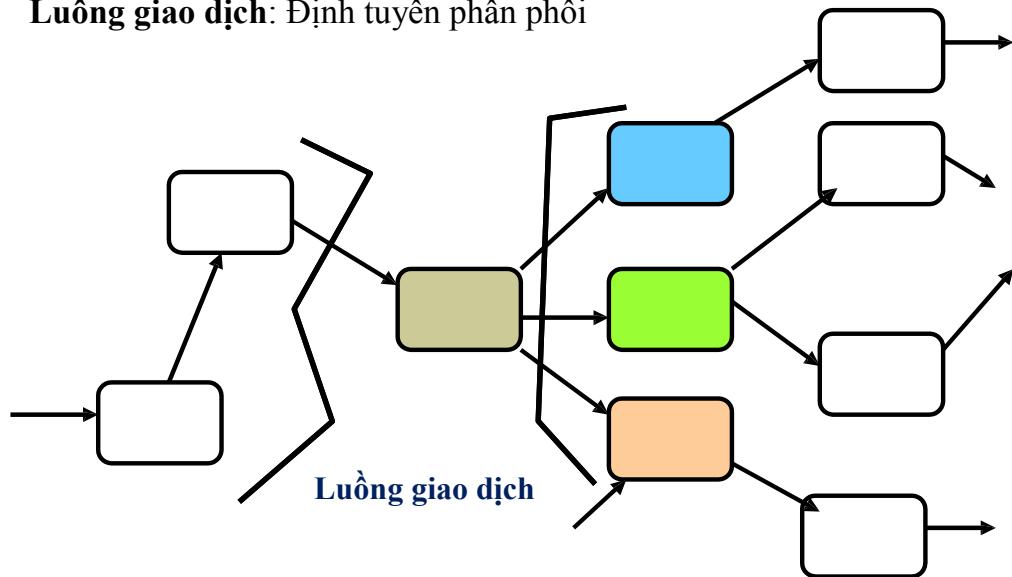


- ❖ *Luồng chuyển đổi tập trung* là luồng mà có các dữ liệu đầu vào **được tập trung xử lý ở một số tiến trình** rồi cho kết quả đầu ra là các tiến trình thực hiện lưu trữ, truyền đi hay biểu diễn thông tin
- ❖ *Các bước xây dựng kiến trúc chương trình từ luồng chuyển đổi*
 - ✓ Cô lập, xác định biên của các module vào/ra, xác định module **xử lý tập trung**.
 - ✓ Chuyển chúng thành các module kiến trúc tương ứng
 - ✓ Thêm các module điều khiển nếu cần thiết
 - ✓ Tinh chỉnh kiến trúc (vi chỉnh - refining) kiến trúc để nâng cao tính module

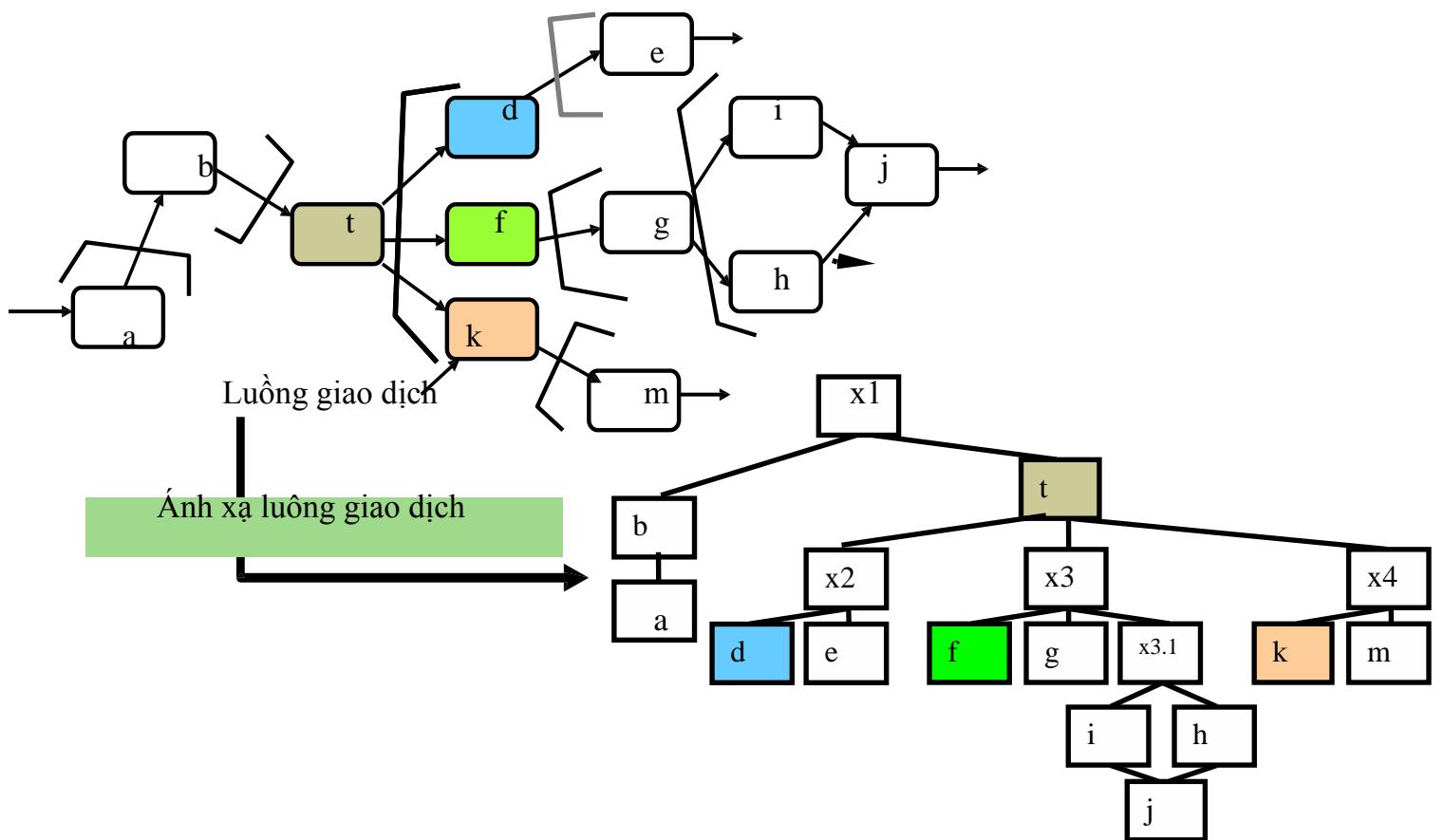
Ví dụ: Kiến trúc luồng chuyển đổi



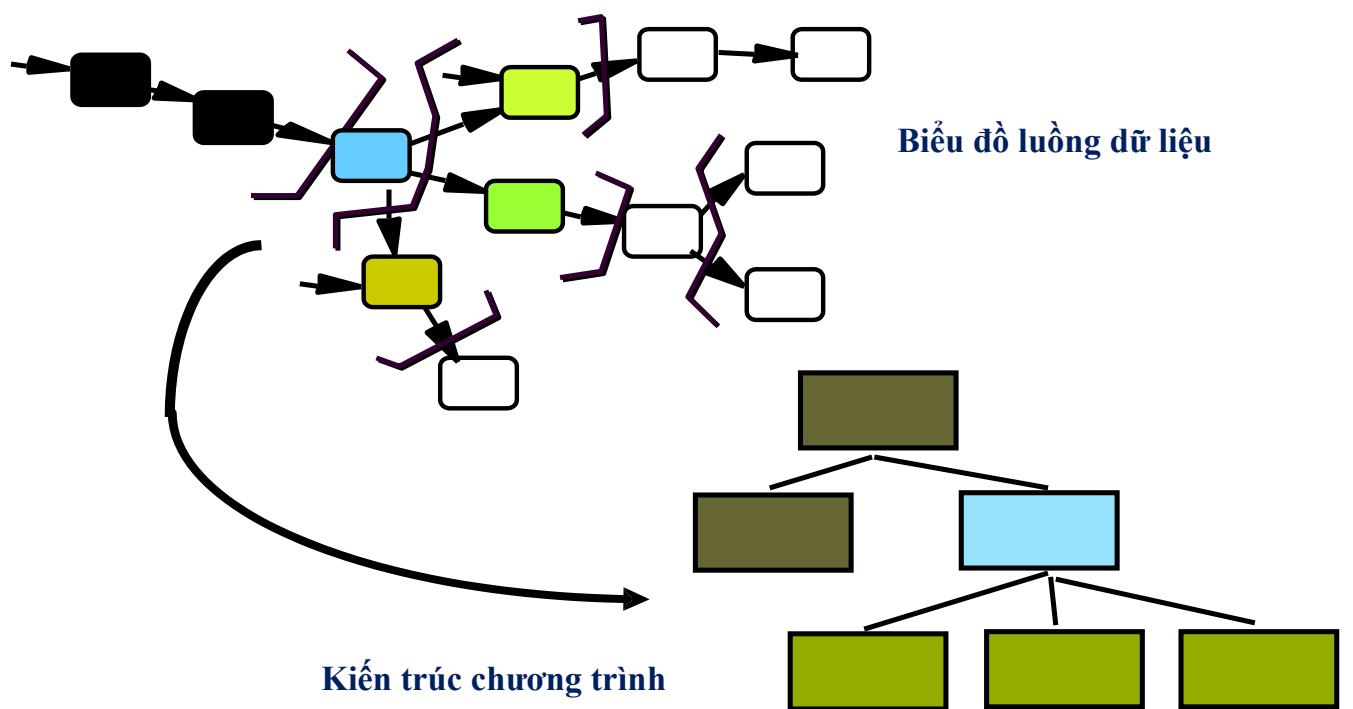
Luồng giao dịch: Định tuyến phân phối



- ❖ *Luồng giao dịch:* là luồng mà **mỗi đầu vào được nhận dạng và chuyển** cho các tiến trình xử lý tương ứng với nó. **Tiến trình** làm nhiệm vụ nhận dạng và chuyển dữ liệu đến nơi cần gọi là **trung tâm giao dịch**.
- ❖ *Các bước xây dựng kiến trúc chương trình từ luồng giao dịch*
 - ✓ Xác định các luồng vào
 - ✓ Xác định các luồng thực hiện
 - ✓ Xác định trung tâm giao dịch (mô đun phân phối)
 - ✓ Biến đổi riêng rẽ từng luồng hành động.
- ❖ *Ví dụ:* Kiến trúc luồng giao dịch

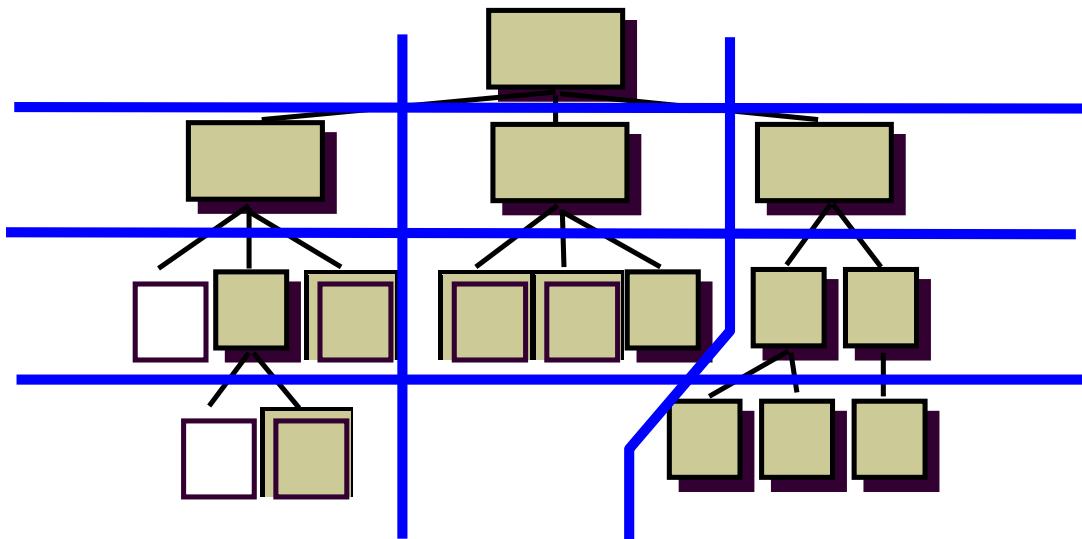


Kiến trúc chương trình được xây dựng từ DFD như minh họa ở hình

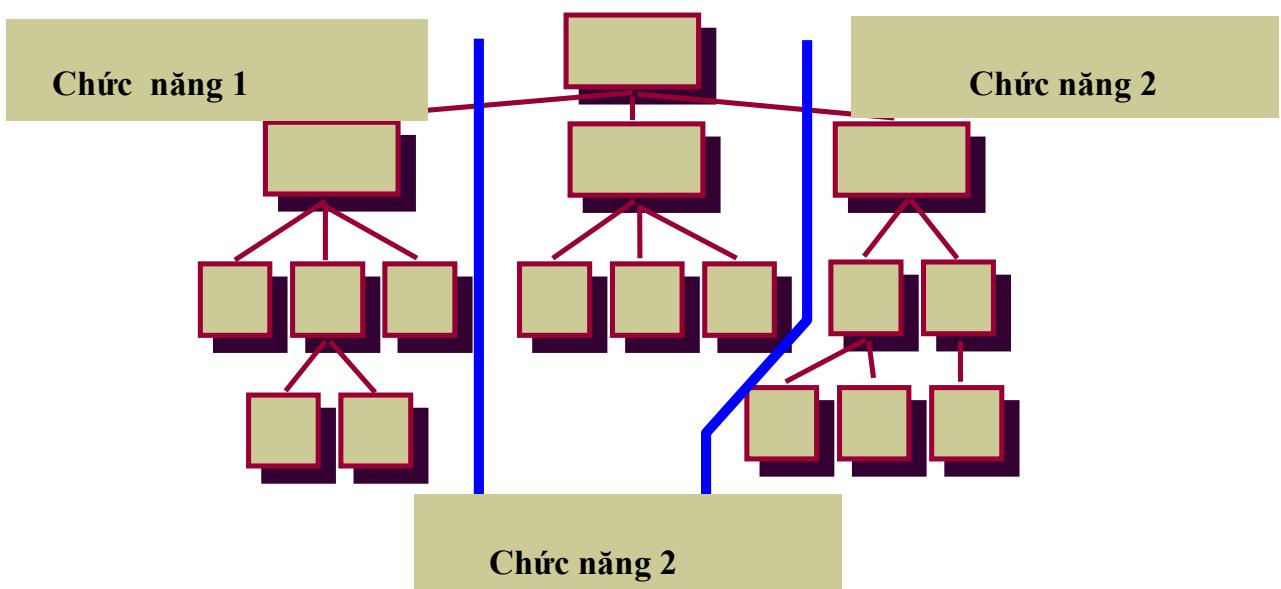


b. Phân hoạch kiến trúc

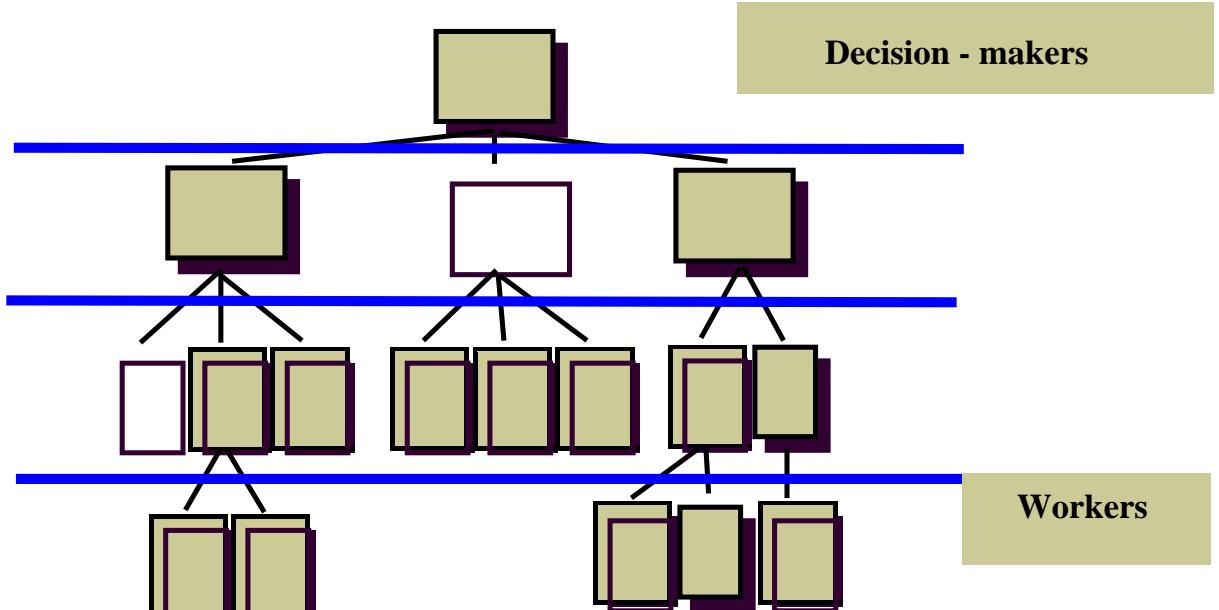
Cần phân hoạch kiến trúc theo chiều ngang và dọc



Phân hoạch kiến trúc dọc: Xác định các nhánh rẽ riêng cho các chức năng chủ chốt.
Sử dụng các module điều khiển để điều phối thông tin giữa các chức năng.



Phân hoạch kiến trúc ngang: Phân tầng các module thành từng mức. Mức ra quyết định (điều khiển) và module thao tác. Các module ra quyết định cần được xếp ở tầng cao



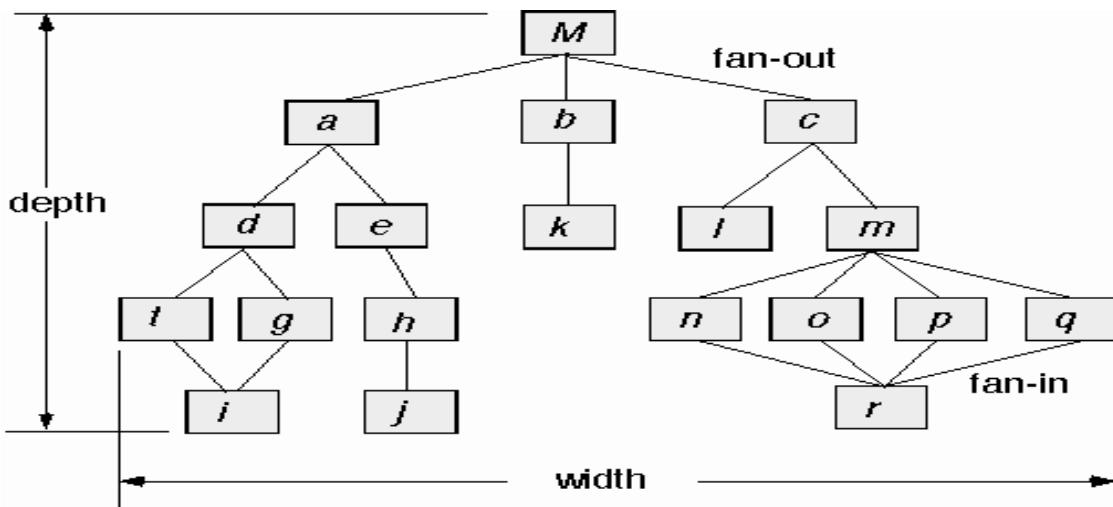
Tại sao phải phân hoạch kiến trúc: Phân hoạch kiến trúc giúp chúng ta tạo ra những phần mềm đảm bảo:

- ✓ Dễ kiểm thử
- ✓ Dễ bảo trì
- ✓ Hạn chế hiệu ứng phụ khi sửa đổi
- ✓ Dễ mở rộng.

5.2.10 Thiết kế hướng đối tượng

a. Những vấn đề gặp phải của thiết kế hướng thủ tục

Kiến trúc phần mềm truyền thống được chỉ ra trong hình....



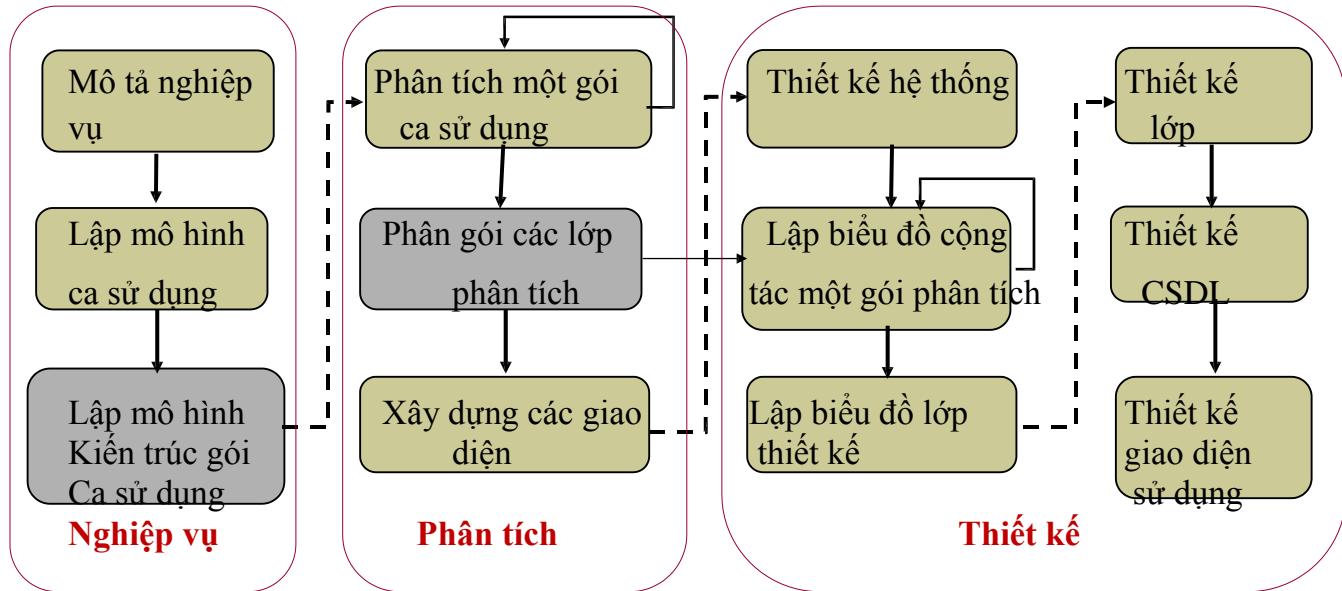
Thiết kế thủ tục gắp những vấn đề sau:

- ✓ Dữ liệu là chung cho cả hệ thống
 - Mọi thủ tục thao tác trên CSDL chung, đặc trưng cho trạng thái toàn hệ thống
 - Thao tác sai của một thủ tục lén dữ liệu gây sai lan truyền sang phần tử khác sử dụng dữ liệu này
 - Sửa đổi một thủ tục có nguy cơ ảnh hưởng tới phần khác liên quan.
- ✓ Thay đổi cấu trúc dữ liệu dẫn đến thay đổi tổng thể hệ thống -> dữ liệu cần tổ chức tốt
- ✓ Hệ thống lớn, phức tạp: bảo trì khó khăn.

b. Phân tích và Thiết kế hướng đối tượng

Các mô hình phân tích hướng đối tượng	Các mô hình thiết kế hướng đối tượng
<ul style="list-style-type: none"> ✓ Mô hình nghiệp vụ <ul style="list-style-type: none"> ○ Mô hình miền ○ Biểu đồ hoạt động ✓ Mô hình ca sử dụng ✓ Mô hình lớp phân tích ✓ Mô hình gói lớp 	<ul style="list-style-type: none"> ✓ Mô hình cấu trúc gói ✓ Mô hình cộng tác ✓ Mô hình lớp ✓ Đặc tả lớp, giao diện

Tiến trình phân tích và thiết kế hướng đối tượng chỉ ra ở hình ...



5.3 Thiết kế giao diện người dùng

5.3.1 Vai trò, tầm quan trọng của giao diện người - máy

Khi xây dựng một hệ thống phần mềm, chúng ta phải luôn nhớ một nguyên tắc quan trọng đó là: Người sử dụng không quan tâm đến cấu trúc bên trong của hệ thống là 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 cả hệ thống; cho dù hệ thống đó có đáp ứng tất cả các chức năng nghiệp vụ mà họ muốn. Giao diện là quan trọng vì những lý do sau đây:

- + *Giao diện là phương tiện để người dùng sử dụng hệ thống :*
 - Giao diện thiết kế nghèo nàn, người dùng dễ mắc lỗi
 - Giao diện thiết kế tồi là lý do nhiều phần mềm không được sử dụng.
- + *Giao diện trợ giúp người dùng làm việc với khả năng của họ*
 - Giao diện trợ giúp tốt, người dùng thành công
 - Giao diện trợ giúp tồi, người dùng khó khăn -> thất bại
- ❖ *Một giao diện tốt nên được thiết kế:*
 - ✓ Hướng đến người dùng
 - ✓ Che dấu các chi tiết kỹ thuật bên trong
 - ✓ Kết hợp 3 mặt người dùng, chức năng và công nghệ

5.3.2 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 các đặc điểm chung của họ liên quan đến giao diện hệ thống: Một số đặc điểm cơ bản:

- ✓ *Khả năng nhớ tức thời của con người bị hạn chế*: con người chỉ có thể nhớ ngay khoảng 7 mục 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 ...

5.3.3 Các nguyên tắc thiết kế giao diện

1. Cân phản ánh vào thiết kế giao diện:
 - a. Kinh nghiệm, năng lực, nhu cầu người dùng
 - i. Khả năng dùng bàn phím, chuột
 - ii. Tốc độ phản ứng, khả năng nhớ.
 - b. Sở thích, văn hóa, lứa tuổi: Màu sắc, ngôn ngữ, ...
 - c. Những hạn chế về mặt vật chất và tinh thần của người dùng (trí nhớ, vụng về, ... có thể mắc lỗi)
2. Luôn bao gồm việc làm bản mẫu để người dùng đánh giá.
3. Giao diện cần có các tính chất sau đây:
 - a. *Tính thân thiện*: thuật ngữ, khái niệm, thói quen, trình tự nghiệp vụ của người dùng

- b. *Tính nhất quán*: Vị trí hiển thị, câu lệnh, thực đơn, biểu tượng, màu sắc, định dạng.
- c. *Ít gây ngạc nhiên*
- d. *Có cơ chế phục hồi* tình trạng trước lỗi
- e. *Cung cấp kịp thời phản hồi và trợ giúp mọi lúc, mọi nơi*
- f. *Tiện ích tương tác đa dạng*: Nên hỗ trợ nhiều loại tương tác cho nhiều loại người sử dụng khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị. Tương tác giữa người sử dụng và hệ thống được chia thành 5 loại sau:
 - Thao tác trực tiếp
 - Lựa chọn menu, chọn biểu tượng
 - Điền vào biểu mẫu (Form)
 - Ngôn ngữ ra lệnh
 - Ngôn ngữ tự nhiên

Bảng sau đây chỉ rõ các ưu và nhược điểm của từng kiểu tương tác giữa người dùng và hệ thống:

Interaction style	Main advantages	Main disadvantages	Application examples
Direct manipulation	Fast and intuitive interaction Easy to learn	May be hard to implement. Only suitable where there is a visual metaphor for tasks and objects.	Video games CAD systems
Menu selection	Avoids user error Little typing required	Slow for experienced users. Can become complex if many menu options.	Most general-purpose systems
Form fill-in	Simple data entry Easy to learn Checkable	Takes up a lot of screen space. Causes problems where user options do not match the form fields.	Stock control, Personal loan processing
Command language	Powerful and flexible	Hard to learn. Poor error management.	Operating systems, Command and control systems
Natural language	Accessible to casual users Easily extended	Requires more typing. Natural language understanding systems are unreliable.	Information retrieval systems

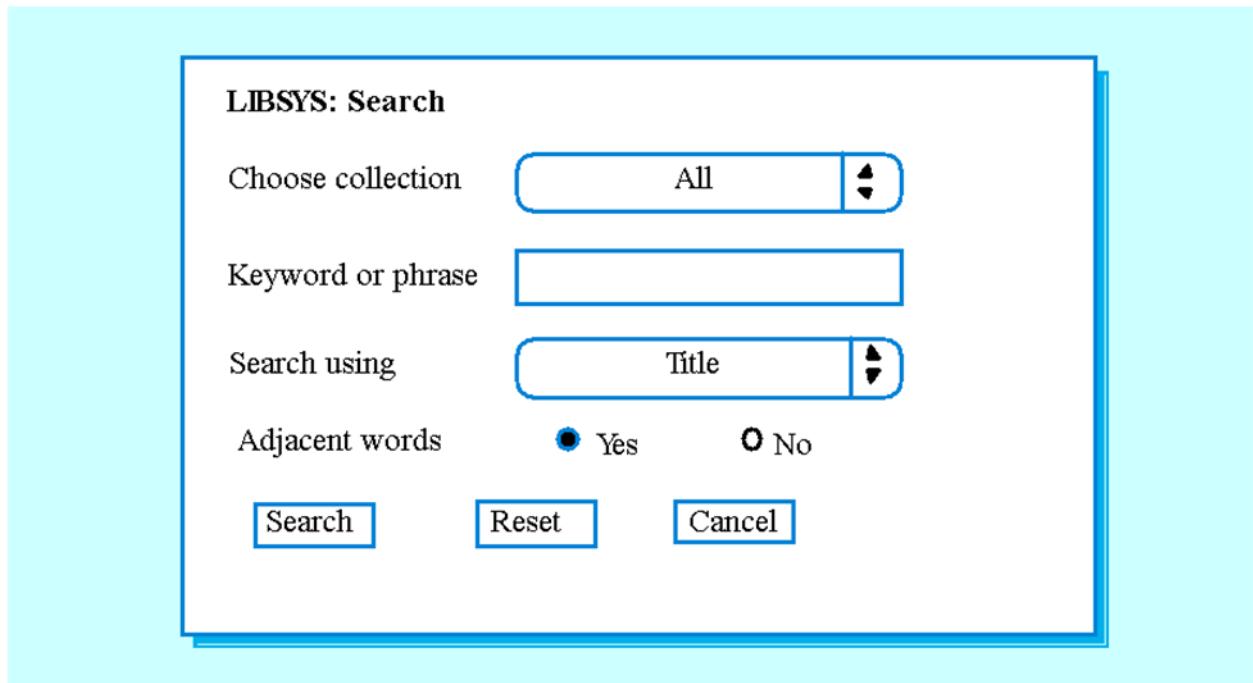
5.3.4 Các vấn đề trong thiết kế giao diện UI

Hai vấn đề phải được giải quyết trong thiết kế các hệ thống tương tác:

- ✓ Thông tin nhập bởi người dùng nên được cung cấp cho hệ thống máy tính như thế nào, hay ta phải thiết kế các giao diện nhập liệu như thế nào.
- ✓ Thông tin sau khi xử lý bởi hệ thống máy tính nên được trình bày đến người dùng như thế nào, hay ta phải thiết kế các giao diện hiển thị dữ liệu như thế nào.

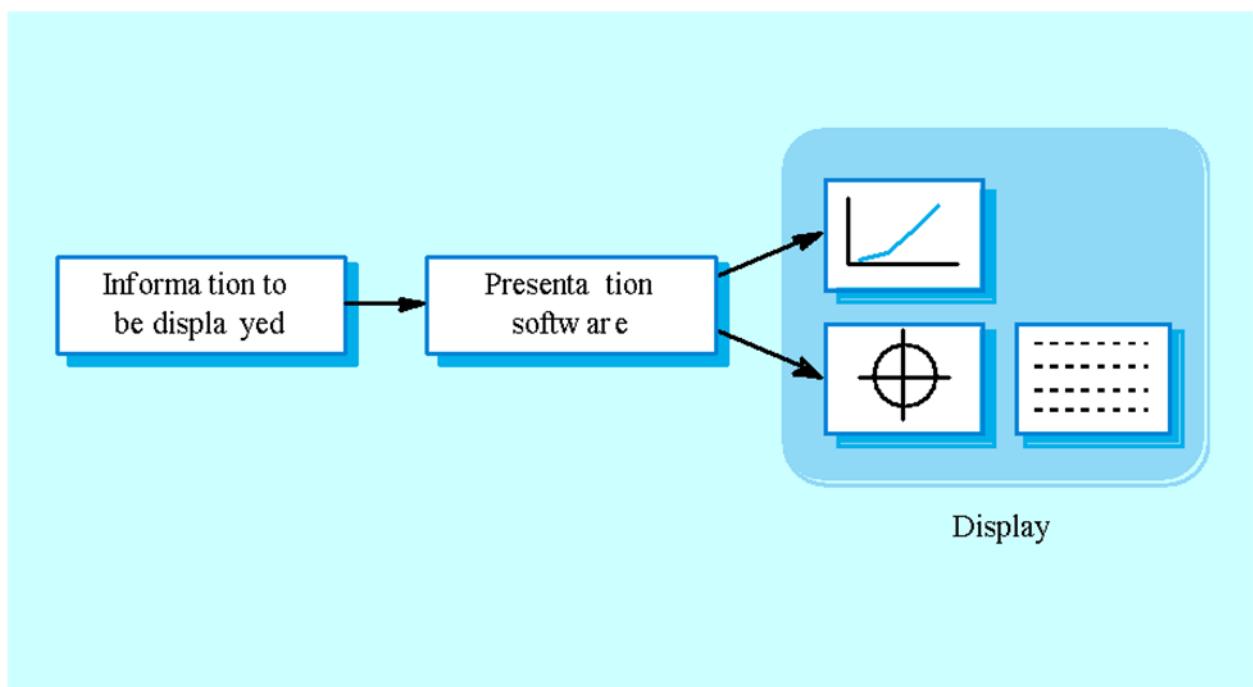
Tương tác người dùng và *biểu diễn* thông tin có thể được tích hợp qua một framework kết dính ẩn sau giao diện người dùng.

Ví dụ: Giao diện nhập liệu cho LIBSYS

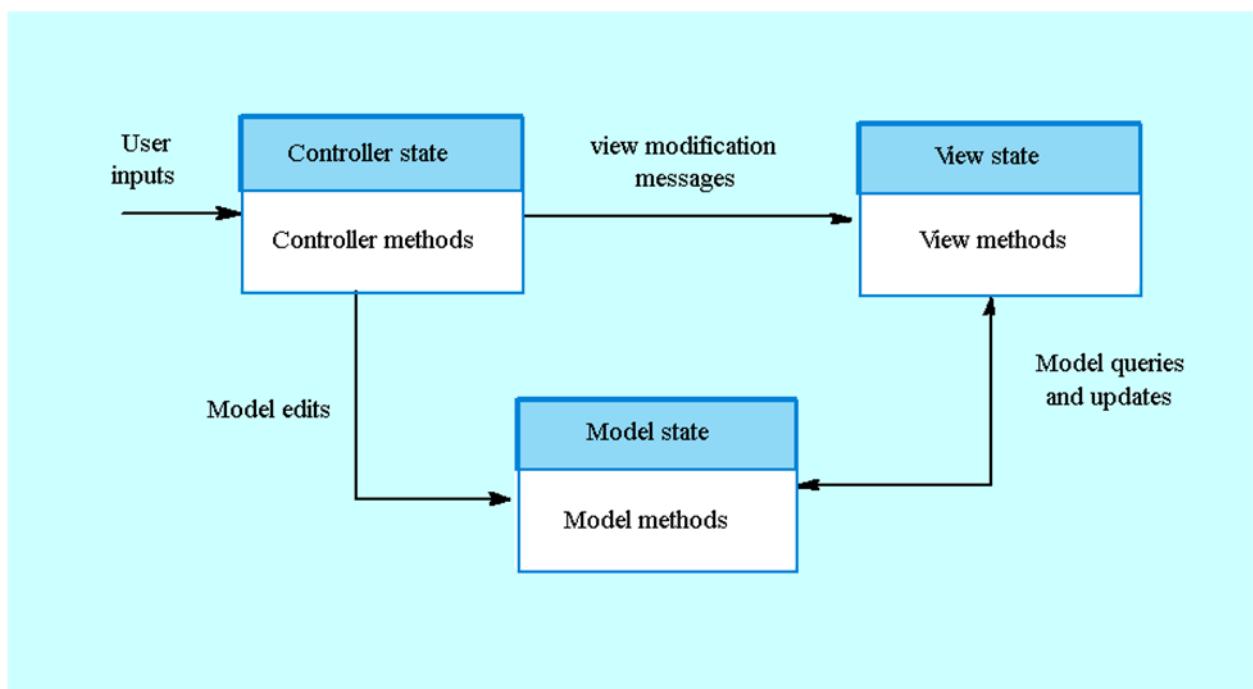


5.3.5 Biểu diễn thông tin

Biểu diễn thông tin liên quan tới việc hiển thị các thông tin sau khi được xử lý bởi 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 họa, âm thanh, mô hình,



Cách tiếp cận Model-View-Controller là một cách hỗ trợ các biểu diễn nhiều dữ liệu:



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 thể hiện để người sử dụng có thể quan sát.

Các yếu tố cần quan tâm khi thiết kế các giao diện hiển thị thông tin:

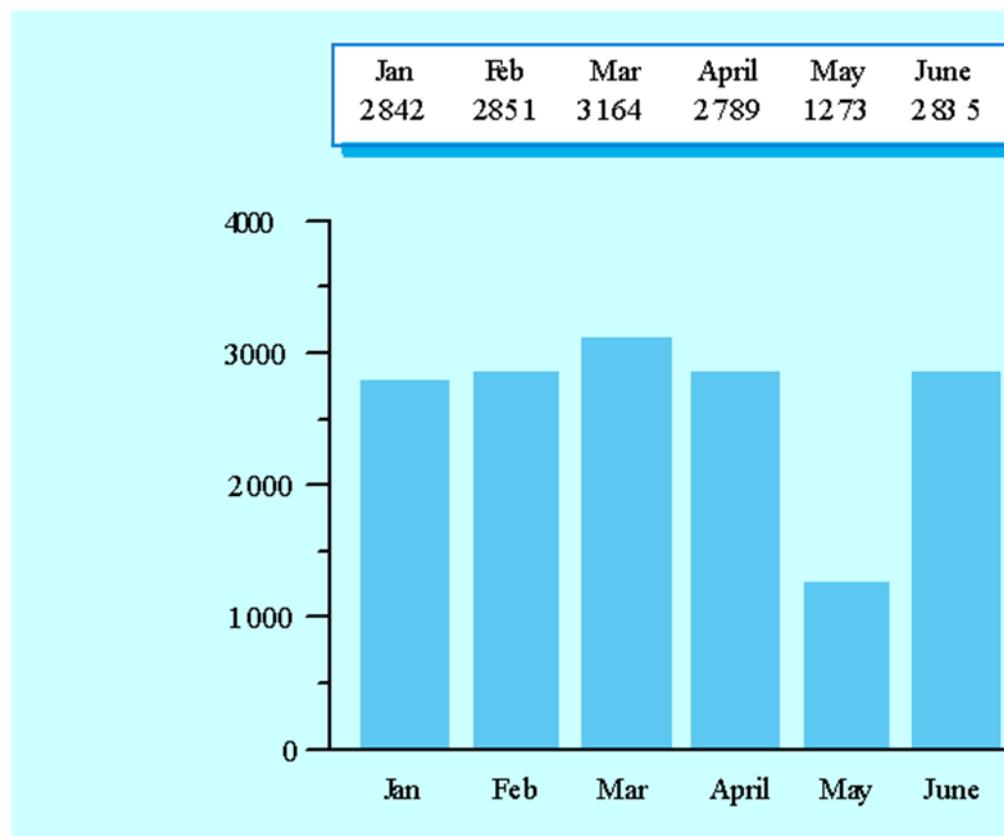
Khi biểu diễn thông tin trên giao diện, cần quan tâm đến việc trả lời các câu hỏi sau:

- ✓ Người quan tâm đến thông tin chính xác hay các mối quan hệ dữ liệu?
- ✓ Các 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 một số hành động để phản ứng lại 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?

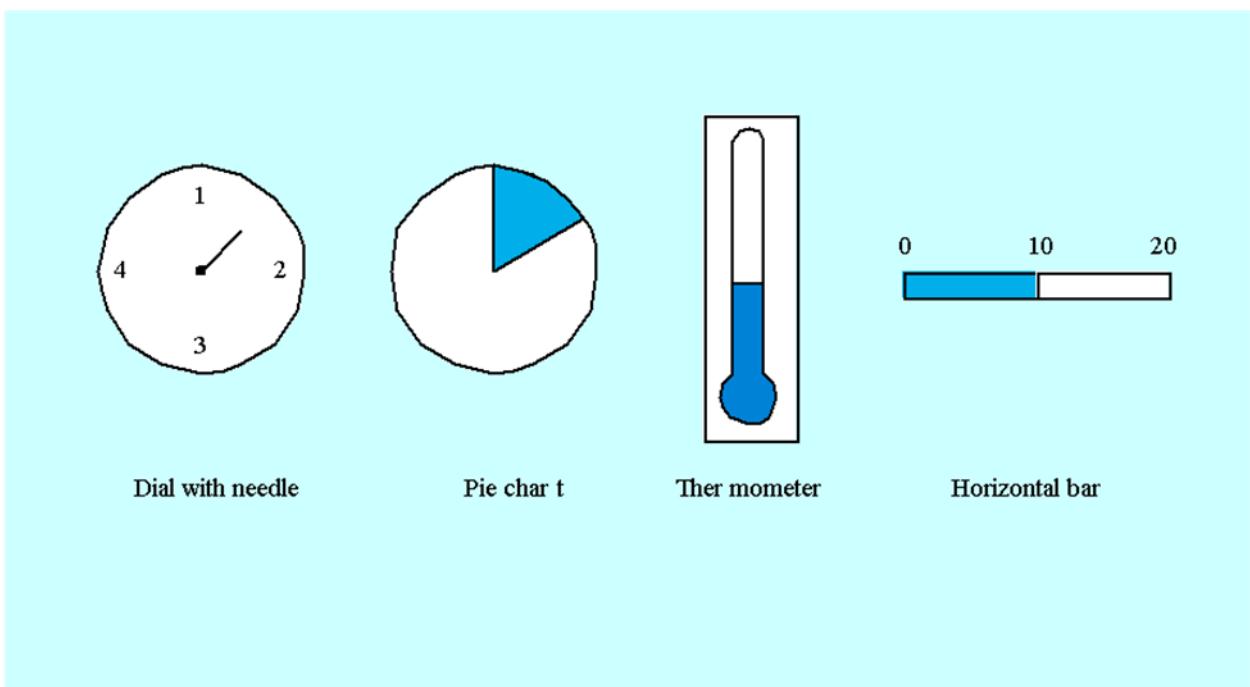
- ✓ Biểu diễn digital hay analogue?
 - ❖ Biểu diễn Digital
 - Súc tích – Chiếm ít không gian màn hình;
 - Biểu diễn các giá trị chính xác.
 - ❖ Biểu diễn Analogue
 - Dễ tạo cảm giác xem qua giá trị;
 - Có thể chỉ ra các giá trị quan hệ;
 - Dễ hơn cho việc xem xét các giá trị dữ liệu ngoại lệ.

Một số ví dụ về biểu diễn thông tin:

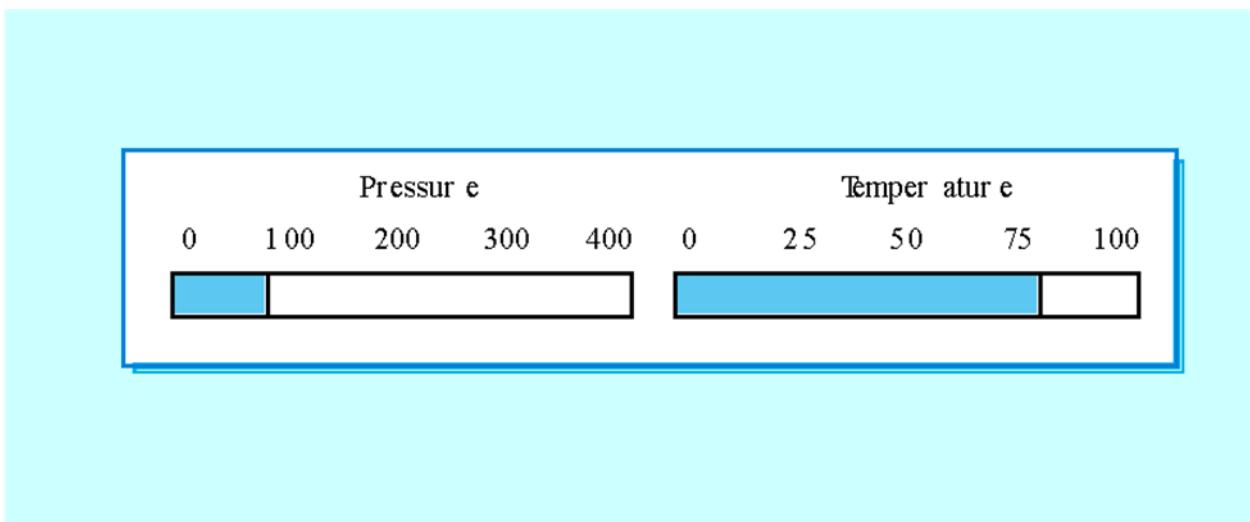
Các biểu diễn thông tin thay thế:



Các phương pháp biểu diễn:



Hiển thị các giá trị liên quan:



Trực quan hóa dữ liệu hiển thị

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hóa dữ liệu. Trực quan hóa có thể giúp ta 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, các trang web được biểu diễn như một cây thứ bậc,

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 khi sử dụng màu:

- ✓ 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 có găng thực hiện.
- ✓ Sử dụng màu một cách thống nhất và cẩn thận.
- ✓ Cẩn thận khi sử dụng các cặp màu.

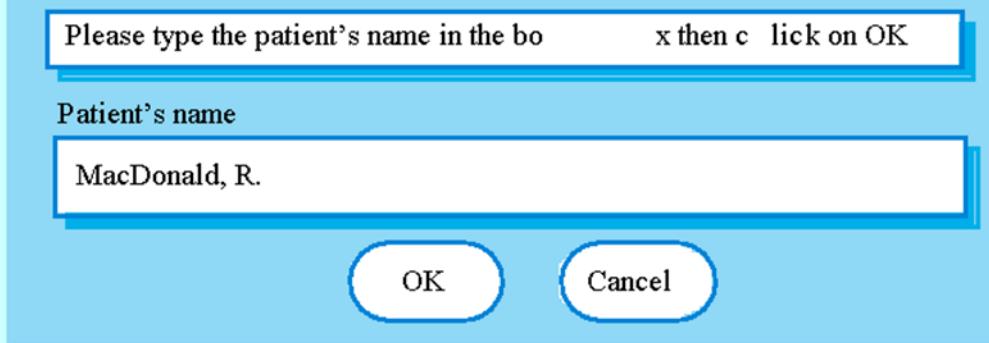
Thiết kế các thông báo lỗi:

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, việc thiết kế các 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. Các thông báo lỗi nên ngắn gọn, xúc tích, thống nhất và có cấu trúc. Kiến thức và kinh nghiệm của người dùng là nhân tố quyết định trong thiết kế thông điệp. Các yếu tố cần quan tâm khi thiết kế các thông điệp:

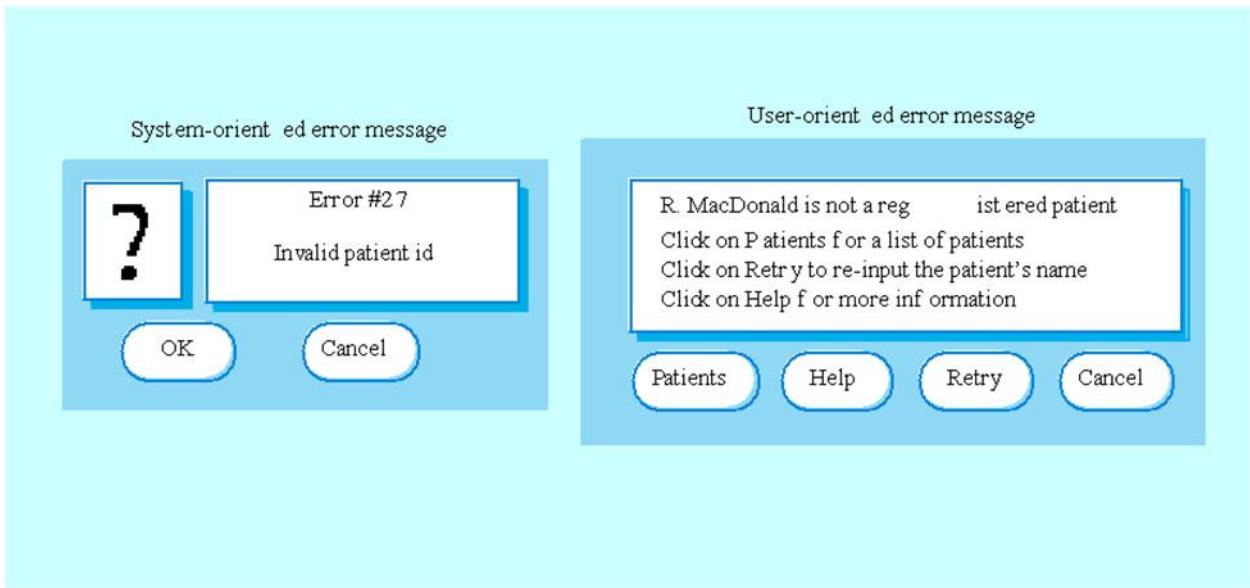
Factor	Description
Context	Wherever possible, the messages generated by the system should reflect the current user context. As far as is possible, the system should be aware of what the user is doing and should generate messages that are relevant to their current activity.
Experience	As users become familiar with a system they become irritated by long, ÓnaningfulÓ messages. However, beginners find it difficult to understand short terse statements of a problem. You should provide both types of message and allow the user to control message conciseness.
Skill level	Messages should be tailored to the userÓskills as well as their experience. Messages for the different classes of user may be expressed in different ways depending on the terminology that is familiar to the reader.
Style	Messages should be positive rather than negative. They should use the active rather than the passive mode of address. They should never be insulting or try to be funny.
Culture	Wherever possible, the designer of messages should be familiar with the culture of the country where the system is sold. There are distinct cultural differences between Europe, Asia and America. A suitable message for one culture might be unacceptable in another.

Ví dụ: Thông điệp báo lỗi do người dùng gây ra

Giả sử y tá gõ sai tên bệnh nhân muốn tìm kiếm:

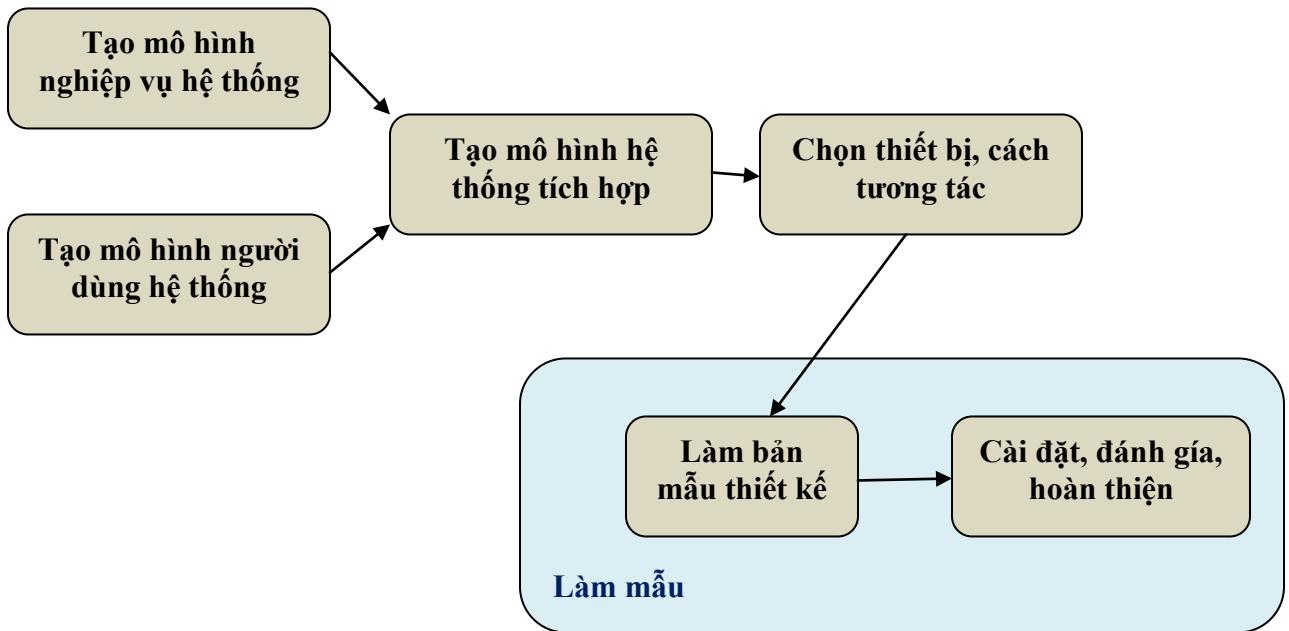


Thiết kế thông báo tốt và không tốt:



5.3.6 Tiến trình thiết kế giao diện chung

Thiết kế giao diện là một tiến trình lặp, các hoạt động trong tiến trình được lặp lại nhiều lần. Những người dùng hệ thống và những nhà thiết kế giao diện cùng tham gia thực hiện tiến trình này. Tiến trình thiết kế giao diện chung được chỉ ra tại hình

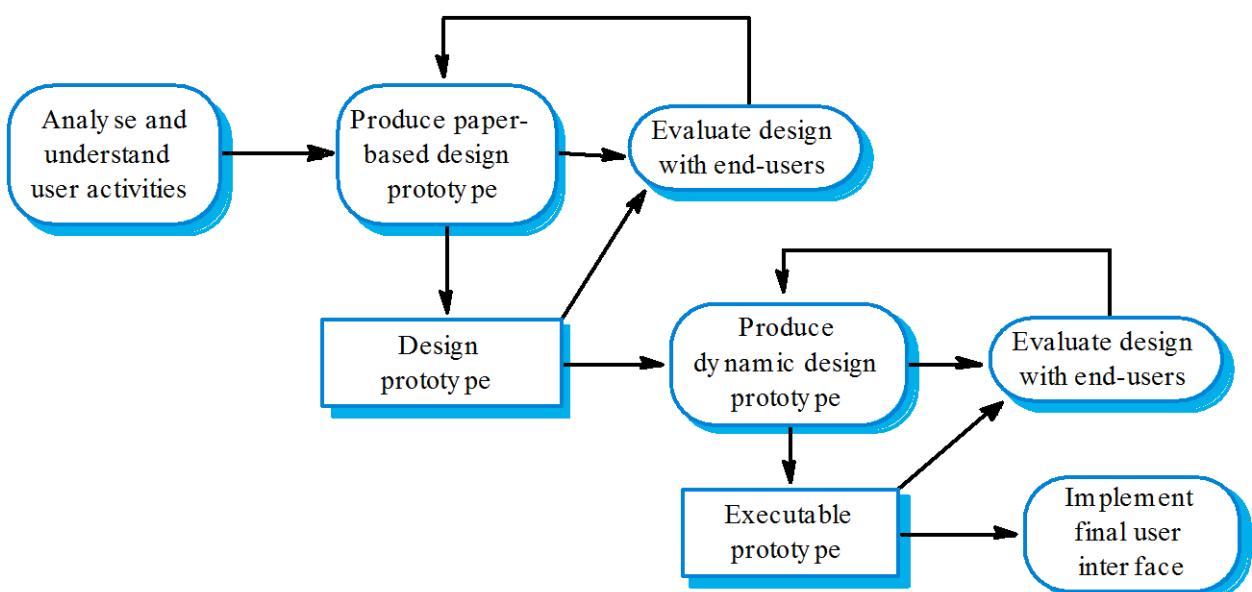


Các thiết bị tương tác thường gặp như màn hình, bàn phím, chuột, bút tử, màn hình cảm biến, Mic/Speaker, các thẻ thông minh, bóng xoay, ...

Tiến trình thiết kế giao diện làm mẫu bao 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*: Thủ nghiệm các mẫu thử cùng với người sử dụng.

Hình ... mô tả các hoạt động trong tiến trình thiết kế giao diện người máy.



a. Phân tích và hiểu các hoạt động người dùng

Nếu chúng ta không hiểu những gì người dùng muốn làm với hệ thống, ta sẽ không có khả năng thực tế để thiết kế một giao diện hiệu quả. Do đó, hoạt động này đóng vai trò quan trọng trong tiến trình thiết kế giao diện. Các kịch bản mô tả những tình tiết cụ thể của người dùng tác động lên hệ thống, là một trong các cách mô tả các phân tích này. Những phân tích người dùng phải được mô tả theo các thuật ngữ mà người dùng và những người thiết kế khác có thể hiểu được chúng.

Ví dụ về kịch bản mô tả tương tác giữa người dùng và hệ thống

Jane là một sinh viên trong các sv sùng đạo/tôn giao đang thực hiêu một tiểu luận về kiến trúc Ấn Độ và cách thức nó bị ảnh hưởng bởi các thực tiễn tôn giáo. Để trợ giúp cho sự hiều biết này, cô ấy muốn xem một số bức tranh cụ thể về các tòa nhà nhưng cô ấy không thể tìm được bất cứ bức trang nào trong thư viện riêng của mình.

Cô ấy tiếp cận nhân viên thư viện để trao đổi những nhu cầu của mình và anh ấy gợi ý một số thuật ngữ tìm kiếm có thể dùng. Anh ấy cũng gợi ý một số thư viên trong Delhi và London mà có thể chứa tài liệu này vì chúng được nhập vào trong các danh mục của thư viện nên có thể tìm kiếm chúng sử dụng một số thuật ngữ tìm kiếm này. Jane đã tìm một số nguồn tài liệu và yêu cầu các bản photo các bức tranh này và các chi tiết kiến trúc đã được đưa trực tiếp đến Jane.

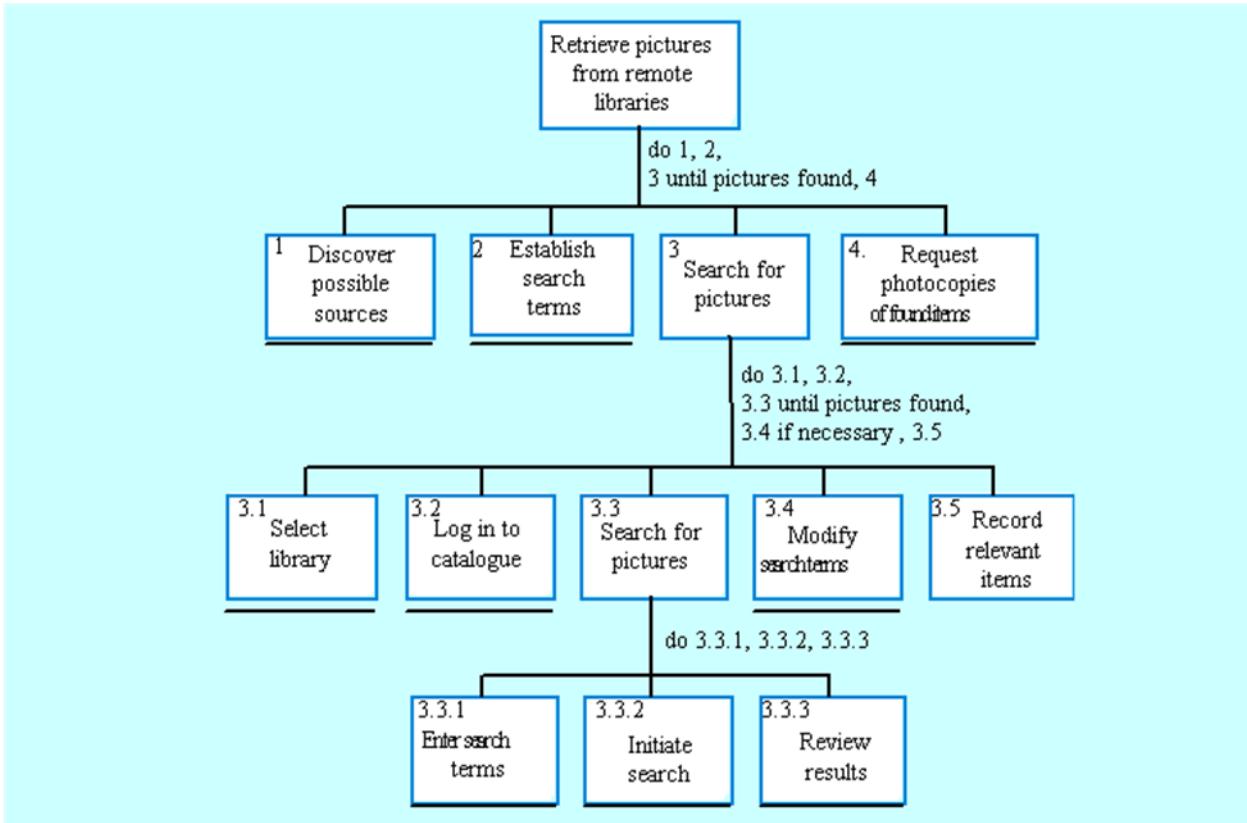
Từ các ngữ cảnh sử dụng đã mô tả, chúng ta có thể lấy được rất nhiều yêu cầu của người sử dụng về giao diện hệ thống. Xét ví dụ trên, chúng ta xác định được các yêu cầu sau:

- ✓ Người dùng có thể không biết các thuật ngữ tìm kiếm phù hợp, do đó cần có cách giúp họ lựa chọn các thuật ngữ cần tìm kiếm.
- ✓ Người dùng phải có thể lựa chọn những bộ sưu tập để tìm kiếm trong đó
- ✓ Người dùng phải có thể thực hiện được thao tác tìm kiếm và đưa ra yêu cầu sao chép các tài liệu liên quan.

Các kỹ thuật phân tích người dùng

- ✓ *Phân tích nhiệm vụ*: Mô hình hóa đầy đủ các bước cần thực hiện để hoàn thành một nhiệm vụ. Chúng ta thường phân tích nhiệm vụ theo thứ bậc.

Ví dụ: Xét nhiệm vụ tra cứu, tìm kiếm các bức ảnh trong các thư viện từ xa



- ✓ *Phỏng vấn:* Hỏi người dùng về công việc 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.
- ✓ *Quan sát:* Quan sát người dùng làm việc để xem những gì người dùng thực hiện và hỏi họ về cách họ làm việc. phương pháp này khá hiệu quả vì nhiều nhiệm vụ người dùng thuộc về trực giác và họ nhận thấy nhiều khó khăn khi mô tả và giải thích chúng. Phương pháp này cũng giúp chúng ta hiểu được vai trò của các tác động tổ chức và xã hội lên công việc.

b. Lập mẫu thử giao diện người dùng

Mục đích của lập mẫu thử giao diện người dùng là cho phép người dùng có được những kinh nghiệm thực tế với giao diện. Nếu không có những kinh nghiệm này 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:
 - ❖ Cho phép người dùng thực hiện các hoạt động qua các kịch bản mà sử dụng các bức họa về giao diện.
 - ❖ Sử dụng bảng tường thuật để biểu diễn một chuỗi các tương tác với hệ thống.
 - ❖ Mẫu thử trên giấy là một cách hiệu quả để có được các tương tác người dùng.
- ✓ Tinh chỉnh mẫu thử và xây dựng chúng

Các kỹ thuật lập mẫu thử:

- ✓ Mẫu thử hướng kịch bản: sử dụng công cụ như Macromedia Director để xây dựng một tập hợp các kịch bản 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 hỗ trợ thiết kế giao diện để thiết kế nhanh các mẫu thử giao diện. Ví dụ như ngôn ngữ Visual Basic, Delphi,
- ✓ Mẫu thử dựa Internet: sử dụng trình duyệt Web và các kịch bản kết hợp.

c. Đánh giá giao diện người dùng

Hoạt động này nhằm xác định khả năng phù hợp của các giao diện đối với người dùng. Tuy nhiên, việc đánh giá trên phạm vi rộng tốn khá nhiều chi phí và không thể thực hiện được đối với hầu hết các hệ thống. Một cách lý tưởng, mỗi giao diện nên

được đánh giá dựa trên bản đặc tả khả năng sử dụng Tuy nhiên, hiếm khi các đặc tả như thế được tạo.

Các thuộc tính về khả năng sử dụng của giao diện

Attribute	Description
Learnability	How long does it take a new user to become productive with the system?
Speed of operation	How well does the system response match the user's work practice?
Robustness	How tolerant is the system of user error?
Recoverability	How good is the system at recovering from user errors?
Adaptability	How closely is the system tied to a single model of work?

5.4 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 lưu trữ dữ liệu:

- ✓ Đảm bảo tính đơn giản của cấu trúc lưu trữ,
- ✓ Trong một số trường hợp dùng 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
<ul style="list-style-type: none">✓ Các quan hệ chuẩn✓ Các khóa✓ Các tham chiếu✓ Các cấu trúc thao tác dữ liệu	<ul style="list-style-type: none">✓ Các file✓ Các kiểu dữ liệu✓ Kích cỡ

5.5 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 lũy,...

BÀI TẬP CHƯƠNG 5

6.1 Giới thiệu

Một sản phẩm phần mềm được xem là tốt nếu nó được phân tích tốt, thiết kế tốt, lập trình tốt và kiểm thử chặt chẽ. Lập trình là một công đoạn trong việc phát triển phần mềm và nó được tiến hành sau hoạt động thiết kế. Khi lập trình phong cách lập trình và các đặc trưng của ngôn ngữ lập trình ảnh hưởng lớn đến chất lượng chương trình.

Phong cách lập trình: Phong cách tốt cho ta chương trình tốt. Một chương trình tốt mang lại các lợi thế sau:

- ✓ Dễ hiểu mã, dễ bảo trì
- ✓ Giảm chi phí xây dựng, bảo trì, cải tiến
- ✓ Mã nguồn có chất lượng

Chọn ngôn ngữ cho ứng dụng: Chọn ngôn ngữ cho ứng dụng đóng vai trò rất quan trọng. Khi đã làm việc trên ứng dụng ta không có sự lựa chọn về ngôn ngữ. Nếu ban đầu chọn sai ngôn ngữ thì ta phải liên tục sửa đổi yêu cầu để phù hợp với những giới hạn của ngôn ngữ. Việc lựa chọn ngôn ngữ lập trình phải xem ngôn ngữ lập trình đó có phù hợp với kiểu ứng dụng hay không và xem nó có phù hợp với việc dùng để phát triển ứng dụng.

6.2 Các yêu cầu đối với lập trình viên

Lập trình là công việc của nhóm lập trình viên. Mỗi lập trình viên cần:

- ✓ Có năng lực cá nhân
- ✓ Hiểu biết các công cụ lập trình & các phương pháp lập trình
- ✓ Nắm vững các nguyên lý lập trình
- ✓ Có kinh nghiệm

Lập trình viên tốt là người đảm bảo chương trình:

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

6.3 Tiết hóa 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.

6.3.1 *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, ...

6.3.2 *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 módun 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,...

6.3.3 *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 trị 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
<i>Begin</i> <i>NhapDL();</i> <i>XulyDL();</i> <i>XuatDL();</i> <i>End.</i>	Chương trình là một biểu thức/hàm <i>XuatDL(XulyDL(NhapDL(...)))</i>

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

6.3.4 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#

6.3.5 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, ...

6.3.6 Kỹ thuật lập trình thẻ 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.

6.4 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 phần mềm
- ✓ *Các đặc trưng của ngôn ngữ*
 - **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
 - **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.
 - 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 mớiCó các công cụ trợ giúp hiệu quả: editor, debugger, linker, make...IDE (Integrated Develop Environment).
- ✓ **Năng lực, kinh nghiệm của nhóm lập trình viên**
 - Chọn NN mà lập trình viên làm chủ

- ✓ Miền ứng dụng của ngôn ngữ
 - Phần mềm hệ thống: Chương trình hiệu quả, vạn năng, dễ mở rộng.
Ngôn ngữ hỗ trợ như C, C++
 - Hệ thời gian thực: Ngôn ngữ C, C++, ADA, Assembly
 - Hệ thống nhúng: C++, Java, ...
 - 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

6.5 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 soát: 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 thlut, mỗi lệnh/1 dòng

BÀI TẬP CHƯƠNG 6

Chương 7

XÁC MINH VÀ THẨM ĐỊNH PHẦN MỀM

7.1 Giới thiệu về xác minh và thẩm định

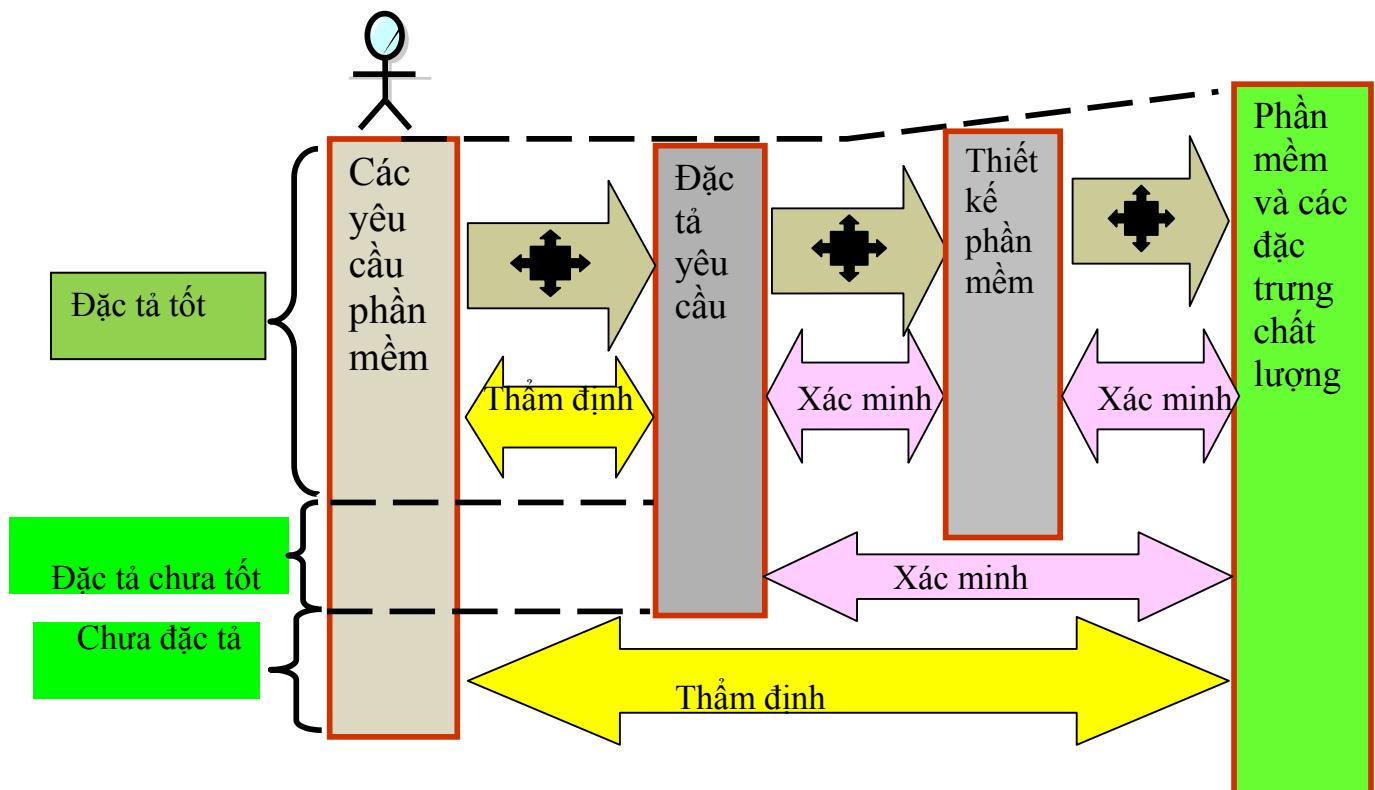
7.1.1 Khái niệm về xác minh và thẩm định

Xác minh (Verification) là việc kiểm tra xem phần mềm làm ra có đúng đặc tả (yêu cầu, thiết kế) hay không.

Thẩm định (Validation) là việc kiểm tra xem phần mềm có đáp ứng yêu cầu của người dùng hay không.

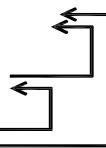
Đây là hai hoạt động cốt yếu đảm bảo chất lượng phần mềm, diễn ra trong suốt quá trình phát triển phần mềm.

Kiểm chứng phần mềm là sự kết hợp của hoạt động thẩm định và xác minh, được thực hiện ở mọi giai đoạn phát triển phần mềm, với các sản phẩm khác nhau, do các đối tượng khác nhau thực hiện



7.1.2 Các hoạt động xác minh

Cơ sở cho hoạt động xác minh

- ✓ Bản đặc tả yêu cầu
 - ✓ Các bản thiết kế
 - ✓ Mã nguồn
- 

Các hoạt động xác minh

- ✓ Rà soát (thanh tra, xét duyệt, kiểm toán)
- ✓ Kiểm thử (kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống)

7.1.3 Các hoạt động thẩm định

Cơ sở cho hoạt động thẩm định

- ✓ Bản đặc tả yêu cầu
- ✓ Mã nguồn

Các hoạt động thẩm định

- ✓ Rà soát: Thanh tra, xét duyệt
- ✓ Kiểm toán
- ✓ Kiểm thử thẩm định (chấp thuận)

Hai hoạt động chính của thẩm định và xác minh đó là rà soát và kiểm thử.

7.1.4 Các hình thức thẩm định và xác minh

a. Thẩm định và xác minh tĩnh

Thẩm định và xác minh tĩnh là hoạt động rà soát, xét duyệt các tài liệu phần mềm như bản kế hoạch, tài liệu đặc tả yêu cầu, tài liệu đặc tả thiết kế, tài liệu mã nguồn nhằm phát hiện một số loại lỗi nhất định. Phương pháp này thường khó đánh giá tính hiệu quả của sản phẩm.

b. Thẩm định và xác minh động

Thẩm định và xác minh động được thực hiện trên cơ sở cho vận hành sản phẩm phần mềm:

- ✓ Làm mẫu yêu cầu
- ✓ Vận hành chương trình (kiểm thử)

- ✓ Mô phỏng hệ thống

Người phát triển/người dùng trực tiếp kiểm tra đánh giá

Hình thức thẩm định và xác minh này có thể phát hiện mọi lỗi và khiếm khuyết của phần mềm, hiệu quả cao hơn so với hình thức thẩm định và xác minh tĩnh.

7.2 Rà soát phần mềm

7.2.1 Khái niệm rà soát phần mềm

Rà soát là hoạt động xem xét, đánh giá sản phẩm được tiến hành ở mỗi giai đoạn để phát hiện ra những khiếm khuyết cần sửa trước khi chuyển sang giai đoạn sau.

Mục tiêu của rà soát phần mềm:

- ✓ Chỉ ra các khiếm khuyết cần phải cải thiện
- ✓ Khẳng định những sản phẩm đạt yêu cầu.
- ✓ Kiểm soát việc đạt chất lượng kỹ thuật tối thiểu của sản phẩm (có diện mạo không đổi, ổn định).

Rà soát phần mềm có khả năng áp dụng tại các thời điểm khác nhau trong quá trình phát triển phần mềm.

7.2.2 Các hình thức rà soát và tiến trình rà soát

a. Các hình thức rà soát

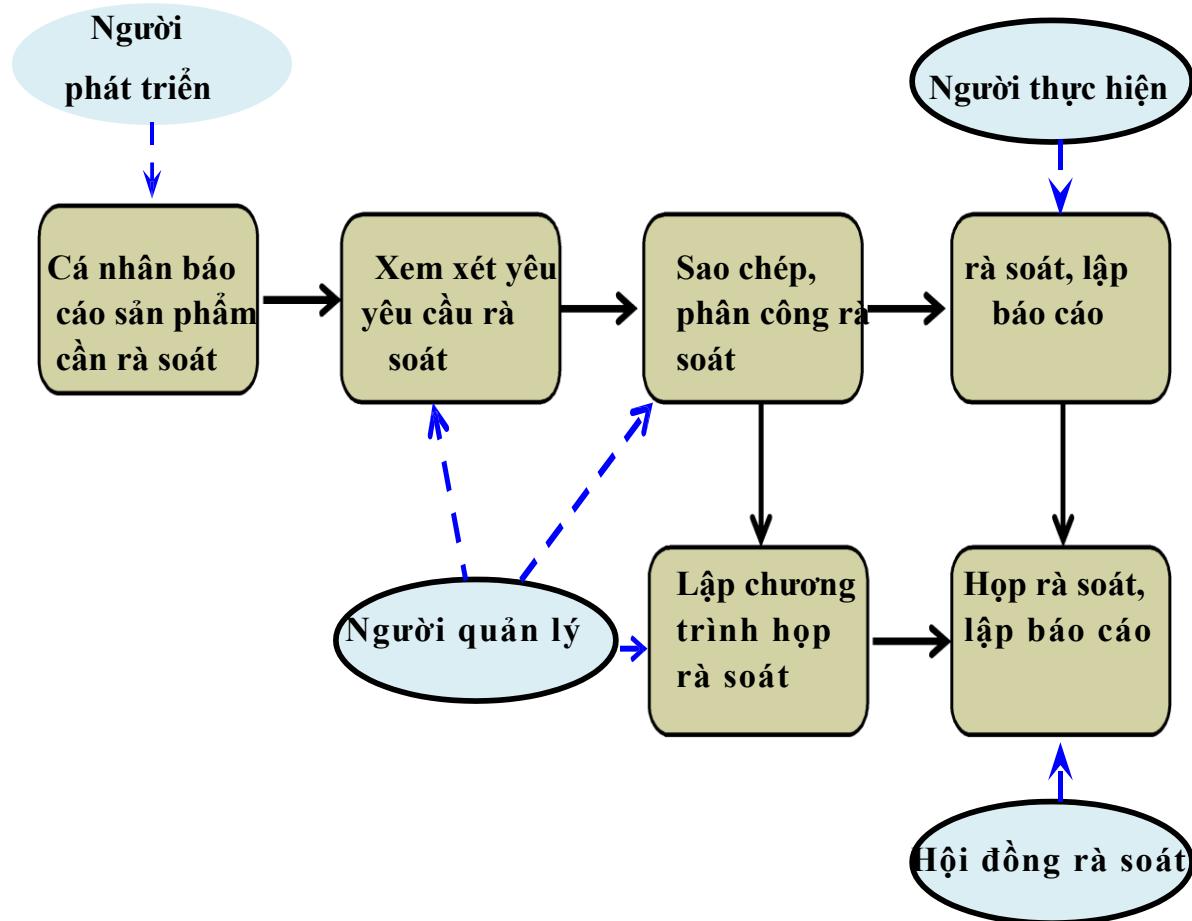
Rà soát phần mềm có thể tiến hành theo hình thức *thanh tra, họp xét duyệt không chính thức, họp xét duyệt chính thức* với các thành viên: Khách hàng, nhà quản lý, nhân viên kỹ thuật. Họp xét duyệt chính thức còn được gọi là hoạt động xét duyệt kỹ thuật chính thức – Formal Technical Review (FTR).

Rà soát kỹ thuật chính thức là hoạt động đảm bảo chất lượng phần mềm do những người đang tham gia phát triển thực hiện. Đây là một phương thức hiệu quả để cải thiện chất lượng hệ thống. Mục tiêu cụ thể của hoạt động này là:

- ✓ Phát hiện các lỗi trong chức năng, lỗi logic trong chương trình và lỗi triển khai
- ✓ Kiểm thử sự phù hợp của phần mềm với yêu cầu
- ✓ Khẳng định phần kiểm thử đạt yêu cầu
- ✓ Đảm bảo phần mềm phù hợp với các chuẩn đã định trước

- ✓ Đảm bảo phần mềm được phát triển theo một cách thức nhất quán (uniform manner)
- ✓ Làm cho dự án dễ quản lý hơn
- ✓ Ngoài ra, làm cơ sở huấn luyện các kỹ sư trẻ và có ích ngay cả cho những kỹ sư đã có kinh nghiệm.

b. Tiến trình rà soát



Cuộc họp rà soát

- ✓ Thành phần: lãnh đạo rà soát, các cá nhân rà soát và người tạo ra sản phẩm được rà soát (+ khách hàng).
- ✓ Kết luận đưa ra là một trong 3 quyết định sau:
 - Chấp nhận sản phẩm không cần chỉnh sửa
 - Khước từ sản phẩm với những lỗi nghiêm trọng

- Chấp nhận cho chỉnh sửa sản phẩm, sau khi chỉnh sửa phải rà soát lại.
- ✓ Mọi thành viên tham gia cuộc họp phải ký vào quyết định.

Sản phẩm rà soát

Sản phẩm của cuộc họp rà soát bao gồm:

- ✓ Một báo cáo các vấn đề nảy sinh do cá nhân rà soát nêu ra
- ✓ Một danh sách các vấn đề cần giải quyết. Danh sách này phục vụ cho việc:
 - Nhận ra các vùng có vấn đề trong sản phẩm được rà soát
 - Dùng như một danh sách các khoản, mục để chỉ cho người làm ra sản phẩm cần chỉnh sửa
 - Thiết lập thủ tục để đảm bảo rằng các khoản mục trong danh sách đã có sẽ được chỉnh sửa thực sự
- ✓ Một bản tổng kết cuộc họp rà soát phải chỉ rõ đã rà soát cái gì, ai rà soát, và tìm thấy cái gì và kết luận ra sao.

Thực thi rà soát

Mọi sản phẩm được tạo ra ở mỗi bước đều được rà soát không chỉ là sản phẩm cuối cùng. Tiến trình phát triển sản phẩm phần mềm chung nhất bao gồm 4-5 giai đoạn:

- ✓ Kỹ nghệ hệ thống: sản phẩm là bản kế hoạch triển khai
- ✓ Phân tích, xác định yêu cầu phần mềm: sản phẩm là bản đặc tả yêu cầu hệ thống phần mềm.
- ✓ Thiết kế phần mềm: Sản phẩm là bản thiết kế
- ✓ Mã hóa: sản phẩm là bản mã nguồn chương trình.
- ✓ Kiểm thử phần mềm: Sản phẩm là bản kế hoạch kiểm thử
- ✓ Bảo trì: Kế hoạch bảo trì.

Khi rà soát ta cần bám theo sản phẩm của các giai đoạn này.

7.3 Kiểm thử phần mềm

7.3.1 Khái niệm

Kiểm thử là một pha không thể thiếu được trong quá trình phát triển hệ thống. Kiểm thử giúp cho người xây dựng hệ thống và khách hàng đều thấy được rằng hệ thống mới đã thoả mãn yêu cầu đề ra hay chưa.

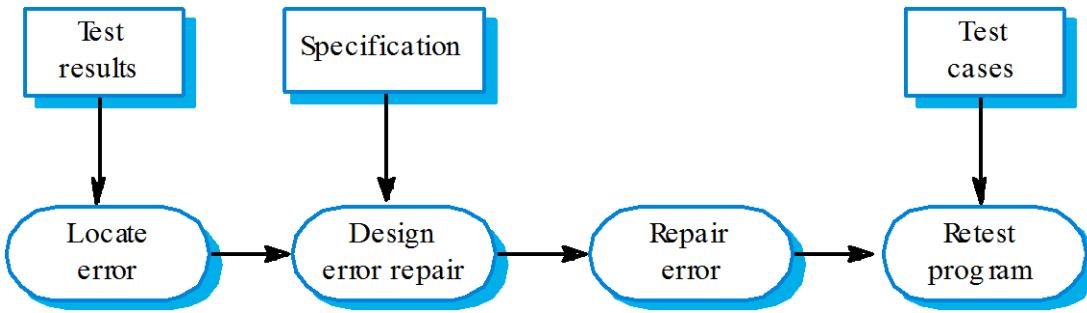
- ✓ *Theo IEEE:* Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần của hệ thống dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra các đánh giá về hệ thống hoặc thành phần đó.
- ✓ *Theo Myers[the art of software testing]:* Kiểm thử là tiến trình vận hành chương trình với mục đích tìm thấy lỗi.

=> kiểm thử là tổ chức, vận hành phần mềm một cách có kế hoạch và phương pháp để tìm ra lỗi. Bản kế hoạch cần chỉ rõ cần vận hành như thế nào để hiệu xuất tìm ra lỗi là cao nhất và chi phí (thời gian, công sức) là ít nhất. Nội dung của hoạt động kiểm thử bao gồm:

- ✓ Kế hoạch kiểm thử
- ✓ Phương pháp kiểm thử
- ✓ Chiến lược kiểm thử và
- ✓ Kỹ thuật sử dụng.

Hoạt động kiểm thử hoàn toàn khác với hoạt động gỡ rối chương trình. Nhưng chúng có quan hệ chặt chẽ với nhau để đạt tới mục tiêu hệ thống không còn lỗi và đáp ứng mọi yêu cầu đã đặt ra đối với hệ thống.

- Mục tiêu của kiểm thử là nhằm phát hiện các lỗi;
- Mục tiêu của gỡ rối:
 - Xác định bản chất của lỗi và định vị lỗi trong chương trình
 - Tiến hành sửa lỗi theo tiến trình chỉ ra trong hình ...



7.3.2 Một số khái niệm cơ bản

- ✓ **Sai sót (error)**: là một sự nhầm lẫn hay một sự thiếu sót của người phát triển trong quá trình phát triển phần mềm.
- ✓ **Lỗi (fault, defect)**: Lỗi xuất hiện trong chương trình là kết quả của một sai sót
- ✓ **Hỗng hóc (failure)**: là kết quả của một hoặc một số lỗi. Chúng làm cho chương trình không hoạt động được hoặc nếu hoạt động được thì cho ra kết quả không như mong đợi.

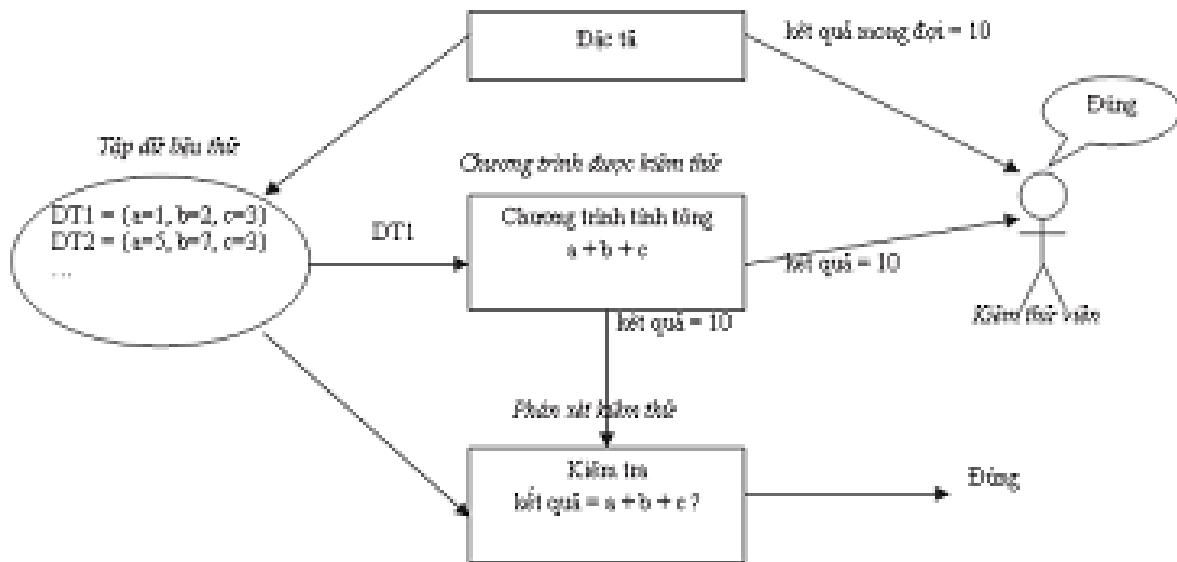
Từ đây ta dẫn đến mối quan hệ giữa các khái niệm trên được mô tả như hình...



- ✓ **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

- o Kết quả mong đợi

Ví dụ về ca kiểm thử:



7.3.3 Kế hoạch kiểm thử

Trước khi tiến hành hoạt động kiểm thử, chúng ta cần lập bản kế hoạch kiểm thử. Bản kế hoạch kiểm thử tổng thể thường có cấu trúc như sau:

1. Giới thiệu chung

- ✓ Mô tả hệ thống cần kiểm thử
- ✓ Các mục tiêu kiểm thử
- ✓ Phương pháp sử dụng
- ✓ Tài liệu hỗ trợ

2. Kế hoạch

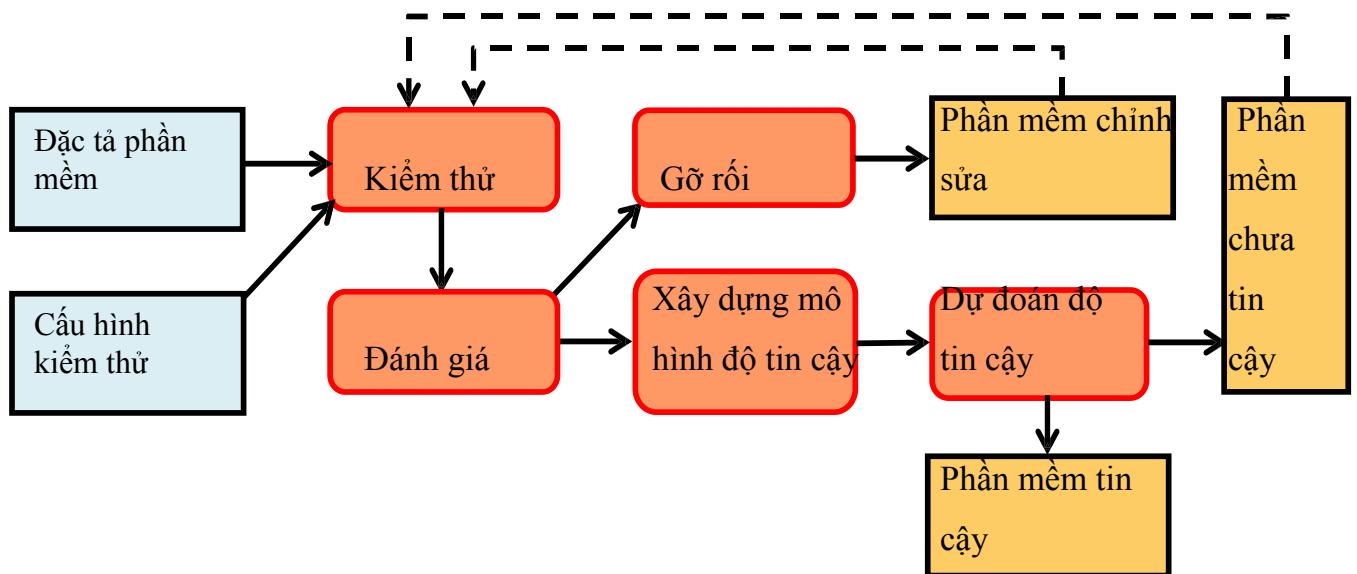
- ✓ Thời gian, địa điểm
- ✓ Tài liệu kiểm thử: Các ca kiểm thử, tiến trình, lịch trình
- ✓ Điều kiện

3. Các yêu cầu: Phần cứng, phần mềm, nhân sự

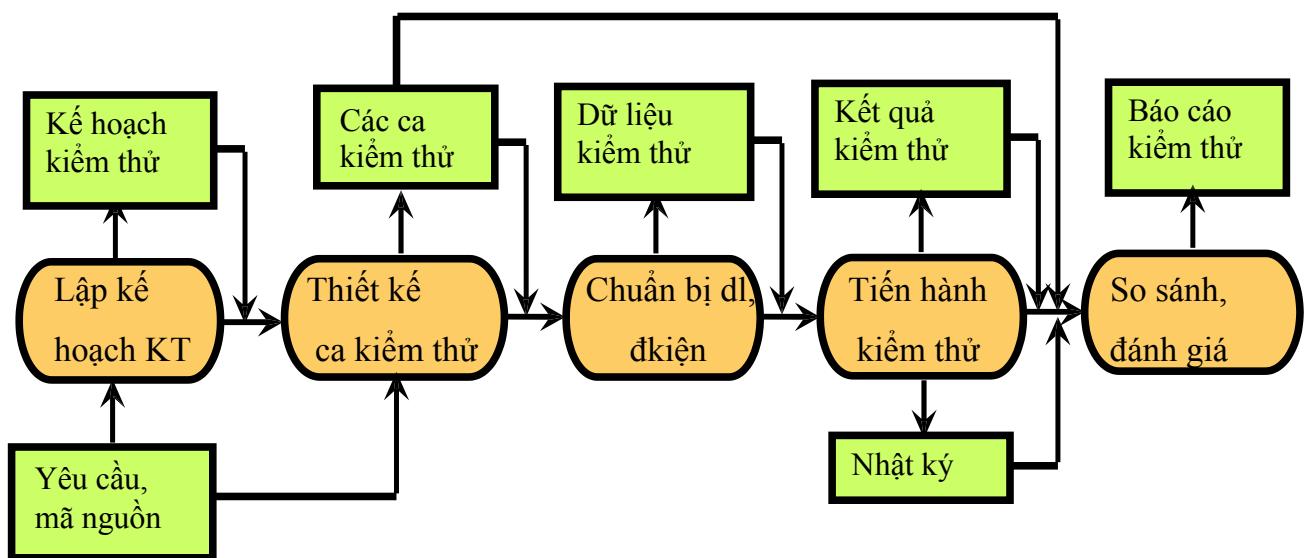
4. Kiểm soát quá trình kiểm thử

7.3.4 Tiến trình kiểm thử

Sơ đồ dòng thông tin của tiến trình kiểm thử:



Tiến trình thực hiện ca kiểm thử được mô tả như hình:



Trong đó:

1. *Thiết kế các ca kiểm thử*: Một ca kiểm thử bao gồm tập dữ liệu thử, điều kiện thực thi và kết quả mong đợi. Đầu ra của hoạt động này là một tập hợp các mẫu kiểm thử có khả năng đánh giá hiệu quả và phát hiện các khuyết điểm của hệ thống. Mục tiêu thiết kế ca kiểm thử nhằm tìm ra nhiều sai sót nhất và với công

sức và thời gian là nhỏ nhất có thể. Một phương pháp thiết kế ca kiểm thử tốt phải đảm bảo:

- ✓ Tính đầy đủ: Không sót phần nào
- ✓ Cung cấp khả năng thực sự phát hiện được sai sót
- ✓ Ca kiểm thử hiệu quả là ca kiểm thử phát hiện ra ít nhất một lỗi.

Các phương pháp thiết kế các ca kiểm thử gồm:

- ✓ *Kiểm thử dựa trên yêu cầu*: Một nguyên tắc của kỹ thuật xác định yêu cầu là những yêu cầu của hệ thống phải có khả năng kiểm thử. Kiểm thử dựa trên yêu cầu là kỹ thuật kiểm thử hợp lệ (validation), trong đó ta phải xem xét từng yêu cầu và đưa ra một tập các mẫu thử cho những yêu cầu đó.
 - ✓ *Kiểm thử phân hoạch*: Dữ liệu đầu vào và kết quả đầu ra thường rơi vào các lớp khác nhau, trong đó tất cả các thành viên của lớp đều có quan hệ với nhau. Mỗi lớp này thường là một phân hoạch hoặc một miền ứng dụng mà chương trình chạy theo một cách thích ứng với từng thành viên của lớp. Các trường hợp kiểm thử được lựa chọn từ những phân hoạch này.
 - ✓ *Kiểm thử hướng cấu trúc*: Kiểm thử hướng cấu trúc đưa ra các trường hợp kiểm thử dựa theo cấu trúc chương trình. Những hiểu biết về chương trình được sử dụng để xác định các trường hợp kiểm thử bổ sung.
 - ✓ *Kiểm thử đường đi*: Mục tiêu của kiểm thử đường đi nhằm đảm bảo rằng tập hợp các mẫu thử trên từng đường đi qua hệ thống sẽ được thực hiện ít nhất một lần. Đầu vào của kiểm thử đường đi là biểu đồ luồng chương trình, gồm các nút biểu diễn các nhánh của chương trình và các cung biểu diễn luồng điều khiển.
2. *Tạo dữ liệu thử*: Kiểm thử với tất cả các dữ liệu vào của hệ thống là cần thiết, tuy nhiên không thể kiểm thử vét cạn. Do đó, chúng ta nên chọn tập các dữ liệu thử đại diện từ miền dữ liệu vào. Chúng ta cần căn cứ vào các tiêu chuẩn để lựa chọn dữ liệu thử.
 3. *Thực thi chương trình với dữ liệu thử*: cung cấp dữ liệu thử cho chương trình, thực thi chương trình và ghi nhận kết quả

4. *Quan sát kết quả kiểm thử*: Hoạt động này được thực hiện trong hoặc sau khi thúc thi chương trình, nhằm so sánh kết quả nhận được từ chương trình với kết quả mong đợi đã chỉ ra trong ca kiểm thử.

Về mặt lý thuyết, chúng ta phải kiểm thử hệ thống một cách cẩn kẽ thì mới khẳng định được chương trình không còn khiếm khuyết. Tuy nhiên, trong thực tế không thể kiểm thử hệ thống một cách cẩn kẽ được, đặc biệt đối với những hệ thống lớn và phức tạp. Một số chính sách nên sử dụng khi kiểm thử:

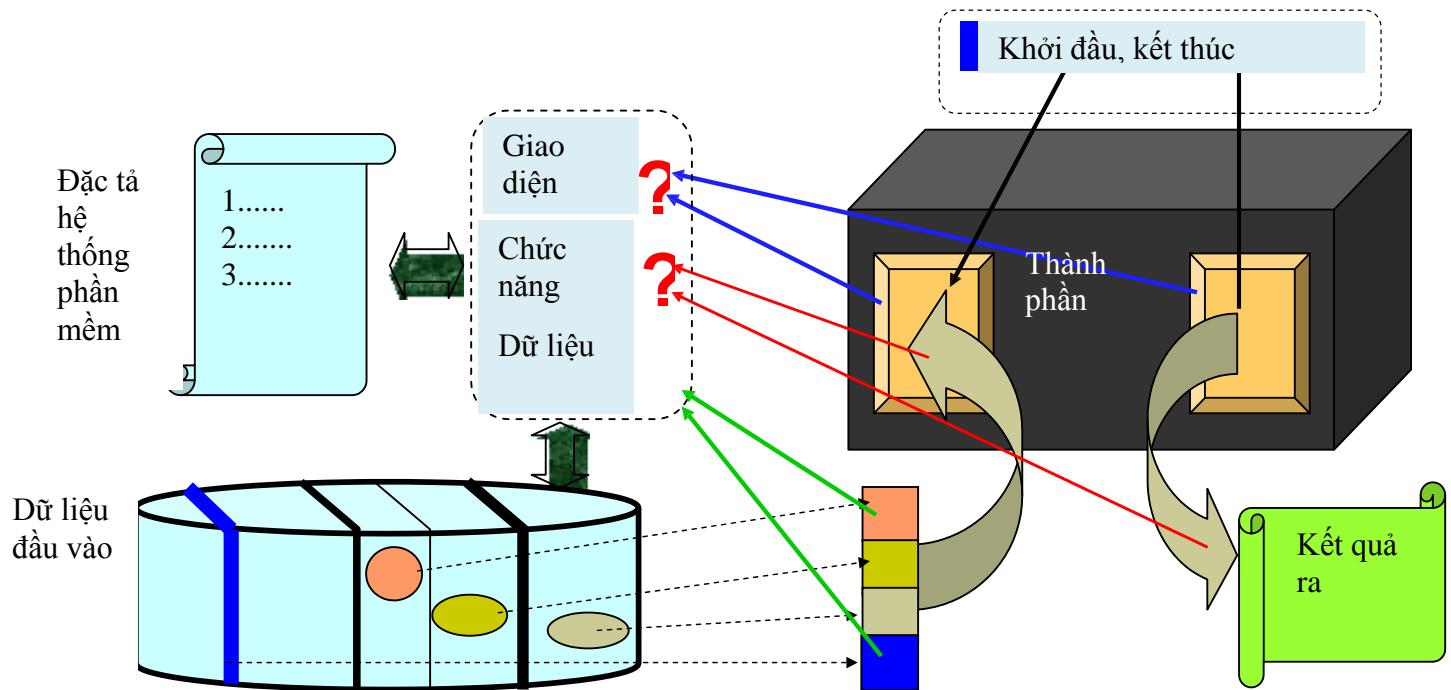
- ✓ Tất cả những chức năng được truy nhập qua menu cần phải kiểm thử
- ✓ Các chức năng kết hợp được truy nhập thông qua cùng một menu cũng phải được kiểm thử.
- ✓ Những nơi người sử dụng phải nhập thông tin đầu vào thì tất cả các chức năng sử dụng dữ liệu này đều phải được kiểm thử với những đầu vào chính xác và không chính xác.

7.3.5 Phương pháp và chiến lược kiểm thử

Hai phương pháp kiểm thử phổ biến đó là kiểm thử hộp trắng và kiểm thử hộp đen.

a. Kiểm thử hộp đen (black – box testing)

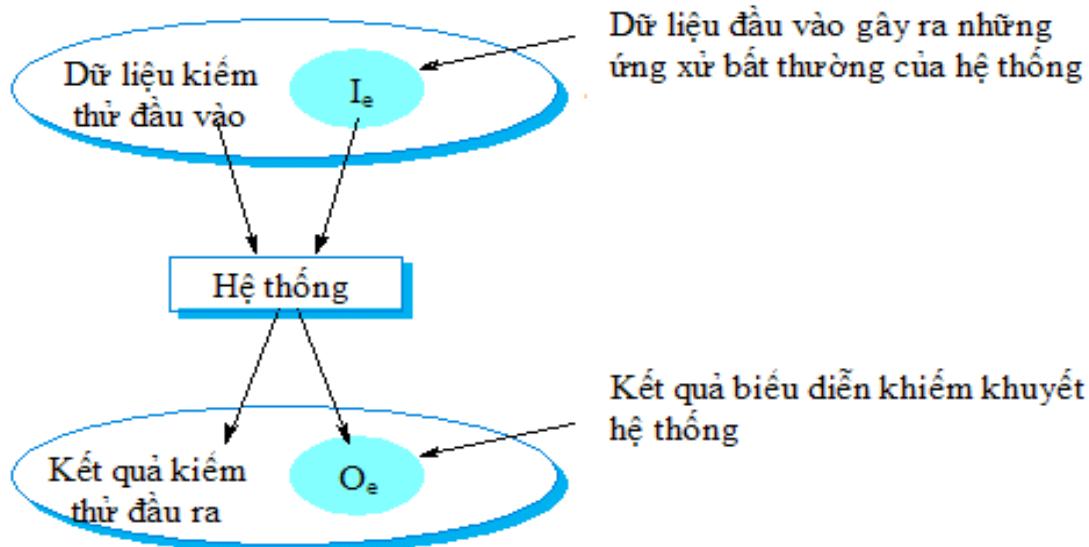
Kiểm thử hộp đen xem đối tượng kiểm thử như một hộp đen, thông qua giao diện của hộp đen để đưa dữ liệu vào và nhận thông tin ra. Phương pháp kiểm thử này được dùng để kiểm thử các yêu cầu chức năng. Đối tượng kiểm thử là module, hệ con hoặc toàn bộ hệ thống. Hình sau đây mô tả khái niệm kiểm thử hộp đen.



Mục đích của kiểm thử hộp đen nhằm bắt các lỗi sai của hộp đen và kiểm tra tính ợp lệ của hộp đen. Các lỗi cần bắt liên quan đến:

- ✓ Chức năng: Có đầy đủ, có đúng đắn không
- ✓ Giao diện: giao diện vào, giao diện ra có đầy đủ, đúng, phù hợp, tiện lợi không
- ✓ Cấu trúc, truy cập dữ liệu: có thông suốt, đúng đắn không
- ✓ Thực thi: trôi chảy, kịp thời, chịu lỗi, phục hồi được
- ✓ Khởi đầu – kết thúc: mọi tiến trình bình thường

Hình... mô tả mục tiêu của kiểm thử hộp đen.



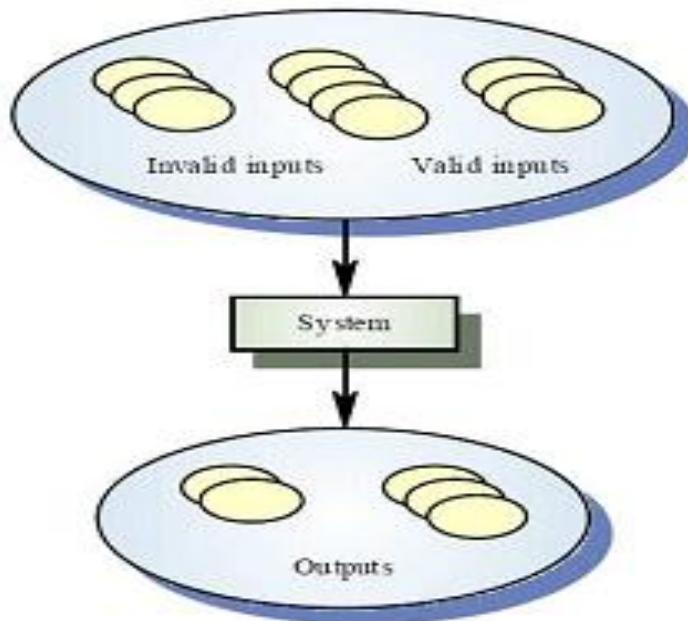
Dữ liệu thử thường được xây dựng trước khi mã hóa/lập trình, dựa chủ yếu vào tài liệu đặc tả phần mềm. Các kỹ thuật kiểm thử hộp đen gồm:

- ✓ Phân hoạch lớp tương đương (equivalence class testing),
- ✓ Phân tích giá trị biên (boundary value analysis)
- ✓ Đồ thị nhân quả (cause – effect graph)

a.Kỹ thuật Phân hoạch lớp tương đương

Phân hoạch lớp tương đương dựa trên ý tưởng:

- ✓ Chia tập dữ liệu thành các lớp dữ liệu có quan hệ với nhau. Mỗi lớp hoặc là đúng hay sai chỉ cần kiểm tra một số giá trị đặc trưng của nó -> rút ra được số ca kiểm thử
- ✓ Hình ... chỉ ra các lớp tương đương



Để phân dữ liệu thành các lớp tương đương ta tiến hành theo các bước như sau:

- ✓ Đối với mỗi dữ liệu, xác định các lớp tương đương từ miền dữ liệu. gồm các lớp dữ liệu vào hợp lệ, các lớp dữ liệu vào không hợp lệ.
- ✓ Chọn dữ liệu đại diện cho mỗi lớp tương đương.
- ✓ Kết hợp các dữ liệu thử của mỗi lớp bởi tích Đề - các để tạo ra bộ dữ liệu thử.

Nguyên tắc phân hoạch các lớp tương đương

- ✓ Nếu dữ liệu vào thuộc vào một khoảng, xây dựng:
 - Một lớp các giá trị lớn hơn giá trị lớn nhất của khoảng
 - Một lớp các giá trị nhỏ hơn giá trị nhỏ nhất của khoảng
 - Lớp gồm N giá trị hợp lệ
- ✓ Nếu dữ liệu vào là tập hợp các giá trị, xây dựng:
 - 1 lớp với tập rỗng
 - 1 lớp với quá nhiều các giá trị
 - Lớp gồm N giá trị hợp lệ.
- ✓ Nếu dữ liệu vào là điều kiện ràng buộc, xây dựng:
 - 1 lớp chứa các ràng buộc được thỏa mãn
 - 1 lớp với các ràng buộc không được thỏa mãn

Ví dụ: Xét bài toán nhập vào 3 cạnh của tam giác.

Các lớp tương đương gồm:

Thường	Nhọn 6,5,3	Vuông 5,6,10	Tù 3,4,5
Cân	6,1,6	7,4,4	$\sqrt{2},2,\sqrt{2}$
Đều	4,4,4	không thâ	không thâ
Không là tam giác			-1,2,8

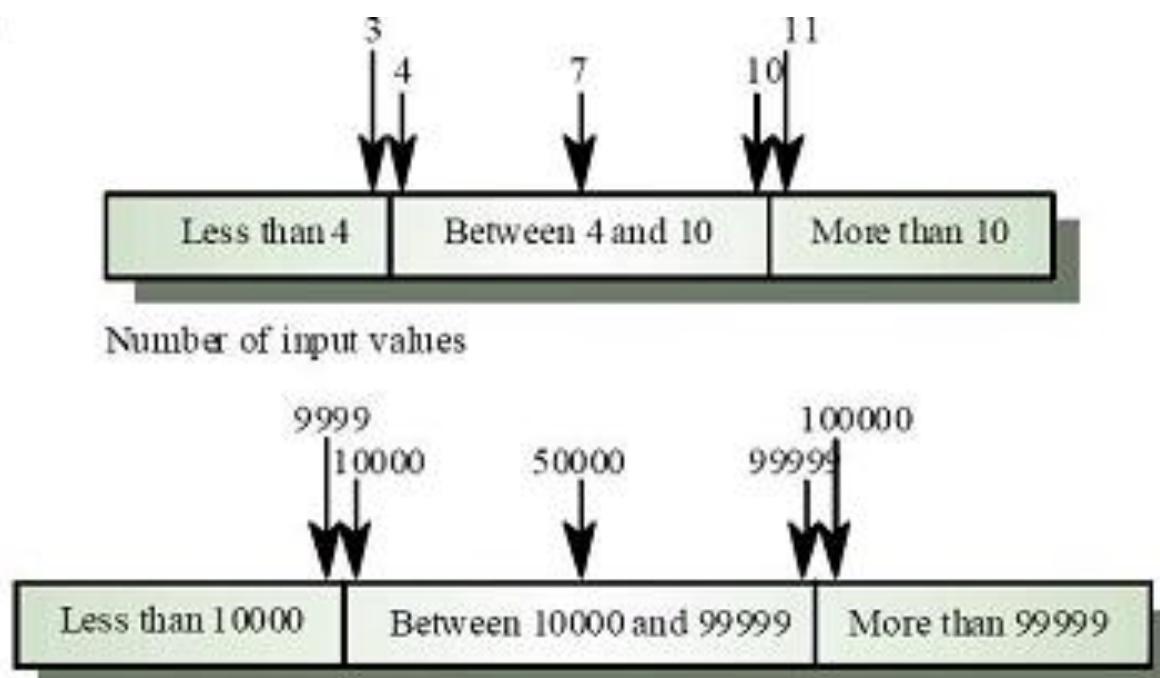
b. Kỹ thuật phân tích giá trị biên

Thực tế cho thấy lỗi thường xuất hiện gần các giá trị biên của miền dữ liệu. Do đó, phương pháp này tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử.

Nguyên tắc kiểm thử dữ liệu vào gồm:

- ✓ Giá trị nhỏ nhất
- ✓ Giá trị gần kề giá trị nhỏ nhất
- ✓ Giá trị bình thường
- ✓ Giá trị lớn nhất
- ✓ Giá trị gần kề giá trị lớn nhất

Ví dụ:



Nguyên tắc chọn dữ liệu thử:

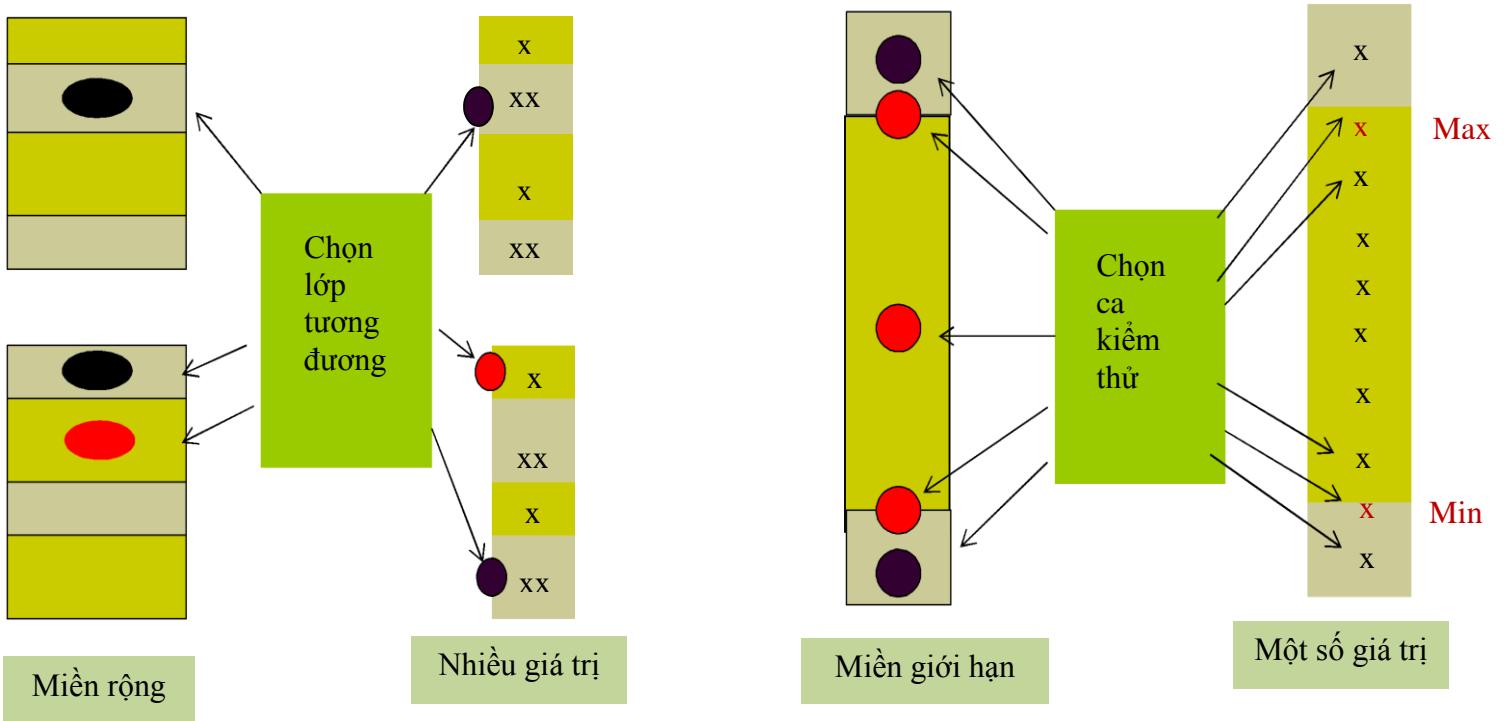
- ✓ Nếu dữ liệu vào thuộc một khoảng, chọn:
 - Hai giá trị biên
 - Bốn giá trị = 2 giá trị biên +/- sai số nhỏ nhất
- ✓ Nếu dữ liệu vào thuộc danh sách các giá trị, chọn:
 - Phần tử thứ nhất,
 - Phần tử thứ 2,
 - Phần tử kế cuối,
 - Phần tử cuối
- ✓ Nếu dữ liệu vào là điều kiện ràng buộc số giá trị, chọn:
 - Số giá trị tối thiểu
 - Số giá trị tối đa
 - Một số giá trị không hợp lệ
- ✓ Ngoài ra, chúng ta cần tự vận dụng khả năng thực tế để chọn các giá trị biên cần kiểm thử

Ví dụ: Viết chương trình nhận vào 3 số thực, kiểm tra ba số thực đó có là độ dài của 3 cạnh tam giác không. Nếu là độ dài 3 cạnh tam giác thì kiểm tra xem đó là tam giác thường, cân, hay đều cũng như kiểm tra xem đó là tam giác nhọn, vuông hay tù.

Chúng ta có thể chọn các dữ liệu thử như sau:

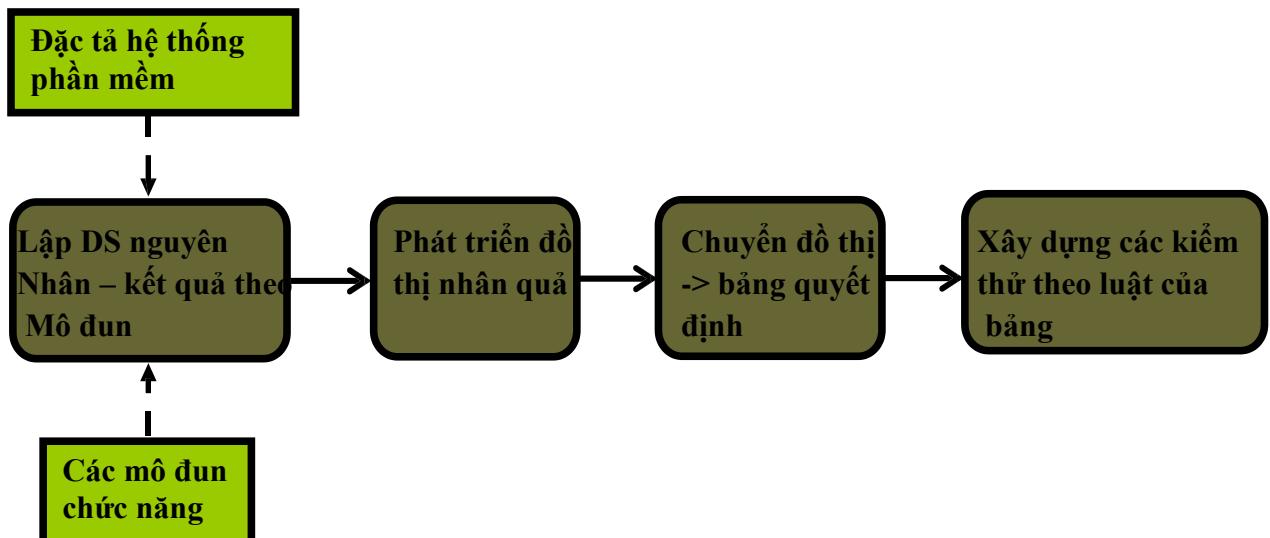
Đoạn dữ liệu thử	Kết quả
1, 1, 2	Không là tam giác
0, 0, 0	Chỉ một điểm
4, 0, 3	Một cạnh bằng không
1, 2, 3.00001	Gần là một tam giác
0.001, 0.001, 0.001	Tam giác rất nhỏ
99999, 99999, 99999	Tam giác rất lớn
3.00001, 3, 3	Tam giác gần đều
2.99999, 3, 4	Tam giác gần cân
3, 4, 5.00001	Tam giác giắc gần vuông
3, 4, 5, 6	Bốn giá trị
3	Chỉ một giá trị
-3, -3, 5	Dữ liệu vào rỗng
	Giá trị âm

Mô hình phân hoạch và phân tích giá trị biên chỉ ra trong hình



c.Kỹ thuật đồ thị nhân quả

Kỹ thuật đồ thị nhân quả là kỹ thuật để thiết kế ca kiểm thử. Nó cung cấp một biểu diễn chính xác giữa các điều kiện logic (đầu vào) và các hành động tương ứng (các đầu ra, kết quả). Đồ thị nhân quả được xây dựng dựa trên các mô đun chức năng, sự logic tiến trình và bản đặc tả hệ thống. Kỹ thuật đồ thị nhân quả gồm 4 bước:



Ví dụ: Kỹ thuật đồ thị nhân quả

Xét danh sách nhân - quả theo mô đun sau:

Modul	Nguyên nhân	Kết quả	Định danh
A	Số $> a$	đúng	A1
	Số $\geq a$	nghi ngờ	A2
	Số $= a$	nghi ngờ	A3
	Số $< a$	sai	A4
B	Số nguyên	đúng	B1

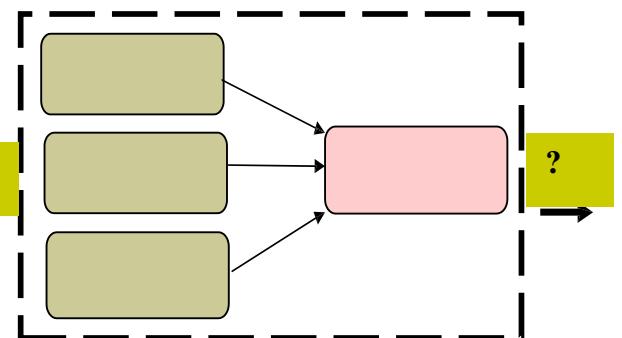
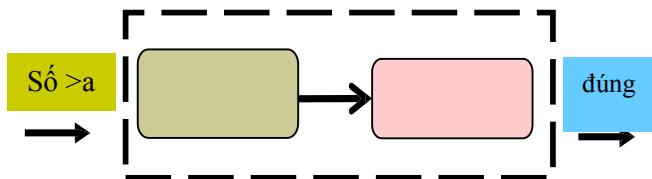
Bảng quyết định đồ thị nhân – quả

Định danh	Điều kiện	đúng	nghi ngờ	sai
A1	Số $> a$	X		
A2, A3	Số $\geq a$		X	
A4	Số $< a$			X
B1	Số nguyên	X		
...

Đồ thị nhân quả được xây dựng từ bảng quyết định

Ca 2: A2, A3, A4 & B1

Ca 1: A1&B1



7.3.7. Kiểm thử hộp trắng (white – box testing)

Kiểm thử hộp trống là phương pháp kiểm thử dựa chủ yếu vào mã nguồn/cấu trúc chương trình. Đối tượng kiểm thử là mã nguồn và thường áp dụng ở mức các mô đun đơn vị của chương trình. Kiểm thử hộp trống nhằm khám xét:

- ✓ Các chi tiết thủ tục (thuật toán)
- ✓ Các con đường logic (luồng điều khiển)
- ✓ Các trạng thái của chương trình (dữ liệu)

Dữ liệu thử được xây dựng sau khi mã hóa/lập trình. Phương pháp này thường phát hiện các lỗi liên quan đến lập trình và thường khó thực hiện hơn phương pháp kiểm thử hộp đen, do đó chi phí thường cao hơn so với hộp đen.

Các yêu cầu đặt ra đối với kiểm thử hộp trống:

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

Các kỹ thuật kiểm thử hộp trống

- ✓ Đồ thị dòng điều khiển (*)
- ✓ Điều khiển theo dòng dữ liệu
- ✓

a. *Kiểm thử dựa trên đồ thị luồng điều khiển*

Đồ thị luồng điều khiển là một đồ thị có hướng, biểu diễn một chương trình. Trong đó:

- ✓ Đỉnh của đồ thị biểu diễn 1 lệnh, một khối lệnh
- ✓ Cung của đồ thị biểu diễn một rẽ nhánh,
- ✓ Một đỉnh vào và một đỉnh ra được thêm vào để biểu diễn điểm vào, điểm ra của chương trình.

Lộ trình trên đồ thị luồng điều khiển: Là một đường đi xuất phát từ đỉnh vào, đi qua các đỉnh và cung của đồ thị và kết thúc tại đỉnh ra.

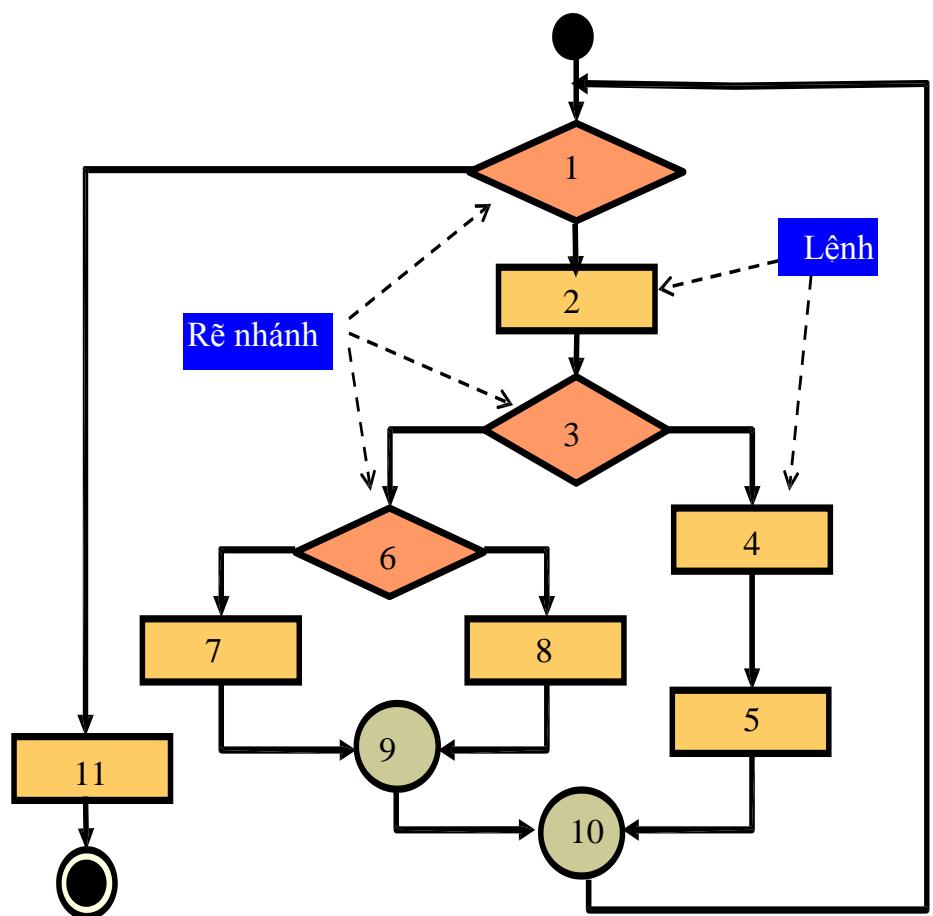
Xây dựng đồ thị luồng điều khiển

✓ Cấu trúc điều khiển một chương trình

Ví dụ 1: Xét biểu đồ điều khiển của một chương trình như hình ...

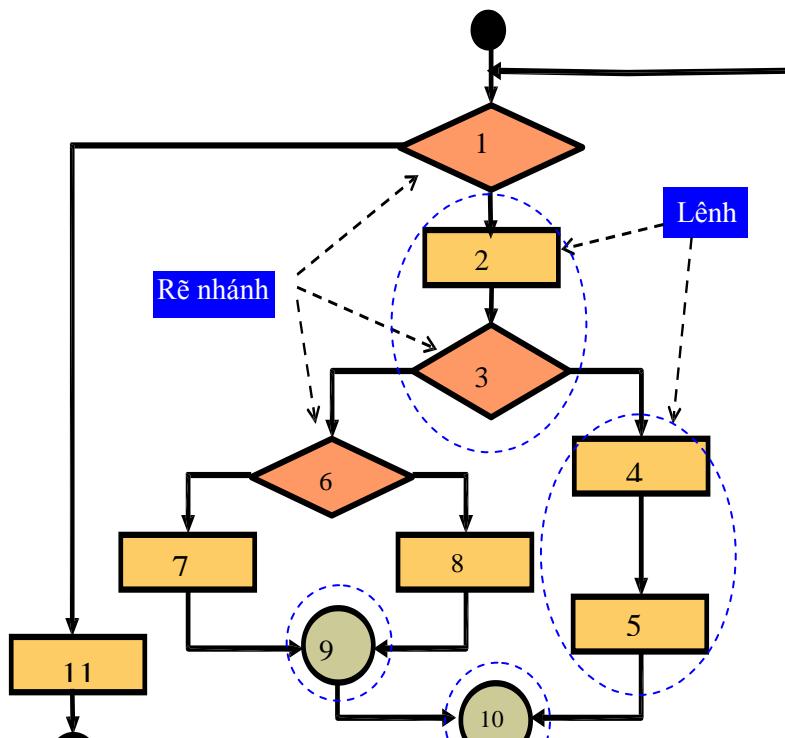
=> Có 4 đường độc lập:

- ✓ 1-11
- ✓ 1-2-3-4-5-10
- ✓ 1-2-3-6-7-9-10
- ✓ 1-2-3-6-8-9-10

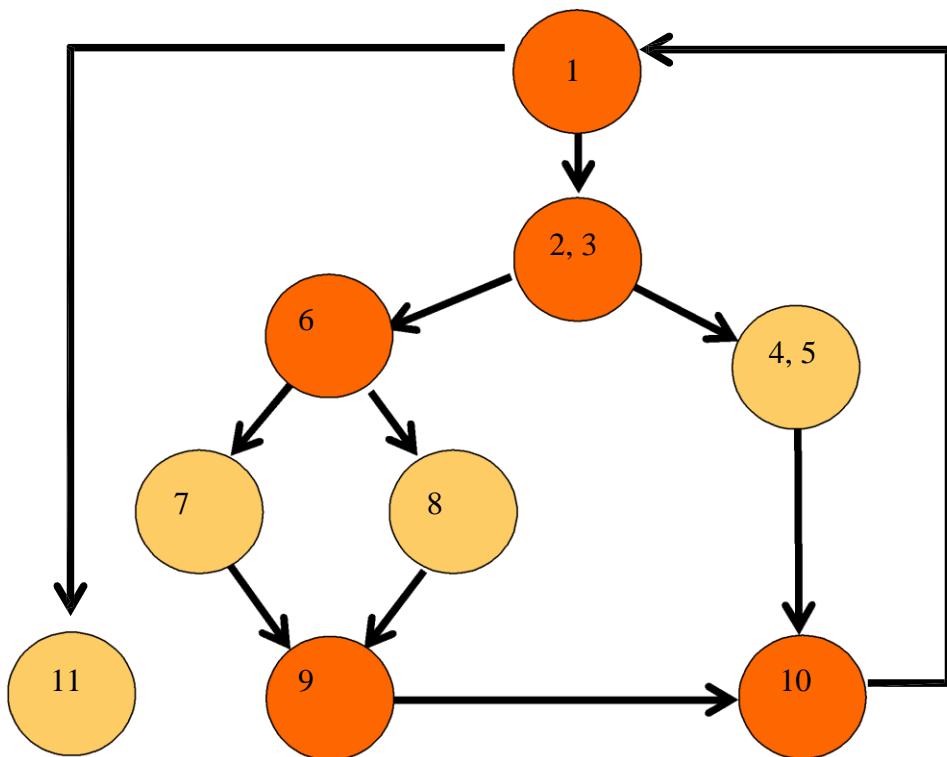


✓ Đồ thị dòng của chương trình

Cấu trúc điều khiển của chương trình



Đồ thị luồng điều khiển tương ứng:



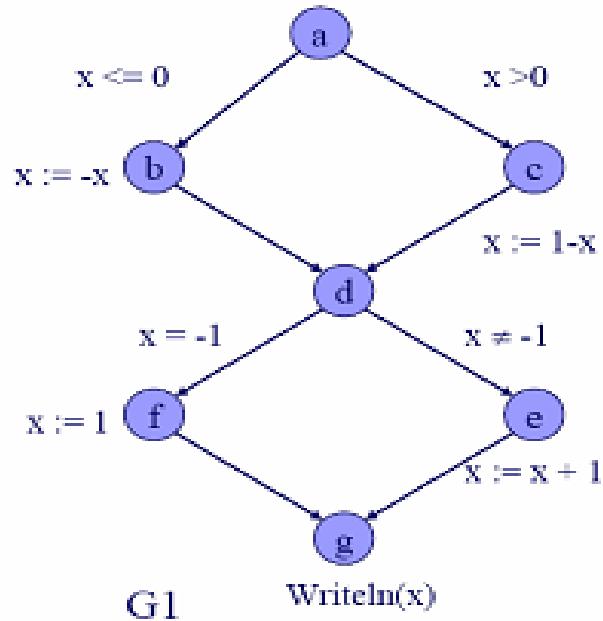
Ví dụ 2: Đồ thị luồng điều khiển của đoạn mã chương trình tương ứng

Ví dụ 1

```

if x <= 0 then
    x := -x
else
    x := 1 - x;
if x = -1 then
    x=1
else
    x := x+1;
writeln(x);

```



Đồ thị trên có 4 lô trình:

[a,b, d,f,g]

[a,b,d,e,g]

[a,c,d,f,g]

[a,c,d,e,g]

Đồ thị G1 có thể được biểu diễn dưới dạng biểu thức chính quy:

$$G1 = abdfg + acdfg + abdeg + acdeg$$

Hay đơn giản hơn

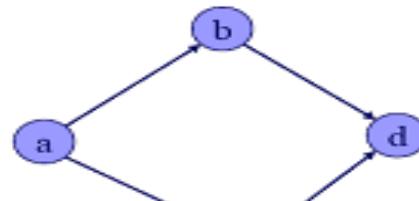
$$G1 = a(b+c)d(f+e)g$$

Các cấu trúc của đồ thị luồng điều khiển được biểu diễn như sau:

- Biểu diễn các cấu trúc



Cấu trúc tuần tự: ab



Cấu trúc rẽ nhánh: b(a + d)c



Cấu trúc lặp: ab(cb)*d

Các tiêu chuẩn bao phủ đồ thị luồng điều khiển

Để thiết kế dữ liệu thử cho chương trình, ta thiết kế dữ liệu dựa vào các tiêu chuẩn phủ sau đây:

- ✓ Phủ tất cả các đỉnh/lệnh:

- Tiêu chuẩn phủ tất cả các đỉnh thỏa mãn khi mỗi lệnh được thực thi ít nhất một lần (mỗi đỉnh của đồ thị được phủ ít nhất một lần). Đây là tiêu chuẩn phủ tối thiểu trong các tiêu chuẩn phủ.
- Các ví dụ:

Ví dụ 1: Xét hàm sau:

Function sum (x, y: integer):integer;

Begin

If(x=0) then sum:=y

Else sum:=x+y;

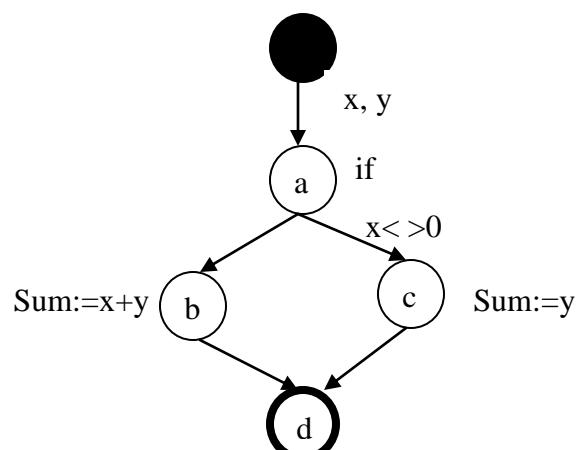
End;

=> Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ:

TD1: {x=3, y=5}

TD2: {x=0, y=5}

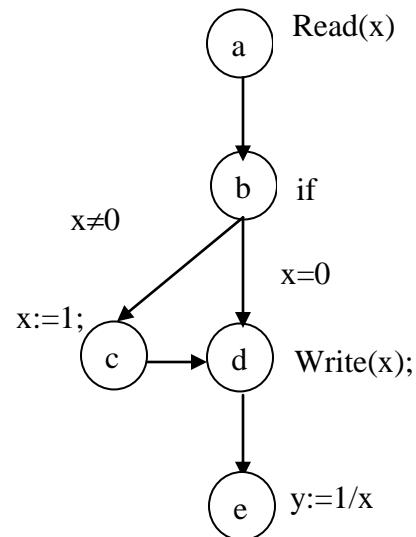
Đồ thị luồng điều khiển biểu diễn hàm là:



Ví dụ 2: Xét đoạn mã lệnh sau:

```
.....  
Read(x);  
If ( $x <> 0$ ) then  $x := 1$ ;  
Write(x);  
 $y := 1/x$ ;  
.....
```

Đồ thị luồng điều khiển biểu diễn đoạn mã là:



Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ là:

TD1: { $x=5$ }

=> Khi chạy dữ liệu thử này ta thấy không phát hiện được lỗi chia cho 0 gây ra bởi câu lệnh tại đỉnh e, vì cung bd chưa được phủ với bộ dữ liệu thử này. Đây chính là hạn chế của tiêu chuẩn phủ này. Khắc phục hạn chế này người ta đưa ra tiêu chuẩn phủ tiếp theo, đó là tiêu chuẩn phủ mọi cung.

✓ Phủ tất cả các cung

- Tiêu chuẩn phủ tất cả các cung thỏa mãn khi tiêu chuẩn phủ tất cả các đỉnh thỏa mãn, đồng thời mỗi cung phải được phủ ít nhất một lần.
- Xét các ví dụ:

Ví dụ 3: Xét đồ thị luồng điều khiển ở ví dụ 2 kể trên

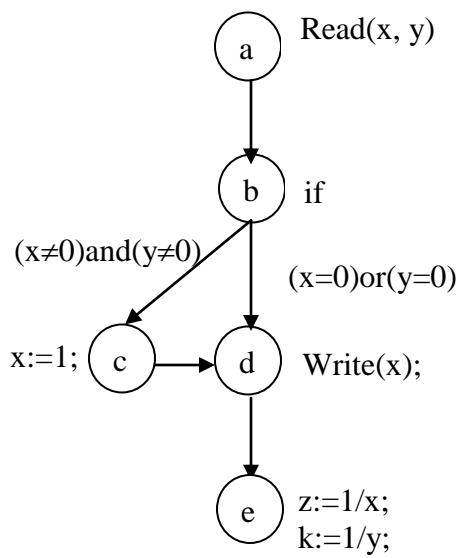
Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ mọi cung:

TD1: { $x=6$ }

TD2: { $x=0$ }

Khi chạy bộ dữ liệu thử này, ta phát hiện được lỗi chia cho 0 gây ra bởi câu lệnh e.

Ví dụ 4: Xét đồ thị luồng điều khiển sau



Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ mọi cung:

TD1: { $x=5, y=6$ }

TD2: { $x=0; y=6$ }

Thỏa mãn tiêu chuẩn phủ mọi cung nhưng chỉ phát hiện được lỗi chia cho 0 gây ra bởi câu lệnh $z := 1/x$ tại đỉnh e mà không phát hiện được lỗi chia cho 0 gây ra bởi câu lệnh $k := 1/y$ khi chạy bộ dữ liệu thử này.

=> đây chính là hạn chế của tiêu chuẩn phủ này.

✓ Phủ tất cả các quyết định

- Tiêu chuẩn phủ tất cả các quyết định thỏa mãn khi tiêu chuẩn phủ tất cả các cung thỏa mãn, đồng thời mỗi biểu thức con của biểu thức điều kiện phải được phủ với mọi giá trị đúng, sai (mỗi giá trị đúng, sai của biểu thức logic là một quyết định)
- Ví dụ: xét biểu thức logic: $a AND b$

Các quyết định		
a	B	$a AND b$
true	True	True
true	False	False
False	True	False
False	False	False

- Các ví dụ về thiết kế dữ liệu thử thỏa mãn tiêu chuẩn phủ

Ví dụ 5: Xét đồ thị luồng điều khiển tại ví dụ 4 kể trên.

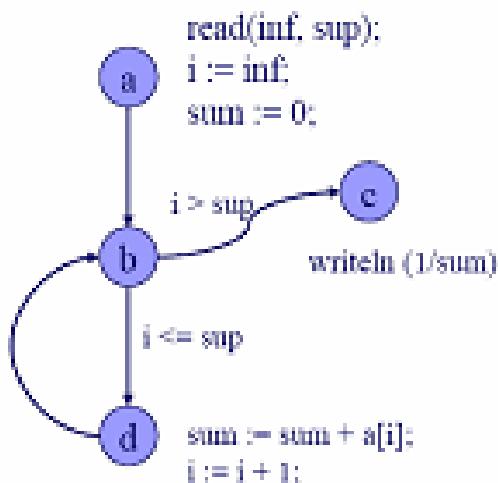
=> Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ mọi quyết định là:

TD1: { $x=0, y=0$ }

TD2: {x=y=5}

Khi chạy bộ dữ liệu thử này, sẽ phát hiện ra mọi lỗi chia cho 0

Ví dụ 6: Xét đồ thị luồng điều khiển sau:



Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ mọi quyết định:

TD: {inf= 1, sup= 5 }

Nhưng khi chạy chương trình với dữ liệu thử này ta không phát hiện được lỗi chia 0 gây ra bởi lệnh `writeln(1/sum)` đặt tại đỉnh c. Đây chính là hạn chế của tiêu chuẩn phủ này.

✓ **Phủ tất cả các lô trình**

- Tiêu chuẩn phủ tất cả các lô trình thỏa mãn khi tiêu chuẩn phủ mọi quyết định thỏa mãn, đồng thời mỗi lô trình phải được phủ ít nhất một lần.

=> Khi thiết kế dữ liệu thử thỏa mãn tiêu chuẩn phủ này ta thường gặp khó khăn khi số vòng lặp vô hạn. Khắc phục điều này chúng ta có thể:

- Chỉ thực hiện một số lần lặp nhất định, hoặc
- Chỉ phủ hai loại lô trình:
 - Lô trình vượt qua vòng lặp mà không thực hiện vòng lặp lần nào
 - Các lô trình đi qua vòng lặp một số lần
- Ví dụ 7: Xét đồ thị luồng điều khiển ở ví dụ 6

Bộ dữ liệu thử thỏa mãn tiêu chuẩn phủ là:

TD1: {inf=6; sup=1}

TD2: {inf =1, sup =3}

=> Khi chạy bộ dữ liệu này, ta sẽ phát hiện ra lỗi chia cho 0 gây ra bởi lệnh đặt tại đinh c.

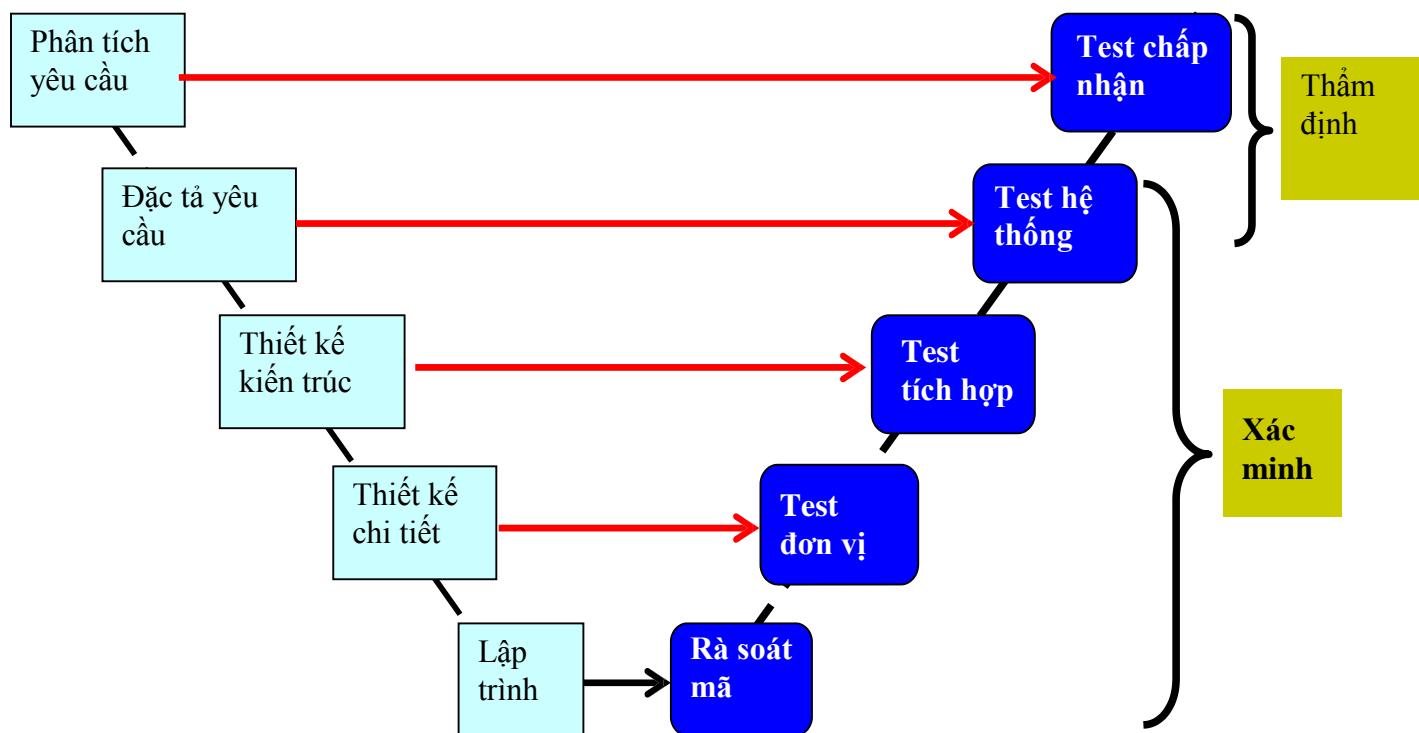
b. Kiểm thử dựa trên đồ thị luồng dữ liệu

.....

7.4 Kiểm thử trong tiến trình phát triển phần mềm

7.4.1 Kiểm thử trong mô hình chữ V

Mối quan hệ giữa các hoạt động kiểm thử và các hoạt động phát triển phần mềm được biểu diễn bởi mô hình chữ V như sau:



7.4.2 Kiểm thử thành phần (unit Testing)

Kiểm thử thành phần (hay còn gọi là kiểm thử đơn vị) là quy trình kiểm thử các thành phần riêng lẻ trong hệ thống (các module đơn vị chương trình). Đây là một

quy trình phát hiện ra các khiếm khuyết của thành phần đó và thẩm định thành phần này. Thành phần được kiểm thử có thể là:

-
- ✓ Chức năng hoặc phương thức của một đối tượng.
 - ✓ Lớp đối tượng với những thuộc tính và phương thức.
 - ✓ Thành phần kết hợp với các giao diện được định nghĩa trước để truy nhập tới các chức năng của nó.
 - ✓
-

Nội dung kiểm thử:

-
- ✓ Giao diện: Dữ liệu qua giao diện, dữ liệu vào, ra
 - ✓ Cấu trúc dữ liệu sử dụng cục bộ
 - ✓ Đường điều khiển
 - ✓ Điều kiện logic
 - ✓ Phép toán xử lý.
-

Kiểm thử đơn vị thường sử dụng kỹ thuật kiểm thử hộp trắng. Hoạt động kiểm thử này thường được thực hiện trên nền phần cứng nơi mà ta phát triển phần mềm.

7.4.3 Kiểm thử tích hợp (Integration Testing)

Kiểm thử tích hợp thường được thực hiện sau khi hoạt động kiểm thử thành phần thành công. Kiểm thử tích hợp nhằm xây dựng hệ thống từ những thành phần của nó và kiểm tra xem có vấn đề gì xảy ra từ các tương tác giữa các thành phần đó hay không. Kiểm thử tích hợp thường sử dụng kỹ thuật kiểm thử hộp đen, trong một số trường hợp sử dụng thêm kỹ thuật kiểm thử hộp trắng. Tuy nhiên khi sử dụng kỹ thuật kiểm thử hộp trắng thông thường chi phí sẽ rất cao và chúng ta thường gặp khó khăn khi thiết kế dữ liệu thử. Do đó chúng ta cần cân nhắc khi lựa chọn kỹ thuật này. Dữ liệu thử thường được thiết kế dựa trên tài liệu thiết kế hệ thống tổng thể.

Có hai cách tích hợp hệ thống:

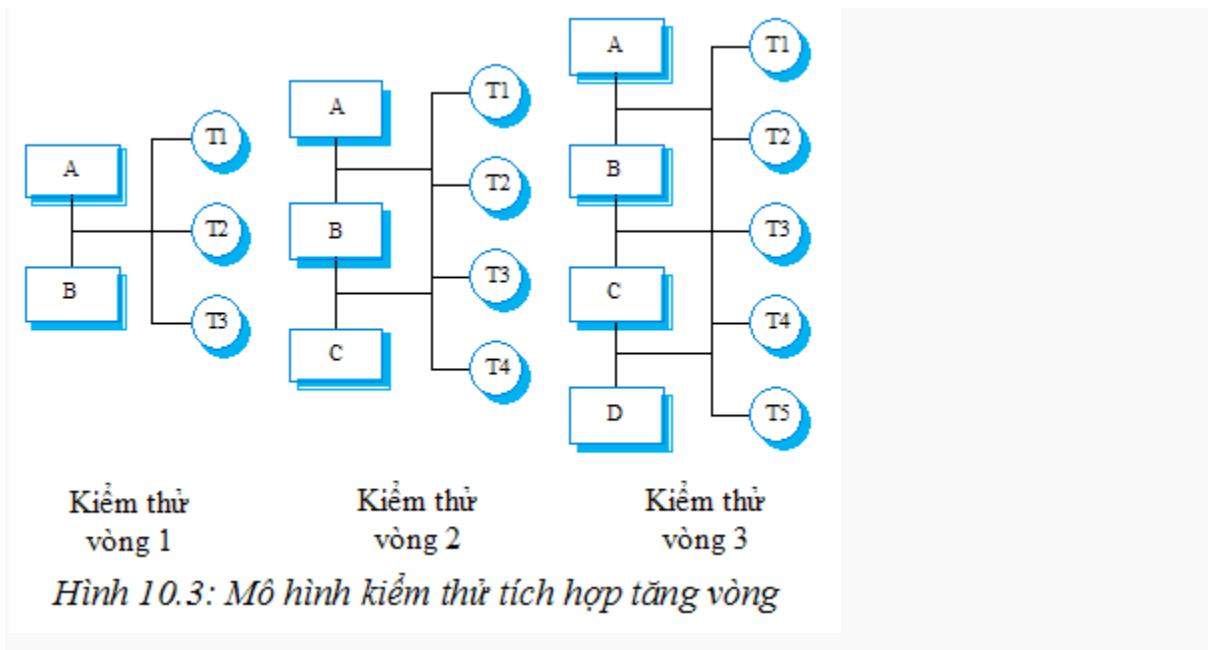
- ✓ **Tích hợp tăng dần:** từ trên xuống, từ dưới lên
 - **Tích hợp từ trên xuống:** Tích hợp những thành phần chính (main) trước, sau đó tích hợp thêm vào những thành phần được gọi trực tiếp bởi

những thành phần chính này. Tiếp tục tích hợp những thành phần được gọi trực tiếp bởi những thành phần vừa được tích hợp. Cứ tiếp tục như vậy cho đến khi ta thu được cây hệ thống hoàn chỉnh.

- *Tích hợp từ dưới lên:* Tích hợp những thành phần mà không gọi đến bất kỳ thành phần nào khác. Sau đó tích hợp thêm những thành phần gọi đến những thành phần này. Cứ như vậy cho đến khi ta thu được hệ thống phần mềm hoàn chỉnh

✓ ***Tích hợp đồng thời một lúc: “big - bang”***

Để đơn giản hóa việc xác định các lỗi, hệ thống nên được tích hợp tăng dần.



Các sai sót có thể gặp khi tích hợp

- ✓ *Dữ liệu bị mất* khi đi qua một giao diện
- ✓ *Hiệu ứng bắt lợi* do một mô đun vô tình gây ra cho các mô đun khác
- ✓ *Sự kết hợp các chức năng phụ* có thể không sinh ra chức năng chính mong muốn
- ✓ *Sự phóng đại* các sai sót riêng rẽ có thể bị đến mức không thể chấp nhận được
- ✓ *Vấn đề của cấu trúc dữ liệu toàn cục* có thể để lộ ra.

Các phương pháp kiểm thử tích hợp:

- ✓ *Dánh giá kiến trúc*: Sử dụng kiểm thử tích hợp từ trên xuống thích hợp nhằm phát hiện ra các lỗi trong kiến trúc hệ thống.
- ✓ *Minh họa hệ thống*: Sử dụng kiểm thử tích hợp từ trên xuống nhằm biểu diễn hệ thống từ những pha ban đầu của quá trình xây dựng hệ thống đến trạng thái hiện thời.
- ✓ *Kiểm thử cài đặt*: dễ dàng hơn với kiểm thử tích hợp từ dưới lên.
- ✓ *Kiểm thử quan sát*: Nhằm phát hiện các vấn đề của tất cả các phương pháp. Có thể bổ sung thêm các mã lệnh để quan sát các mẫu thử.

7.4.4 Kiểm thử hệ thống

Hệ thống muôn nói ở đây là hệ thống dựa trên máy tính (phần cứng và phần mềm) do nhiều bên xây dựng, người phát triển phần mềm chỉ là một. Hệ thống này cần được kiểm tra tổng thể.

Những sai phạm cần kiểm tra gồm:

-
- ✓ Các dữ liệu qua giao diện của các thành phần được kiểm tra
 - ✓ Đường xử lý liên kết các thành phần
 - ✓ Sự tích hợp lỗi từ các thành phần khác nhau
 - ✓ Những hạn chế khác đến năng lực do ảnh hưởng từ các thành phần như chịu lỗi, an toàn, thực thi.

Các loại kiểm thử hệ thống

1. Kiểm thử chức năng (mức hệ thống)

Kiểm thử bao gồm các chức năng giao diện, các chức năng mức người dùng hay đầu ra cuối cùng ra khỏi hệ thống

2. Kiểm thử phục hồi (chịu lỗi)

Kiểm thử phục hồi là bắt phần mềm thất bại để xem khả năng phục hồi của nó đến đâu. Có hai mức phục hồi đó là phục hồi tự động hay phục hồi cần đến sự can thiệp của con người. Độ tin cậy là một độ do đánh giá khả năng phục hồi

3. Kiểm thử an ninh (sức chịu tấn công)

Kiểm tra mọi cơ chế bảo vệ được xây dựng xem có đạt hiệu quả để ra trước các đột nhập hay không. Người kiểm thử đóng vai trò của kẻ đột nhập thực hiện mọi đột nhập có thể để đánh giá

4. Kiểm thử thi hành (thông suốt, kịp thời)

Kiểm thử thi hành được thiết kế để kiểm tra sự vận hành của hệ thống. Việc thi hành đúng phải bao gồm cả số lượng và chất lượng (hoạt động thông suốt và hiệu năng)

5. Kiểm thử chịu tải (quy mô, giá trị nhạy cảm)

Kiểm thử chịu tải là vận hành hệ thống khi sử dụng nguồn lực với số lượng, tần xuất và cường độ dị thường. Ví dụ: Vận hành một Cơ sở dữ liệu với số bản ghi cực lớn, vận hành hệ điều hành mạng với số máy nhiều dần,

7.4.5 Kiểm thử hợp thức hóa (validation testing)

Kiểm thử hợp thức hóa còn được gọi là kiểm thử chấp thuận nhằm kiểm tra xem phần mềm có đáp ứng yêu cầu của khách hàng/người dùng không. Hoạt động kiểm thử này được thực hiện thông qua một loạt các kiểm thử hộp đen. Kế hoạch kiểm thử và các thủ tục kiểm thử được thiết kế cần đảm bảo rằng:

- ✓ Tất cả các yêu cầu được thỏa mãn
- ✓ Các yêu cầu thi hành đã chính xác
- ✓ Tài liệu đúng đắn và
- ✓ Các yêu cầu khác là thỏa đáng.

Có hai loại kiểm thử hợp thức hóa thông dụng là kiểm thử Alpha và kiểm thử Beta.

Kiểm thử Alpha: Kiểm thử Alpha do nhà phát triển tiến hành. Phần mềm được người dùng tiến hành trong bối cảnh “tự nhiên”, trong một môi trường được điều khiển. Người phát triển “nhờm qua vai” người sử dụng để báo cáo các sai sót và các vấn đề về sử dụng (vì thế nó còn được gọi là kiểm thử sau lưng). Dữ liệu thử thường là dữ liệu mô phỏng.

Kiểm thử Beta: Kiểm thử Beta do khách hàng tiến hành, được thực hiện trên môi trường thực tế (môi trường vận hành phần mềm sau phát hành). Khách hàng báo cáo

tất cả các vấn đề họ gặp phải trong quá trình kiểm thử cho người phát triển một cách có định kỳ.

7.4.6 Kiểm thử hồi quy (*Regression Testing*)

Kiểm thử hồi quy là hoạt động kiểm thử lại chương trình sau khi chương trình đã được đưa vào thực tế sử dụng. Hoạt động kiểm thử này là không thể tránh khỏi vì phần mềm sau khi đưa vào sử dụng, có thể có chỉnh sửa do phát sinh các lỗi mới. Sau khi sửa lỗi, chúng ta phải tiến hành kiểm thử lại (kiểm thử hồi quy). Dữ liệu thử ta không cần thiết kế mới mà thường tái sử dụng các bộ dữ liệu thử đã sử dụng trong các giai đoạn trước đó.

BÀI TẬP CHƯƠNG 7

VẬN HÀNH, BẢO TRÌ, CẢI TIẾN CHƯƠNG TRÌNH?

CHƯƠNG 8

BẢO TRÌ, CẢI TIẾN PHẦN MỀM

8.1 Bảo trì phần mềm

8.1.1 FQA về bảo trì phần mềm

Thay đổi phần mềm là một điều không thể tránh khỏi vì những lí do như những yêu cầu mới sẽ xuất hiện khi sử dụng phần mềm, môi trường nghiệp vụ thay đổi, các lỗi phần mềm cần phải sửa chữa khi được phát hiện bởi người dùng, máy tính và các thiết bị mới được bổ sung vào hệ thống, hiệu năng hoặc độ tin cậy của hệ thống phải được cải thiện.....

Vấn đề quan trọng khi tiến hành bảo trì đó là chúng ta phải thực hiện và quản lý được mọi thay đổi đối với hệ thống phần mềm đã tồn tại, và phải thấy được tầm quan trọng của hoạt động bảo trì, cải tiến phần mềm. Các tổ chức thường đầu tư một lượng vốn khá lớn vào các hệ thống phần mềm của họ. Cho nên họ có quyền đòi hỏi phải sở hữu một hệ thống hoàn hảo. Thực tế cho thấy, ngân sách phần mềm chính trong các công ty lớn thường dùng cho việc cải tiến các hệ thống đã tồn tại hơn là phát triển một hệ thống mới. Để có cái nhìn tổng quan về hoạt động bảo trì ta lần lượt trả lời các câu hỏi sau.

Bảo trì là gì?

Bảo trì là công việc tu sửa, thay đổi phần mềm đã được phát triển (chương trình, dữ liệu, các loại tư liệu đặc tả, . .) theo những lý do nào đó. 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 hoặc
- ✓ Bổ sung những thành phần mới cho hệ thống.

Tại sao phải bảo trì?

Bảo trì là không thể tránh khỏi vì những nguyên nhân sau:

- ✓ Các yêu cầu hệ thống thay đổi khi hệ thống đang được xây dựng, do đó hệ thống được chuyển giao có thể không thoả mãn các yêu cầu này. Các sai sót

mắc phải trong quá trình phát triển và chưa được khắc phục trước khi chuyển giao sản phẩm phần mềm đến người dùng.

- ✓ Hệ thống gắn kết chặt chẽ với môi trường vận hành của nó, do đó, khi thay đổi môi trường vận hành các yêu cầu của hệ thống cũng cần thay đổi.
- ✓ Môi trường ứng dụng thay đổi, dẫn đến các yêu cầu nghiệp vụ của hệ thống cũng thay đổi.

Do đó, bảo trì là cần thiết để đảm bảo rằng hệ thống tiếp tục thỏa mãn các yêu cầu mới nảy sinh. Bảo trì có thể được thực hiện thông qua các hoạt động:

- ✓ Sửa các sai sót của phần mềm.
- ✓ Sửa các yêu cầu và các chỗ hổng thiết kế.
- ✓ Cải tiến thiết kế.
- ✓ Tạo các nâng cấp.
- ✓ Giao tiếp với các hệ thống khác.
- ✓ Chuyển đổi chương trình sang nền tảng phần cứng, phần mềm, ... phù hợp.

Phân loại bảo trì?

Hoạt động bảo trì thường được phân làm 3 loại chính sau:

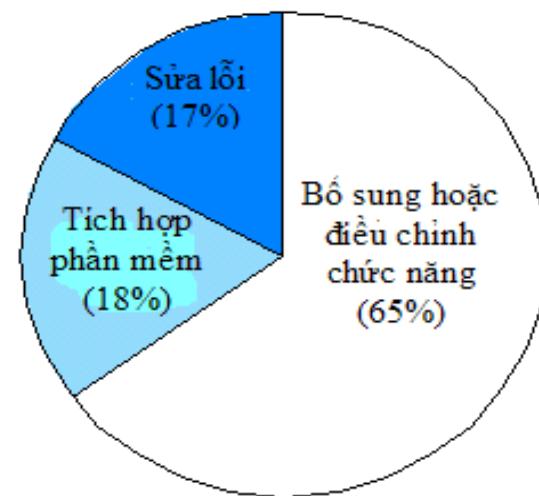
- ✓ Bảo trì để sửa chữa: Là hoạt động sửa đổi phần mềm để *khắc phục những khiếm khuyết* của phần mềm. Một số nguyên nhân điển hình dẫn đến phải thực hiện hoạt động này là:
 - ❖ Kỹ sư phần mềm và khách hiểu nhầm nhau
 - ❖ Lỗi tiềm ẩn của phần mềm do sơ ý của lập trình hoặc khi kiểm thử chưa bao quát hết
 - ❖ Vấn đề tính năng của phần mềm: không đáp ứng được yêu cầu về bộ nhớ, tệp, ... thiết kế sai, biên soạn sai ...
 - ❖ Thiếu chuẩn hóa trong phát triển phần mềm
- ✓ Bảo trì để thích nghi: Là hoạt động *tu chỉnh phần mềm theo những yêu cầu thay đổi nảy sinh từ môi trường bên ngoài hệ thống* nhằm duy trì, thích nghi

và quản lý phần mềm theo vòng đời của nó. Một số nguyên nhân chính dẫn đến phải tiến hành hoạt động bảo trì này là:

- ❖ Thay đổi về phần cứng (ngoại vi, máy chủ,..)
 - ❖ Thay đổi về phần mềm (môi trường): đổi OS
 - ❖ Thay đổi cấu trúc tệp hoặc mở rộng CSDL
 - ❖
- ✓ Bảo trì để cải tiến: Là việc tu chỉnh phần mềm theo các yêu cầu ngày càng hoàn thiện hơn, đầy đủ hơn, hợp lý hơn. Một số nguyên nhân chính dẫn đến hoạt động bảo trì này:
- ❖ Mở rộng thêm chức năng mới cho hệ thống
 - ❖ Cải tiến quản lý kéo theo cải tiến tài liệu vận hành và trình tự công việc
 - ❖ Thay đổi người dùng hoặc thay đổi thao tác
 - ❖

Chi phí bảo trì?

Chi phí bảo trì thường lớn hơn chi phí xây dựng gấp từ 2 đến 100 lần phụ thuộc vào từng ứng dụng. Chi phí bảo trì bị ảnh hưởng bởi cả các yếu tố kỹ thuật và phi kỹ thuật. Nếu bảo trì càng nhiều, sẽ càng làm thay đổi cấu trúc phần mềm và do đó sẽ làm cho việc bảo trì càng trở lên khó khăn hơn. Phần mềm có tuổi thọ càng cao thì cần chi phí bảo trì càng cao. Thực tế cho thấy, việc phân bổ chi phí tương ứng với các loại bảo trì được thống kê như hình vẽ:



Các yếu 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.

Làm thế nào để giảm chi phí bảo trì?

- ✓ Sử dụng các kỹ thuật kiểm thử tốt hơn trong suốt quá trình phát triển
- ✓ Tạo ra các tài liệu tốt hơn, tuân theo các chuẩn và các quy ước
- ✓ Dự đoán các thay đổi trong tương lai trong suốt giai đoạn xác định yêu cầu.
Thiết kế hướng đến sự mở rộng: *chi phí bảo trì dòng mã lớn hơn gấp 20 đến 40 so với giai đoạn xd dòng mã đó*
- ✓ Thiết kế cấu trúc hệ thống để không ràng buộc với các thay đổi trong tương lai
- ✓ Tách ra các thành phần mà có thể thay đổi trong tương lai
- ✓ Giành sự cố gắng hơn trong quá trình thiết kế ban đầu để có được các yêu cầu đúng của người sử dụng.
- ✓ Khi quyết định lựa chọn hoạt động bảo trì hay thay thế phần mềm, ta cần trả lời các câu hỏi sau:
 - ❖ Chí phí bảo trì không quá cao?
 - ❖ Tính tin cậy của hệ thống có được chấp nhận không?
 - ❖ Hệ thống có không mất đi khả năng thích nghi với các thay đổi trong tương lai với một giản đồ và chi phí hợp lý
 - ❖ Hệ thống có thực hiện dựa trên các ràng buộc đã mô tả trước không?
 - ❖ Các chức năng không làm hạn chế tính hữu ích của hệ thống?

- ❖ Không thể có các hệ thống khác mà làm công việc tương tự nhanh hơn, tốt hơn và rẻ hơn?
- ❖ Chi phí bảo trì phần cứng không trở nên đắt đến nỗi mà phần cứng có thể bị thay thế

8.1.2. Dự đoán bảo trì

Dự đoán bảo trì 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 của hệ thống. Khả năng này phụ thuộc vào khả năng bảo trì của các thành phần bị ảnh hưởng bởi sự thay đổi đó. Khi thực hiện các thay đổi có thể làm hỏng hệ thống và giảm khả năng bảo trì của nó.
- ✓ Chi phí bảo trì. Chi phí này phụ thuộc vào số lượng các thay đổi và chi phí thay đổi lại phụ thuộc vào khả năng bảo trì.

8.1.3. Dự đoán thay đổi

Dự đoán thay đổi nhằm 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. Những 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 gồm:

- ✓ Số lượng và độ phức tạp của các giao diện hệ thống
- ✓ Số lượng các yêu cầu bắt 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. Các thông tin mà một phép đo thường quan tâm:

- ✓ 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.
- ✓

8.1.2 Tiến trình bảo trì

8.1.3. Một số hiệu ứng lề khi tiến hành bảo trì

8.1.4. Những vấn đề bảo trì hiện nay

8.1.5 Các kỹ thuật bảo trì phần mềm

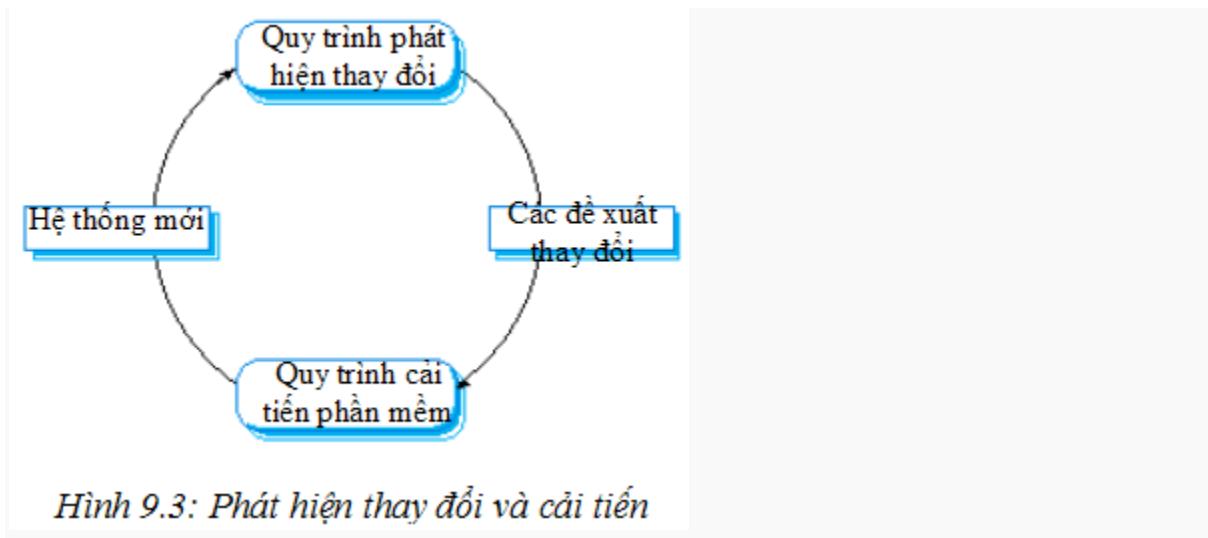
8.2 Bảo trì cải tiến phần mềm

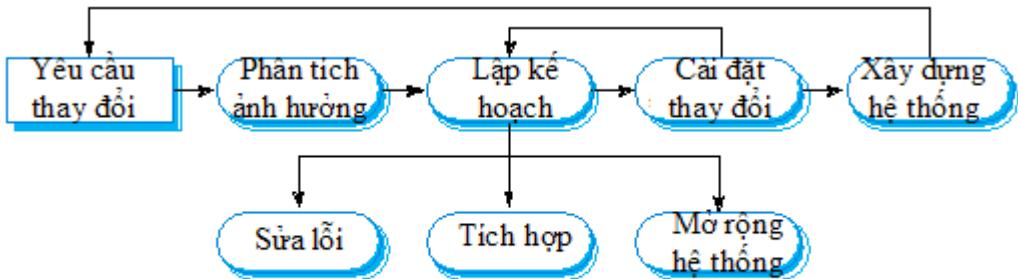
8.2.1 Quy trình bảo trì cải tiến phần mềm

Các quy trình cải tiến phần mềm phụ thuộc vào:

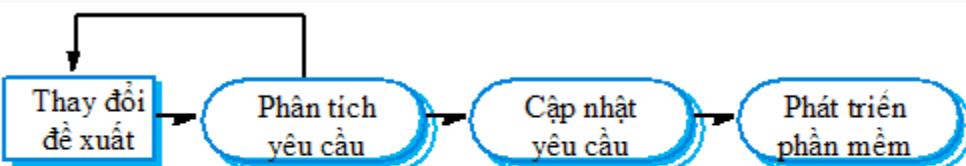
- ✓ Kiểu phần mềm cần bảo trì
- ✓ Quy trình phát triển phần mềm đã được sử dụng
- ✓ Kỹ năng và kinh nghiệm của các stakeholder.

Các đề xuất thay đổi là định hướng để cải tiến hệ thống. Phát hiện thay đổi và thực hiện 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 9.4: Quy trình cải tiến hệ thống

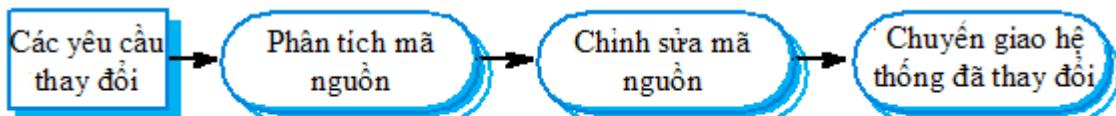


Hình 9.5: Cải đặt thay đổi

Trên đây là những quy trình cơ bản. Tuy nhiên, với các yêu cầu thay đổi khẩn cấp, ta có thể cài đặt chúng ngay 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.

Quy trình thay đổi khẩn cấp được tiến hành qua các bước sau



Hình 9.6: Quy trình cài đặt thay đổi khẩn cấp

Để cải tiến hệ thống, 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 cài 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 nên được tách hệ thống ra khỏi quy trình nghiệp vụ, chỉnh sửa quy trình nghiệp vụ và thay thế hệ thống.
- ✓ 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 kĩ nghệ hệ thống 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/tổ chức sử dụng phần mềm
- ✓ Người quản lý dây chuyền sản xuất của tổ chức
- ✓ Người quản lý công nghệ thông tin của tổ chức
- ✓ Người quản lý cao cấp

Chúng ta có thể 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 như thế nào

Để đo hệ thống, chúng ta cần thu thập các dữ liệu có định lượng để tạo ra bản đánh giá về chất lượng của hệ thống như:

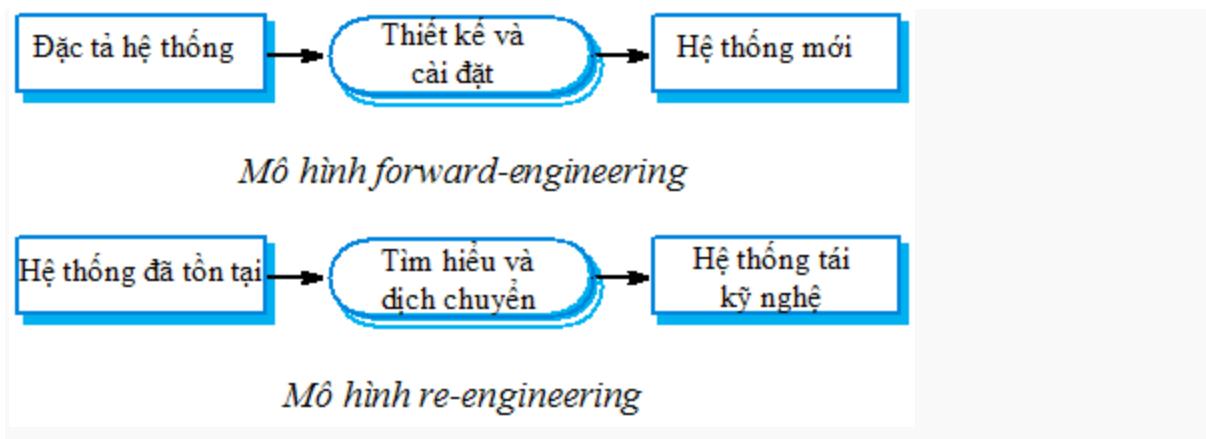
- ✓ Số lượng các yêu cầu thay đổi của hệ thống

- ✓ Số lượng các giao diện người dùng khác nhau
- ✓ Số lượng dữ liệu được sử dụng trong hệ thống.

8.2.2 Tái kĩ nghệ hệ thống

Tái kĩ nghệ hệ thống là kĩ 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 kĩ nghệ giúp giảm rủi ro vì trong quá trình xây dựng phần mềm mới rủi ro có thể xảy ra là khá cao.

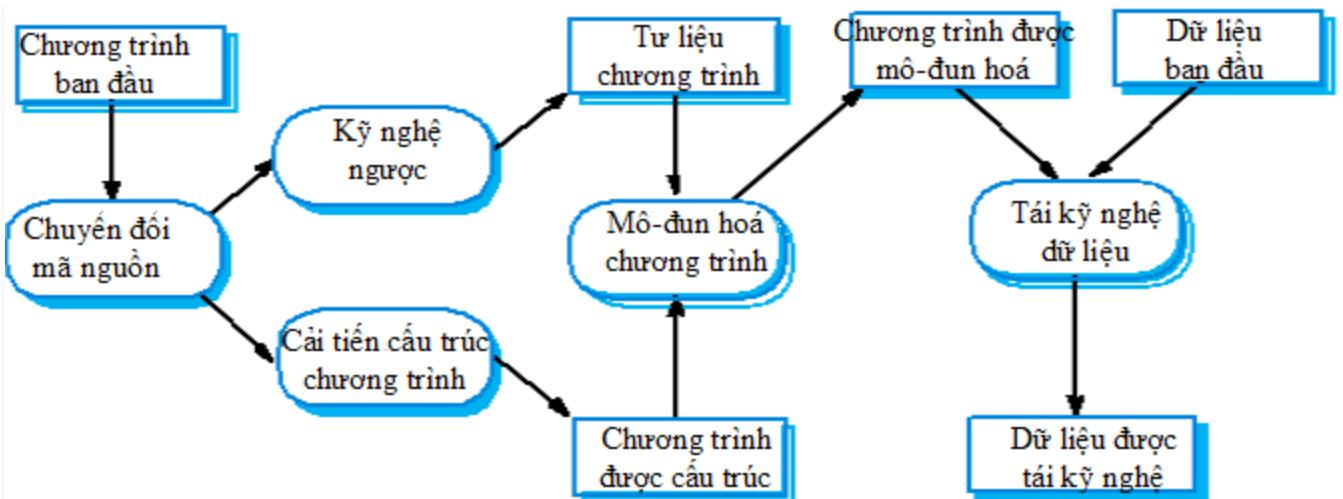
Trong tái kĩ nghệ ngược ta thường dùng quy trình kĩ nghệ ngược hệ thống. Mô hình sau đây giúp phân biệt giữa kĩ nghệ tiến (forward engineering) và tái kĩ nghệ hệ thống (re-engineering):



Quy trình tái kĩ nghệ bao gồm các hoạt động sau:

- ✓ Dịch chuyển mã nguồn: chuyển mã lệnh thành ngôn ngữ mới.
- ✓ Kĩ nghệ ngược: phân tích chương trình để tìm hiểu nó.
- ✓ Cải thiện cấu trúc chương trình
- ✓ Mô-đun hóa chương trình: tổ chức lại cấu trúc chương trình
- ✓ Tái kĩ nghệ dữ liệu: thu gọn và cấu trúc lại dữ liệu hệ thống

Hình ...sau đây mô tả quy trình tái kĩ nghệ 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 9 QUẢN LÝ DỰ ÁN

9.1 Tổng quan về quản lý dự án

9.1.1 Định nghĩa

Dự án là một nhiệm vụ cần hoàn thành để có được một sản phẩm/dịch vụ duy nhất, trong một thời hạn đã cho, với kinh phí dự kiến [thầy vi].

Theo PMI: Dự án là một sự cố gắng nhất thời được tiến hành để tạo ra một sản phẩm hay dịch vụ.

9.1.2 Các đặc trưng của dự án phần mềm

Dự án phần mềm có những đặc trưng sau:

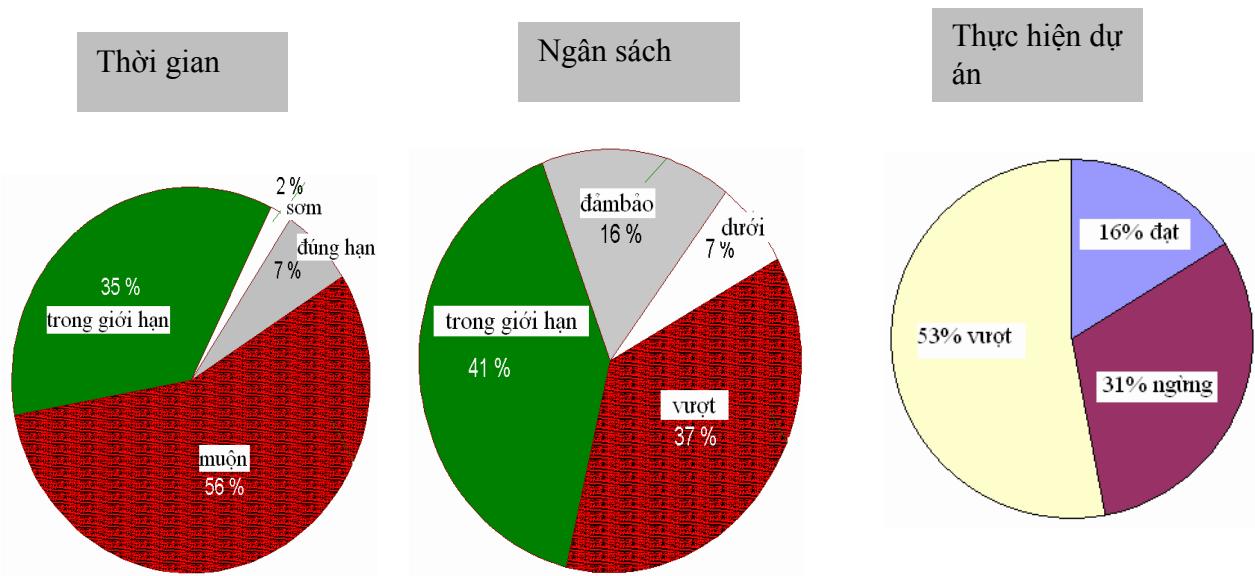
- ✓ Các hoạt động có mục tiêu xác định
- ✓ Mang tính thời điểm: có bắt đầu, có kết thúc
- ✓ Có các ràng buộc xác định: với khung khổ cứng
- ✓ Có nhiều rủi ro: thành công hay thất bại
- ✓ Sản phẩm phần mềm là vô hình
- ✓ Không được xác định duy nhất (với cùng yêu cầu)
- ✓ Không như những nguyên tắc kỹ nghệ thông thường khác (cơ, điện, ...)
- ✓ Tiến trình phát triển tùy biến, không chuẩn hóa
- ✓ Dự án nhiều biến động theo tính chất của sản phẩm và môi trường phát triển
=> với các dự án phần mềm, nếu áp dụng như quản lý dự án thông thường thì khó dẫn tới thành công.

9.1.3 Thực trạng các dự án phần mềm

Các vấn đề thường gặp đối với dự án phát triển phần mềm ngày nay gồm

- ✓ Sản phẩm không đạt yêu cầu
- ✓ Không hoàn thành đúng hạn
- ✓ Chi phí vượt dự toán
- ✓ Rủi ro là tất yếu, khó tránh.

Hình ... chỉ ra con số thống kê minh chứng cho nhận định này.

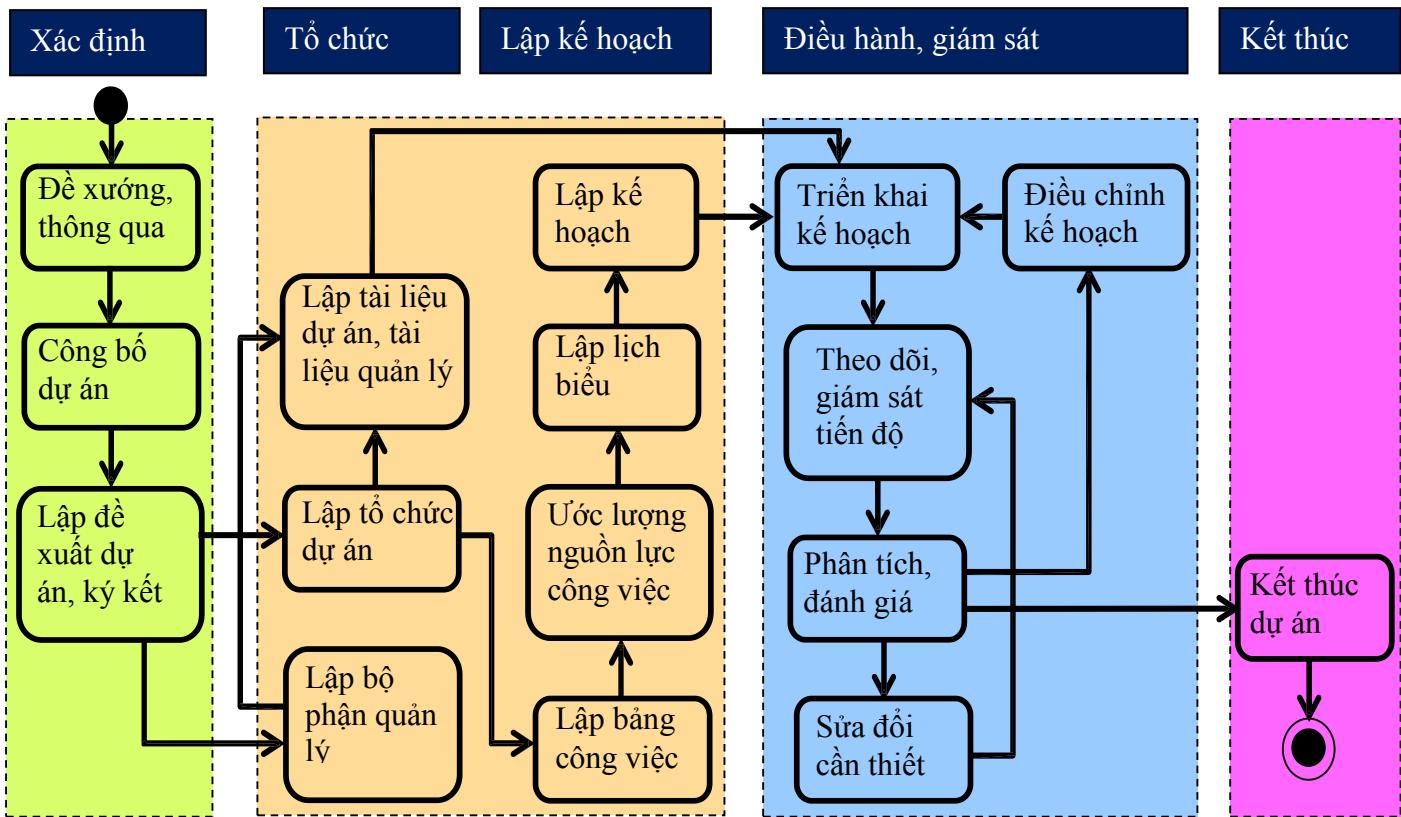


9.1.4 Mục tiêu, phương châm quản lý dự án

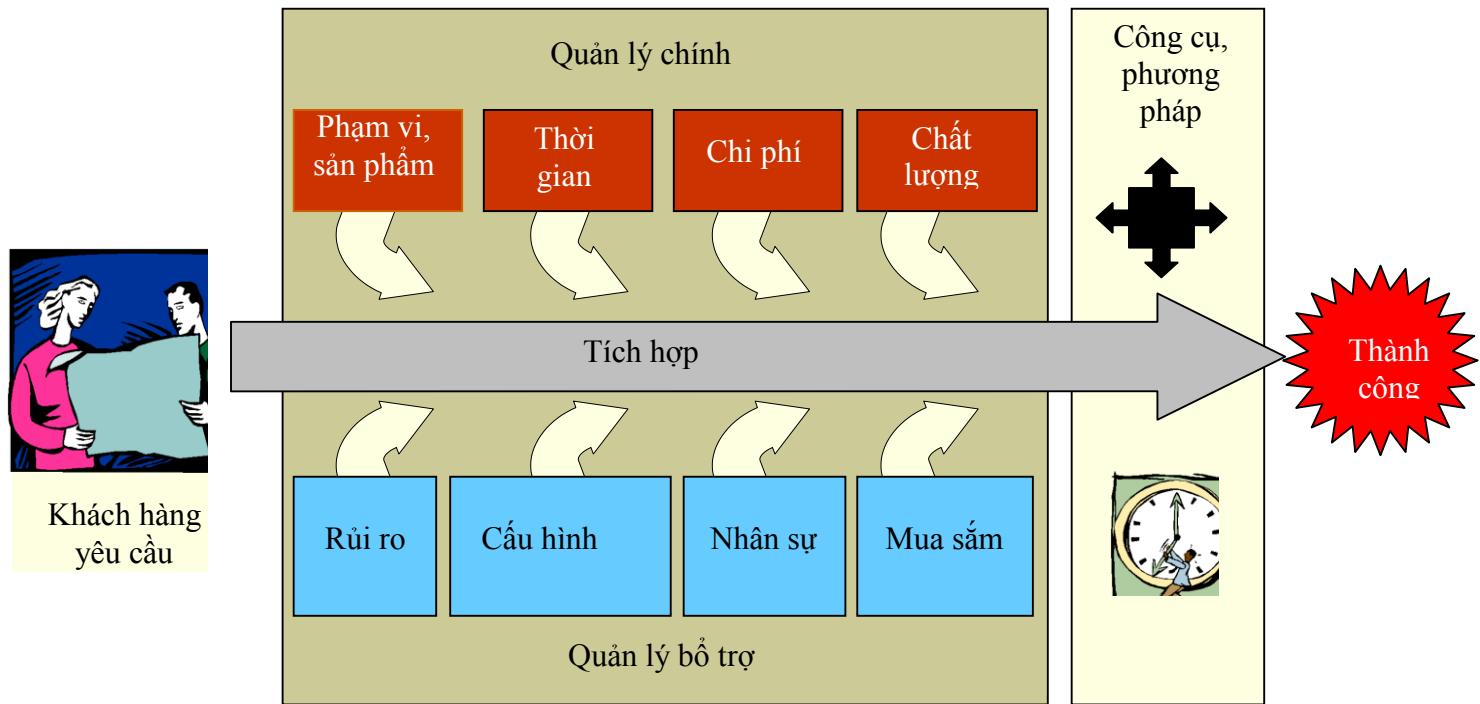
Mục tiêu của quản lý dự án là tạo ra sản phẩm bàn giao đúng thời hạn (thời gian), trong phạm vi chi phí dự toán (chi phí) và phù hợp với yêu cầu của khách hàng (chất lượng). Phương châm của quản lý dự án là tiến hành triển khai dự án:

- ✓ Theo quy trình, lịch biểu, nhưng linh hoạt
- ✓ Hướng kết quả, không hướng nhiệm vụ
- ✓ Huy động, phát huy mọi nguồn lực
- ✓ Làm rõ nhiệm vụ, trách nhiệm từng người
- ✓ Tài liệu cô đọng, chất lượng.

9.1.5 Tiến trình tổng quan triển khai dự án



8.1.6 Các chức năng quản lý dự án



9.2 Xác định dự án

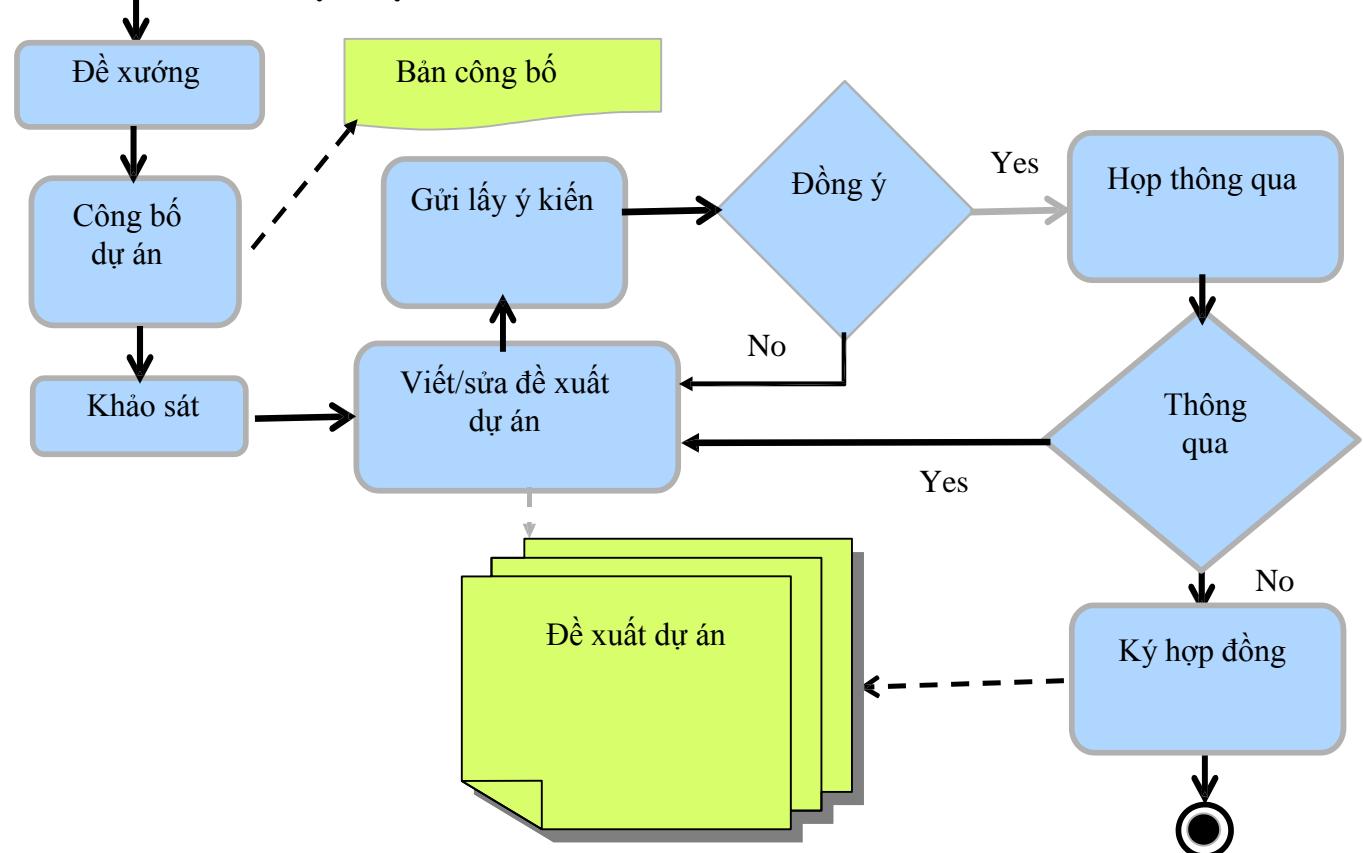
9.2.1 Giới thiệu

Xác định dự án là bước đầu tiên của quản lý dự án. Nó được thể hiện qua bản đề xuất dự án. Để dự án được thông qua nó phải thỏa mãn các tiêu chí sau:

- ✓ Dự án đáp ứng các yêu cầu của người đưa ra bao gồm các chức năng, các ràng buộc và đáp ứng sự mong đợi của họ. Do vậy cần phải đưa ra một số phương án và lựa chọn phương án thích hợp.
- ✓ Sau khi có phương án, cần lập luận tính khả thi trên các khía cạnh về kinh tế, thời gian, hoạt động, pháp lý, ...

Bản đề xuất dự án (project proposal) do người quản lý dự án viết để những người có thẩm quyền tham gia và ký kết. Bản đề xuất dự án chưa được thông qua thì không thể triển khai dự án. Người tham gia xây dựng bản đề xuất dự án phải là những người có chuyên môn cao và có kinh nghiệm. Với dự án lớn, bản đề xuất dự án có thể là một dự án.

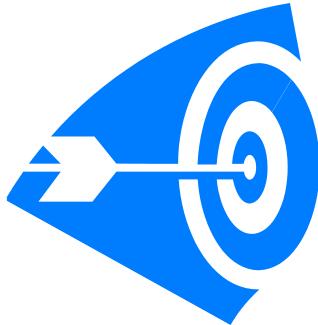
9.2.2 Tiến trình xác định dự án



Nội dung bản đề xuất dự án

1. Mục tiêu của dự án (đáp ứng yêu cầu tổ chức)

Mục tiêu của dự án thường gồm:



✓ Mục tiêu chung: Hướng lâu dài, phù hợp với mục tiêu chiến lược của tổ chức

✓ Mục tiêu cụ thể: Giải quyết những vấn đề, nhiệm vụ hiện tại của tổ chức

Với những dự án nhỏ thường chỉ có mục tiêu cụ thể. Để xác định mục tiêu cần kinh nghiệm và lấy chiến lược và nhiệm vụ của tổ chức làm cơ sở. Mục tiêu cần rõ ràng, đúng đắn vì nó là cơ sở để xây dựng tài liệu khác như phạm vi dự án, mô tả các công việc ... là yếu tố quyết định thành công dự án.

2. Vấn đề và cơ hội (sự cấp thiết, lợi ích)

Phân tích vấn đề và cơ hội là cơ sở thuyết phục nhà tài trợ hay khách hàng đầu tư. Với một nhà đầu tư, lợi ích có phạm vi rộng về kinh tế, xã hội, khoa học, ... tùy thuộc vào dự án. Hoạt động này cũng chỉ ra:

- ✓ Các khó khăn trở ngại đang làm ảnh hưởng tới mục tiêu của tổ chức, gây thiệt hại đến lợi ích của tổ chức và
- ✓ Cơ hội có được nhờ thực hiện dự án và cuối cùng là lợi ích mà nó mang lại thật xứng đáng, kỳ diệu với những đầu tư bỏ ra.

3. Giải pháp đề xuất (giải pháp công nghệ)

Thông thường cần xác định giải pháp cho từng vấn đề, sau đó tổng hợp lại để có giải pháp tổng thể.

Ví dụ:

Vấn đề	Giải pháp
Số đơn hàng đọng tăng lên	Tự động hóa việc cập nhật
Xử lý theo lô nên rất chậm	Xử lý theo thời gian thực
Tỷ lệ thay thế nhân viên cao	Tự động được giải quyết

Phát triển một hệ thống cập nhật đơn hàng được tự động hóa (với một số mức) và xử lý dữ liệu tự động ngay sau khi cập nhật để có kết quả cho khách hàng (khi cần).

Có một số phương án tương ứng với từng mức tự động. Xây dựng một số phương án để lựa chọn, sao cho đáp ứng yêu cầu của khách hàng, thỏa mãn các ràng buộc (thời gian, chi phí, ...) và phù hợp với nhà phát triển. Phương án đáp ứng yêu cầu của khách hàng có thể ở các mức độ tối thiểu, cơ bản và triệt để. Phương án cũng cần cải thiện tình thế để giảm chi phí, tăng hiệu quả để đạt lợi nhuận, ưu thế cạnh tranh.

4. Phương pháp lựa chọn phương án cho dự án (cách tiếp cận)

Khi đã có giải pháp công nghệ, cần xây dựng một số phương án ứng với các yêu cầu của khách hàng đề ra, mức độ của các phương án có thể là giải quyết vấn đề ở mức tối thiểu, mức cơ bản và mức triệt để. Sau đó:

- ✓ Tiến hành tính toán chi phí để kiểm tra sự thỏa mãn các ràng buộc
- ✓ Cho điểm từng tiêu chuẩn (theo mức ưu tiên) để đánh giá định lượng phương án
- ✓ Phân tích, so sánh có tính đến các điều kiện khác để lựa chọn phương án chấp nhận được.

5. Phân tích lợi nhuận và chi phí (khả thi kinh tế)

Để phân tích lợi nhuận và chí phí ta cần ước lượng chi phí phát triển dự án.

Ước lượng phần cứng có thể dựa trên mô hình cấu hình và giá thiết bị để tính ra

Ước lượng chi phí phát triển phần mềm luôn là bài toán khó. Hoạt động ước lượng ở giai đoạn này thường ở mức khái quát cao. Khi ước lượng, ta cần quan tâm đến ba đại lượng:

- ✓ Chi phí công lao động: số người – tháng, số người – tuần, số người – ngày => Vốn (bằng tiền) = Số ngày/giờ công * giá
- ✓ Số lao động: Người
- ✓ Phí thời gian: Ngày, tuần, tháng

Nguyên tắc chung khi ước lượng: Phân nhỏ nhiệm vụ, ước lượng từng phân từ dưới lên, rồi cộng lại. Tiêu chí phân chia có thể theo sản phẩm trọn vẹn (1 module) hoặc theo giai đoạn: đặc tả, thiết kế, mã hóa.

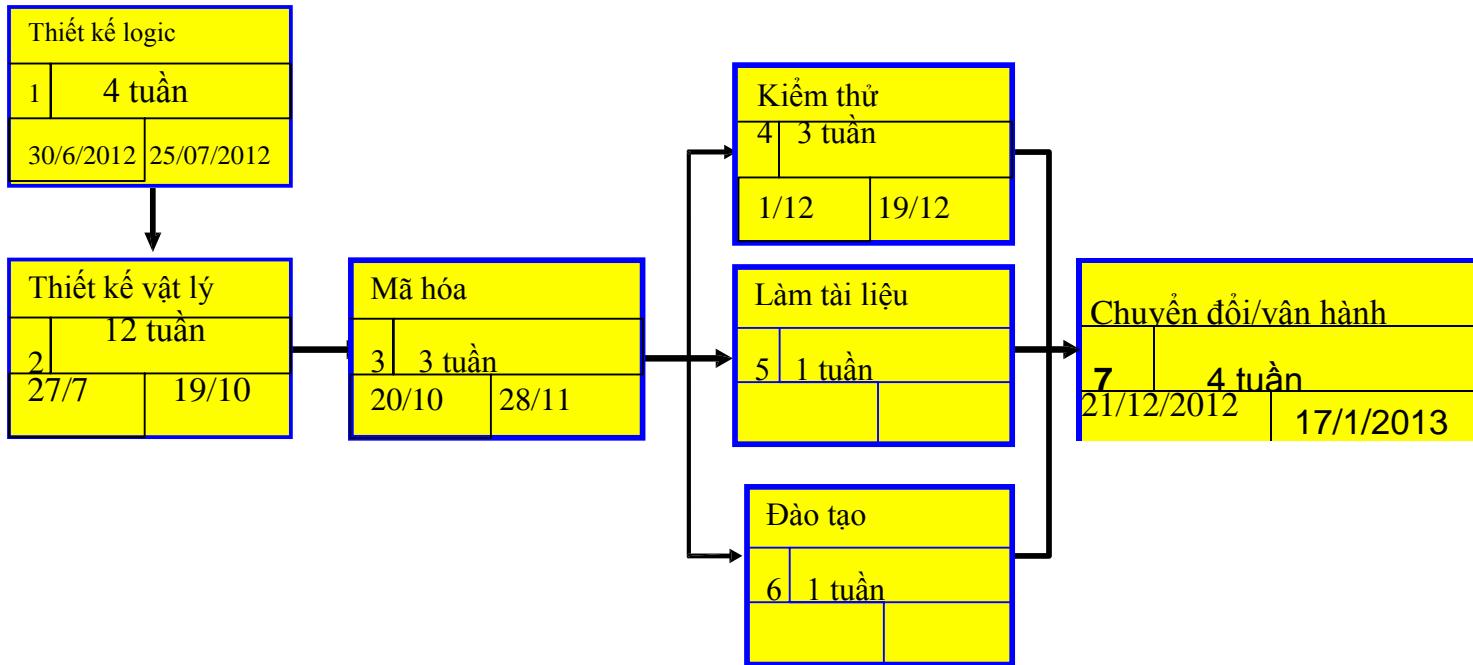
6. Các yêu cầu về nghiệp vụ (sự công tác)
7. Phạm vi dự án (hoạt động chính, bộ phận người – người liên quan và trách nhiệm)
8. Những cản trở và khó khăn chính (lường hết)
9. Phân tích các rủi ro (khả thi khác)
10. Tổng quan lịch trình thực hiện (thời gian kết thúc, các mốc lớn, khả thi thời gian)

Mốc lớn lịch trình cần chỉ ra:

- ✓ Thời gian kết thúc dự án
 - ✓ Các giai đoạn chính đánh dấu bằng một sản phẩm và thời gian lịch tương ứng.
- Các mốc chính thường được chọn lập lịch là:

- Kỹ nghệ hệ thống
- Xác định yêu cầu
- Thiết kế
- Lập trình – kiểm thử đơn vị
- Kiểm thử thẩm định
- Cài đặt và vận hành.

Ví dụ về lịch trình dự án



11. Ma trận trách nhiệm (quan hệ/trách nhiệm)

12. Kế hoạch truyền thông (đảm bảo thông tin)

Tài liệu cuối cùng chỉ cần khi dự án được triển khai, không cần khi dự án mới đề xuất

9.3 Lập kế hoạch dự án

9.3.1 Lập kế hoạch dự án

a. Kế hoạch dự án

Kế hoạch là bản dự kiến công việc (cái gì), người làm (ai), thời gian làm (khi nào, bao lâu), phương tiện dùng (cái gì, bao nhiêu), sản phẩm ra (cái gì), tiêu chí cần có. Đây là công việc lặp lại suốt quá trình dự án. Có nhiều bản kế hoạch cần lập để quản lý dự án và chúng là công cụ chính để quản lý dự án.

Trong quá trình lập lịch, ta cần xét đến các nguồn lực cho dự án gồm con người, phần mềm dùng lại được, phần cứng/công cụ phần mềm chia sẻ. Trong đó nguồn lực con người là quan trọng nhất, mỗi thành viên cần phải có các năng lực nhất định và đồng thời cũng cần có cơ cấu nhân sự phù hợp để điều khiển nhóm. Ngoài nhân tố con người, ta cần xét đến khả năng tái sử dụng lại các phần mềm như

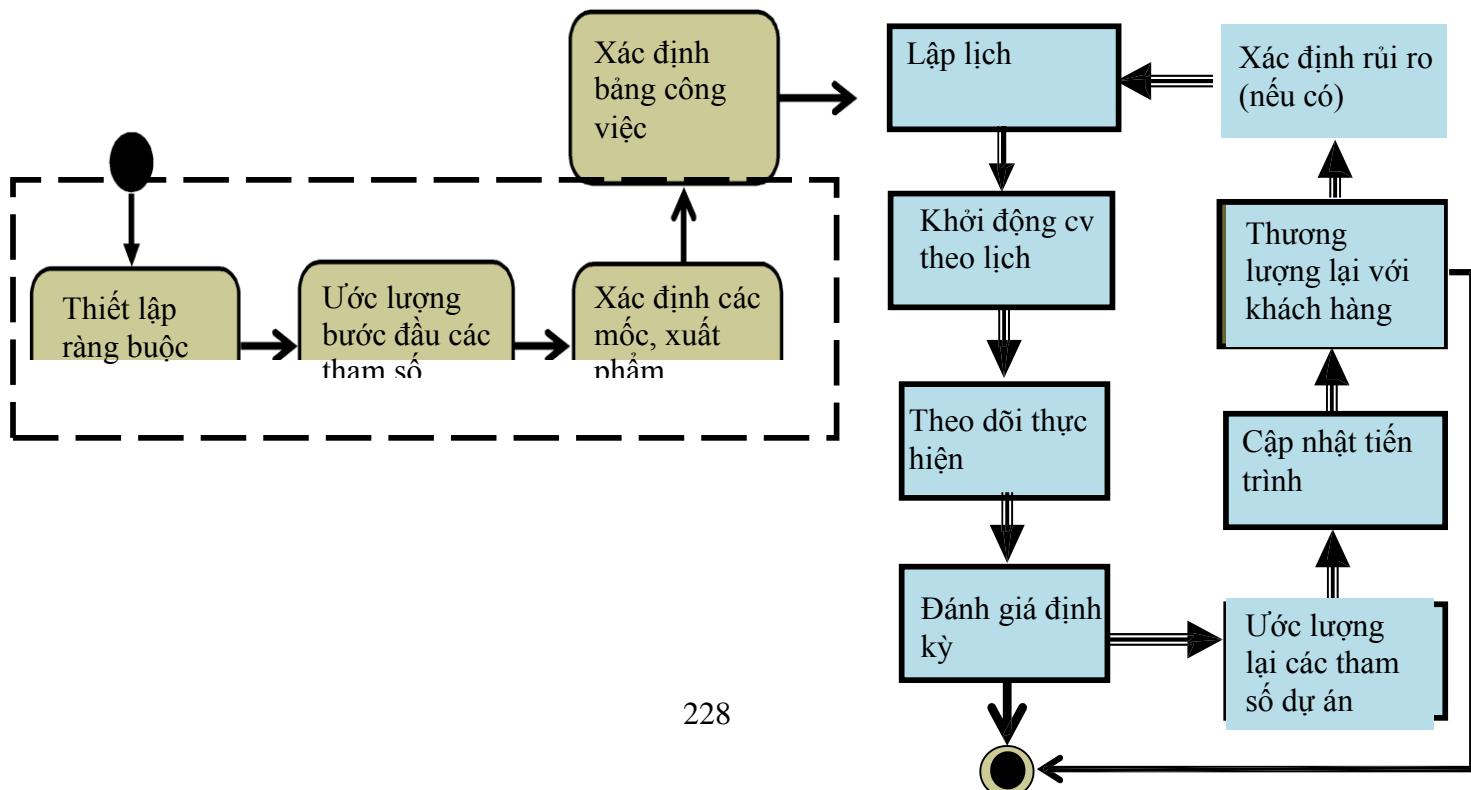
các thành phần đóng gói có thể dùng lại ngay, các thành phần đã được kiểm nghiệm tốt, để sử dụng ta có thể cần sửa đổi đôi chút.

Các kế hoạch là công cụ chính để quản lý dự án, ngoài kế hoạch chính cho hoạt động phát triển/cải tiến hệ thống, chúng ta cần lập các bảng kế hoạch khác như mô tả trong bảng

Tên kế hoạch	Mô tả
Kế hoạch chất lượng	Mô tả thủ tục và các chuẩn chất lượng áp dụng
Kế hoạch thẩm định	Mô tả cách thức, nguồn lực và lịch trình thẩm định
Kế hoạch quản lý cầu hình	Mô tả cấu hình, thủ tục và tiến trình quản lý cầu hình
Kế hoạch bảo trì	Chỉ ra các yêu cầu, chi phí và nguồn lực cần cho bảo trì
Kế hoạch phát triển đội ngũ	Mô tả số lượng, kỹ năng và kinh nghiệm của thành viên dự án cần có

b. Tiến trình triển khai kế hoạch dự án

Tiến trình triển khai dự án bao gồm các hoạt động được lặp lại trong suốt thời gian phát triển dự án như mô tả trong hình



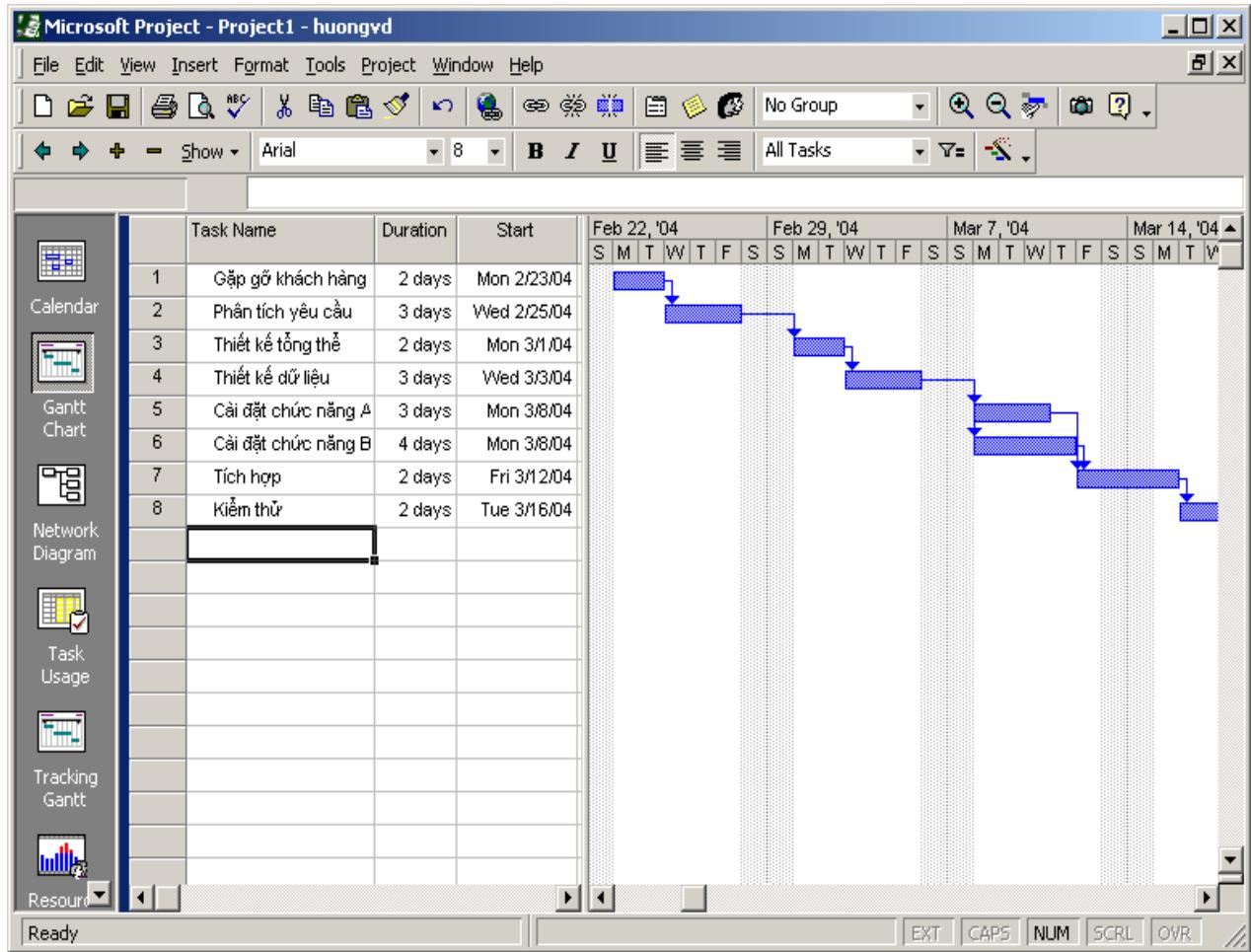
c. Cấu trúc chung của một bản kế hoạch dự án

- ✓ Mở đầu
- ✓ Tổ chức dự án
- ✓ Phân tích rủi ro
- ✓ Các yêu cầu về nguồn lực
 - Nhân lực
 - Phần cứng, phần mềm
- ✓ Phân chia công việc
- ✓ Lịch trình dự án
- ✓ Cơ chế điều hành và báo cáo

9.3.2 Một số công cụ quản lý dự án

Loại dự án	Phần mềm quản lý	Chức năng
Lớn, phức tạp	Primavera, Artimis, OpenPlan	Đáp ứng mọi chức năng
Trung bình	Project Management Workbench, SuperProject	Đáp ứng mọi chức năng
Nhỏ	Microsoft Project, Fast Track, TimeLine, MacProject	Đơn giản, dễ sử dụng, chủ yếu đáp ứng chức năng lập kế hoạch. Chưa đáp ứng việc điều hành, giám sát dự án

Ví dụ: Lập lịch của Microsoft Project



TÀI LIỆU THAM KHẢO

- [1] Ian Sommerville's , 2005, *Software Engineering*; 7th Ed.,
 - [2] Roger S. Pressman (dịch: Ngô Trung Việt), 1997, *Kỹ nghệ phần mềm*, Tập I,II,III, NXB Giáo dục.
 - [3] Lê Đức Trung, 2001, *Công nghệ phần mềm*, NXB Khoa học và Kỹ thuật.,
 - [4] Ngô Trung Việt, Nguyễn Kim ánh (biên soạn), 2003, *Nhập môn Công nghệ phần mềm*, NXB Khoa học và kỹ thuật.
 - [5] Stephen R. Schach, 1999, *Classical and Objecture Oriented Software Engineering with UML and C++*, 4th ed., McGraw-Hill.
- [MCG91] McMlaughli,R., *Một số chú ý về thiết kế chương trình*, *Software Engineering Notes*, vol.16, no.4, oct 1991, pp53-54.
- [SHA95a] Shaw,M and D.Garlan, *Formulation and formalisms in software achitecture*, volume 100-lecture Notes in computer Science, Springer-verlag,1995