

CÔNG NGHỆ PHẦN MỀM

Ấn bản 2015

MỤC LỤC

MỤC LỤC	I
HƯỚNG DẪN.....	V
BÀI 1: GIỚI THIỆU	1
 1.1 CÁC KHÁI NIỆM CƠ BẢN	1
1.1.1 Phần mềm.....	1
1.1.2 Chất lượng phần mềm.....	5
1.1.3 Công nghệ Phần mềm.....	7
 1.2 QUY TRÌNH CÔNG NGHỆ PHẦN MỀM.....	10
1.2.1 Bước Xác định.....	10
1.2.2 Bước Phát triển.....	11
1.2.3 Bước Bảo trì (Vận hành).....	12
 1.3 MỘT SỐ MÔ HÌNH TRIỂN KHAI.....	12
1.3.1 Mô hình Thác nước	12
1.3.2 Mô hình Bản mẫu Phần mềm	15
1.3.3 Mô hình Xoắn ốc	16
 1.4 CÁC PHƯƠNG PHÁP XÂY DỰNG PHẦN MỀM.....	16
1.4.1 Khái niệm.....	16
1.4.2 Phân loại.....	17
1.4.3 Các phương pháp xây dựng phần mềm	17
 1.5 CÔNG CỤ & MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM	21
1.5.1 Mở đầu	21
1.5.2 Phần mềm hỗ trợ thực hiện các giai đoạn	22
1.5.3 Phần mềm hỗ trợ tổ chức, quản lý việc triển khai	22
 1.6 YÊU CẦU ĐỐI VỚI KỸ SƯ PHẦN MỀM	23
TÓM TẮT	24
BÀI 2: TÁC VỤ PHÂN TÍCH & ĐẶC TẢ YÊU CẦU.....	25
 2.1 TỔNG QUAN.....	25
 2.2 QUÁ TRÌNH PHÂN TÍCH	25
2.2.1 Phân tích Phạm vi dự án.....	25
2.2.2 Phân tích Mở rộng Yêu cầu Nghiệp vụ	27
2.2.3 Phân tích Yêu cầu Bảo mật	29
2.2.4 Phân tích Yêu cầu Tốc độ.....	31
2.2.5 Phân tích Yêu cầu Vận hành	32
2.2.6 Phân tích Khả năng Mở rộng Yêu cầu	33
2.2.7 Phân tích Yêu cầu Sẵn dùng	34
2.2.8 Phân tích Yếu tố Con người.....	34
2.2.9 Phân tích Yêu cầu Tích hợp	34
2.2.10 Phân tích Thực tiễn Nghiệp vụ tồn tại.....	35

2.2.11 Phân tích Yêu cầu Khả năng và Quy mô	36
2.3 XÁC ĐỊNH YÊU CẦU.....	36
2.3.1 Yêu cầu và Mô tả yêu cầu.....	37
2.3.2 Phân loại yêu cầu	38
2.3.3 Các bước xác định yêu cầu	41
2.4 MÔ HÌNH HOÁ YÊU CẦU HỆ THỐNG	48
2.4.1 Các Nguyên lý Mô hình hóa	48
2.4.2 Sơ đồ phân rã chức năng (FDD)	49
2.4.3 Mô hình bản mẫu (Prototype)	51
2.4.4 Sơ đồ luồng dữ liệu (Data Flow Diagram, DFD)	52
2.4.5 Mô hình hướng đối tượng	52
2.4.6 Ví dụ minh họa từ yêu cầu sang mô hình hóa.....	57
TÓM TẮT	58
BÀI 3: TÁC VỤ THIẾT KẾ PHẦN MỀM.....	59
3.1 TỔNG QUAN VỀ THIẾT KẾ.....	59
3.1.1 Kỹ thuật thiết kế.....	60
3.1.2 Thiết kế giao diện người dùng	67
3.1.3 Thiết kế hướng chức năng	69
3.1.4 Thiết kế hướng đối tượng	69
3.2 KIẾN TRÚC PHẦN MỀM.....	70
3.3 PHƯƠNG PHÁP THIẾT KẾ PHẦN MỀM	72
3.4 VÍ DỤ MINH HỌA	74
TÓM TẮT	76
BÀI 4: TÁC VỤ THIẾT KẾ & TỔ CHỨC DỮ LIỆU	77
4.1 TỔNG QUAN	77
4.2 KẾT QUẢ CỦA THIẾT KẾ	77
4.3 QUÁ TRÌNH THIẾT KẾ	81
4.4 PHƯƠNG PHÁP THIẾT KẾ DỮ LIỆU	85
4.4.1 Phương pháp Trực tiếp.....	85
4.4.2 Phương pháp Gián tiếp.....	86
TÓM TẮT	92
BÀI 5: TÁC VỤ THIẾT KẾ GIAO DIỆN	93
5.1 TỔNG QUAN	93
5.1.1 Kết quả thiết kế	94
5.1.2 Phân loại màn hình giao diện	96
5.1.3 Quá trình thiết kế.....	97
5.1.4 Các vấn đề khác.....	100
5.1.5 Một số chú ý chung	103
5.2 THIẾT KẾ MÀN HÌNH.....	104
5.2.1 Mô tả màn hình chính	104
5.2.2 Thiết kế màn hình chính dùng thực đơn (menu)	105

5.2.3 Ví dụ	105
5.3 THIẾT KẾ MÀN HÌNH TRA CỨU	106
5.3.1 Mô tả màn hình tra cứu.....	106
5.3.2 Thể hiện tiêu chuẩn tra cứu	107
5.3.3 Thể hiện kết quả tra cứu	107
5.3.4 Thao tác người dùng và xử lý của phần mềm	109
5.4 THIẾT KẾ MÀN HÌNH NHẬP LIỆU	110
5.4.1 Mô tả màn hình nhập liệu	110
5.4.2 Thiết kế màn hình nhập liệu dạng danh sách	112
5.4.3 Thiết kế màn hình nhập liệu dạng hồ sơ.....	113
5.4.4 Thiết kế màn hình nhập liệu dạng phiếu.....	113
TÓM LƯỢC.....	114
BÀI 6: TÁC VỤ XÂY DỰNG CHƯƠNG TRÌNH.....	115
6.1 TỔNG QUAN.....	115
6.2 MÔI TRƯỜNG LẬP TRÌNH.....	118
6.2.1 Tiêu chí về Chất lượng của ngôn ngữ lập trình.....	118
6.2.2 Tiêu chí về Khả năng mô-đun hóa của ngôn ngữ lập trình.....	118
6.2.3 Tiêu chí về Giá trị tài liệu kỹ thuật của ngôn ngữ lập trình	119
6.2.4 Tiêu chí về Cấu trúc dữ liệu trong ngôn ngữ lập trình	119
6.2.5 Ví dụ	120
6.3 PHONG CÁCH LẬP TRÌNH	120
6.3.1 Tính cấu trúc	121
6.3.2 Ưu điểm của diễn đạt.....	121
6.3.3 Cách thức trình bày bên ngoài	123
6.4 ĐÁNH GIÁ CHẤT LƯỢNG CÔNG VIỆC	123
6.4.1 Hiện thực tăng cường.....	123
6.4.2 Đánh giá lại thiết kế và chương trình (Design and Code Review)	124
6.5 VÍ DỤ MINH HOẠ	125
TÓM TẮT	126
BÀI 7: TÁC VỤ KIỂM THỬ PHẦN MỀM	127
7.1 TỔNG QUAN.....	127
7.1.1 Lịch sử	127
7.1.2 Giới thiệu	128
7.1.3 Mô hình chữ V.....	129
7.2 YÊU CẦU ĐỐI VỚI KIỂM THỬ.....	130
7.3 CÁC KỸ THUẬT KIỂM THỬ.....	131
7.3.1 Phương pháp Hộp đen (Black box).....	131
7.3.2 Phương pháp Hộp trắng (White box).....	131
7.3.3 Phương pháp Hộp xám (Grey box).....	132
7.4 CHIẾN LƯỢC & CÁC GIAI ĐOẠN.....	133
7.4.1 Kiểm thử Đơn vị (Unit Test)	134

7.4.2 Kiểm thử Chức năng (Functional Test)	136
7.4.3 Kiểm thử Tích hợp (Integration Test)	136
7.4.4 Kiểm thử Hệ thống (System Test)	138
7.4.5 Kiểm thử Chấp nhận (Acceptance Test).....	138
7.4.6 Kiểm thử beta.....	139
7.4.7 Một số công việc khác.....	139
7.5 VÍ DỤ MINH HOẠ	141
TÓM TẮT	143
BÀI 8: TÁC VỤ BẢO TRÌ PHẦN MỀM	144
8.1 GIỚI THIỆU.....	144
8.2 TẦM QUAN TRỌNG CỦA BẢO TRÌ	147
8.3 KẾ HOẠCH BẢO TRÌ PHẦN MỀM	148
8.4 QUY TRÌNH BẢO TRÌ PHẦN MỀM	149
TÓM TẮT & ÔN TẬP	150
BÀI 9: QUẢN TRỊ DỰ ÁN PHẦN MỀM	151
9.1 GIỚI THIỆU.....	151
9.1.1 Khái niệm dự án.....	151
9.1.2 Các vấn đề thường xảy ra đối với một dự án phần mềm	151
9.2 TÓM LƯỢC VỀ QUẢN TRỊ DỰ ÁN	152
9.3 HOẠT ĐỘNG CỦA QUẢN TRỊ DỰ ÁN	153
9.4 ĐỘ ĐO PHẦN MỀM.....	156
9.4.1 Đo lường kích cỡ phần mềm	156
9.4.2 Độ đo dựa trên thống kê	157
9.5 CÁC TÁC VỤ CẦN THIẾT.....	157
9.5.1 Ước lượng.....	157
9.5.2 Quản lý nhân sự.....	159
9.5.3 Quản lý cấu hình.....	160
9.5.4 Quản lý rủi ro	161
TÓM TẮT	162
BÀI 10: QUY TRÌNH PHÁT TRIỂN PHẦN MỀM	163
10.1 GIỚI THIỆU	163
10.2 GIỚI THIỆU VỀ QUY TRÌNH.....	164
10.3 QUY TRÌNH ISO, CMM/CMMI	166
TÓM TẮT	168
PHỤ LỤC A – BÀI TẬP	169
PHỤ LỤC B – THAM KHẢO	178
PHỤ LỤC C – QUY TRÌNH RUP	181
TÀI LIỆU THAM KHẢO.....	195

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Môn học này nhằm cung cấp cho các sinh viên các kiến thức cơ sở liên quan đến các đối tượng chính yếu trong lĩnh vực Công nghệ Phần mềm (quy trình công nghệ, phương pháp kỹ thuật thực hiện, phương pháp tổ chức quản lý, công cụ và môi trường triển khai phần mềm,...), đồng thời giúp sinh viên hiểu và biết tiến hành xây dựng phần mềm một cách có hệ thống, có phương pháp. Trong quá trình học, sinh viên sẽ được giới thiệu nhiều phương pháp khác. Từ đó giúp sinh viên hiểu, phân tích và vận dụng linh hoạt các mô hình và phương pháp xây dựng phần mềm để hiện thực phát triển phần mềm tùy theo nhu cầu thực tế của bản thân, tổ chức và doanh nghiệp cũng như các kiến thức khác.

NỘI DUNG MÔN HỌC

- Bài 1: GIỚI THIỆU: trình bày các khái niệm cơ bản, quy trình công nghệ phần mềm, các phương pháp xây dựng phần mềm, công cụ & môi trường phát triển phần mềm, yêu cầu đối với kỹ sư phần mềm.
- Bài 2: TÁC VỤ PHÂN TÍCH & ĐẶC TẢ YÊU CẦU: giới thiệu tổng quan và mô hình hoá yêu cầu hệ thống.
- Bài 3: TÁC VỤ THIẾT KẾ PHẦN MỀM: trình bày tổng quan về thiết kế, kiến trúc phần mềm, phương pháp thiết kế phần mềm, ví dụ minh họa.
- Bài 4: TÁC VỤ THIẾT KẾ & TỔ CHỨC DỮ LIỆU: giới thiệu tổng quan về thiết kế, kết quả của thiết kế, quá trình thiết kế, phương pháp thiết kế dữ liệu.
- Bài 5: TÁC VỤ THIẾT KẾ GIAO DIỆN: trình bày tổng quan về thiết kế màn hình, thiết kế màn hình tra cứu, thiết kế màn hình nhập liệu.
- Bài 6: TÁC VỤ XÂY DỰNG CHƯƠNG TRÌNH: giới thiệu tổng quan về tác vụ hiện thực chương trình, môi trường lập trình, phong cách lập trình, đánh giá chất lượng công việc, ví dụ minh họa.
- Bài 7: TÁC VỤ KIỂM THỬ PHẦN MỀM: trình bày các vấn đề tổng quan về kiểm thử, yêu cầu đối với kiểm thử, các kỹ thuật kiểm thử, chiến lược kiểm thử & các giai đoạn, ví dụ minh họa.

- Bài 8: TÁC VỤ BẢO TRÌ PHẦN MỀM: giới thiệu về tầm quan trọng của bảo trì phần mềm, kế hoạch bảo trì phần mềm, quy trình bảo trì phần mềm.
- Bài 9: QUẢN TRỊ DỰ ÁN PHẦN MỀM: giới thiệu tóm lược về quản trị dự án, hoạt động của quản trị dự án, độ đo phần mềm, các tác vụ cần thiết.
- Bài 10: QUY TRÌNH PHÁT TRIỂN PHẦN MỀM: giới thiệu về quy trình, quy trình ISO, CMM/CMMI.

KIẾN THỨC TIỀN ĐỀ

Môn học Công nghệ Phần mềm đòi hỏi sinh viên có kiến thức về ngôn ngữ lập trình và đã từng lập trình các ứng dụng cơ bản, có khả năng áp dụng những cấu trúc dữ liệu và giải thuật, có hiểu biết về kiến trúc máy tính, mạng máy tính.

YÊU CẦU MÔN HỌC

Người học cần đi học đầy đủ, đọc các nội dung sẽ được học trước khi đến lớp, làm các bài tập về nhà và đảm bảo thời gian tự học ở nhà.

CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần ôn các bài đã học, trả lời các câu hỏi và làm đầy đủ bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, người học đọc trước mục tiêu và tóm tắt bài học, sau đó đọc nội dung bài học. Kết thúc mỗi ý của bài học, người đọc trả lời câu hỏi ôn tập và kết thúc toàn bộ bài học, người đọc làm các bài tập.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Môn học được đánh giá gồm hai thành phần.

- Điểm quá trình: 30%. Hình thức và nội dung do giáo viên quyết định, phù hợp với quy chế đào tạo và tình hình thực tế tại nơi tổ chức học tập.
- Điểm thi: 70%. Đánh giá dựa trên đồ án môn học.

BÀI 1: GIỚI THIỆU

Học xong bài này người học sẽ:

- *Hiểu được các khái niệm cơ bản về Công nghệ Phần mềm.*
- *Biết được quy trình công nghệ phần mềm, các phương pháp xây dựng phần mềm, các công cụ & môi trường phát triển phần mềm.*

1.1 CÁC KHÁI NIỆM CƠ BẢN

1.1.1 Phần mềm

❖ Các khái niệm

- *Chương trình máy tính:* là trình tự các chỉ thị để hướng dẫn máy tính làm việc nhằm hoàn thành công việc nào đó do người dùng yêu cầu.
- *Phần mềm:* là một hệ thống các chương trình có thể thực hiện trên máy tính nhằm hỗ trợ các chuyên gia trong từng lĩnh vực chuyên ngành thực hiện tốt nhất các thao tác nghiệp vụ của mình.
- *Nhiệm vụ chính yếu của phần mềm:* là để các chuyên gia thực hiện các công việc của họ trên máy tính dễ dàng và nhanh chóng hơn so với khi thực hiện cùng công việc đó trong thế giới thực.
- *Hoạt động của phần mềm:* là sự mô phỏng lại các hoạt động của thế giới thực trong một góc độ thu hẹp nào đó trên máy tính.
- *Quá trình sử dụng một phần mềm:* là quá trình thực hiện các công việc trên máy tính để hoàn tất công việc tương đương trong thế giới thực.
- *Lớp phần mềm:* là hệ thống các phần mềm trên cùng lĩnh vực hoạt động nào đó. Do cùng lĩnh vực hoạt động nên các phần mềm này thường có cấu trúc và chức năng (công việc mà người dùng thực hiện trên máy tính) tương tự nhau.

- *Mục tiêu của ngành công nghệ phần mềm:* là hướng đến không những xây dựng được các phần mềm có chất lượng mà còn cho phép xây dựng dễ dàng một phần mềm mới từ các phần mềm đã có sẵn trong cùng lĩnh vực (thậm chí trong các lĩnh vực khác).

Bảng 1.1: Các phần mềm và lớp phần mềm tương ứng

STT	Lớp phần mềm	Các phần mềm
1	Hỗ trợ giải bài tập	lượng giác, hình học, giải tích, số học, ...
2	Trò chơi	cờ carô, cờ tướng, cờ vua, xếp hình, ...
3	Xếp lịch biểu	thi đấu, thời khóa biểu, hội nghị, ...
4	Xét tuyển	nhân sự, học sinh lớp 10...
5	Bình chọn	Sản phẩm, cầu thủ, ...
6	Quản lý học sinh	Mầm non, trung học, trung tâm...
7	Bán hàng	thuốc tây, vật liệu xây dựng, máy tính
8	Quản lý thuê bao	điện, điện thoại, nước, ...
9	Cho mượn	sách, truyện, phim, ...

❖ Phân loại

- Phần mềm hệ thống:
 - Là những phần mềm đảm nhận công việc tích hợp và điều khiển các thiết bị phần cứng,
 - Tạo ra môi trường thuận lợi để các phần mềm khác và người sử dụng thao tác trên đó như một khối thống nhất mà không cần quan tâm đến những chi tiết kỹ thuật phức tạp bên dưới (như cách trao đổi dữ liệu giữa bộ nhớ chính và đĩa, cách hiển thị văn bản lên màn hình ...)
- Phần mềm ứng dụng:
 - Là những phần mềm được dùng để thực hiện công việc xác định,
 - Có thể chỉ gồm một chương trình đơn giản (như chương trình xem ảnh) hoặc một nhóm các chương trình cùng tương tác với nhau để thực hiện một công việc nào đó (như chương trình xử lý bản tính, chương trình xử lý văn bản ...)

Ngoài ra, phần mềm còn được chia làm 2 loại như sau:

- *Sản phẩm đại trà (Generic Product)*: được phát triển để bán ra ngoài thị trường, đối tượng sử dụng tương đối đa dạng và phong phú. Những sản phẩm loại này thường là những phần mềm dành cho máy tính.
- *Sản phẩm theo đơn đặt hàng (Bespoke Product hoặc Customised Product)*: được phát triển theo yêu cầu cho khách hàng riêng lẻ. Ví dụ: hệ thống phần mềm chuyên dụng cho một doanh nghiệp riêng lẻ ...

❖ **Kiến trúc phần mềm**

Về mặt cấu trúc, phần mềm bao gồm 3 thành phần:

Thành phần Giao tiếp (giao diện)

Đây là hệ thống các phương thức chuyên về việc nhập/xuất dữ liệu (hàm nhập/xuất) cùng với hình thức trình bày và tổ chức lưu trữ dữ liệu tương ứng, mục tiêu chính của các hàm này là đưa dữ liệu từ thế giới bên ngoài phần mềm vào bên trong hoặc ngược lại. Cụ thể là:

- Tiếp nhận yêu cầu về việc cần thực hiện, cung cấp nguồn dữ liệu liên quan đến việc đó hoặc từ thiết bị thu thập dữ liệu (cân, đo nhiệt độ ...)
- Trình bày các kết quả của việc thực hiện các yêu cầu cho người dùng (kết quả của công việc khi thực hiện trên máy tính) hoặc điều khiển hoạt động các thiết bị điều khiển (đóng mở cửa, bật mở máy...)

Thành phần Dữ liệu

Đây là hệ thống các chức năng chuyên về đọc ghi dữ liệu (hàm đọc/ghi) cùng với mô hình tổ chức dữ liệu tương ứng. Mục tiêu chính của các chức năng này là chuyển đổi dữ liệu giữa bộ nhớ chính và bộ nhớ phụ. Cụ thể là:

- Cho phép lưu trữ lại (hàm ghi) các kết quả đã xử lý (việc mượn sách đã được kiểm tra hợp lệ ...) trên bộ nhớ phụ với tổ chức lưu trữ được xác định trước (tập tin có cấu trúc, tập tin nhị phân, cơ sở dữ liệu).
- Cho phép truy xuất lại (hàm đọc) các dữ liệu đã lưu trữ phục vụ cho các hàm xử lý tương ứng.

Thành phần xử lý

Đây là hệ thống chức năng chuyên về xử lý tính toán, biến đổi dữ liệu. Chúng dùng dữ liệu nguồn từ chức năng trong thành phần giao diện (hàm nhập) hay thành phần dữ liệu (hàm đọc dữ liệu) để kiểm tra tính hợp lệ (hàm kiểm tra), tiến hành xử lý (hàm xử lý) nếu cần thiết, để tạo kết quả và hiển thị cho người xem qua chức năng trong thành phần giao diện (hàm xuất), hoặc lưu trữ bằng chức năng trong thành phần dữ liệu (hàm ghi). Cụ thể là:

- Kiểm tra tính hợp lệ của dữ liệu nguồn do người dùng nhập theo quy trình ràng buộc trong thế giới thực (cho mượn tối đa 3 quyển sách, ...)
- Xử lý tạo kết quả mong đợi theo quy định có trong thế giới thực (quy tắc tính tiền phạt khi trả sách trễ ...) hoặc theo thuật giải tự đề xuất,
- Việc xử lý dựa trên dữ liệu nguồn từ người sử dụng cung cấp hoặc dữ liệu lưu trữ đã có sẵn hoặc cả hai tùy vào xử lý cụ thể.
- Tương tự, việc xử lý cho ra kết quả có thể dùng để xuất cho người xem qua thành phần giao diện (xuất tiền phạt ...), hay có thể lưu trữ lại qua thành phần dữ liệu (số sách hiện đang được mượn của một độc giả...) hoặc cả hai (bảng tồn kho ...)

Bảng 1.2: Danh sách các hàm và ý nghĩa tương ứng

STT	Thành phần	Hàm	Ý nghĩa	Ghi chú
1	Thành phần giao diện	Hàm nhập	Nhập yêu cầu, dữ liệu nguồn.	Cần xác định hình thức nhập/xuất và tổ chức dữ liệu tương ứng
		Hàm xuất	Xuất kết quả đã xử lý	
2	Thành phần xử lý	Hàm kiểm tra	Kiểm tra tính hợp lệ của dữ liệu.	Sử dụng hàm nhập, hàm đọc.
		Hàm xử lý	Xử lý tính toán, phát sinh, biến đổi trên dữ liệu	Sử dụng hàm nhập, hàm đọc, hàm xuất, hàm ghi
3	Thành phần dữ liệu	Hàm đọc	Đọc dữ liệu từ bộ nhớ phụ vào bộ nhớ chính.	Cần xác định cách thức tổ chức lưu trữ dữ liệu
		Hàm ghi	Ghi dữ liệu từ bộ nhớ chính vào bộ nhớ phụ	

1.1.2 Chất lượng phần mềm

Phần mềm tốt là phần mềm phải đáp ứng các chức năng theo yêu cầu, có hiệu năng tốt, có khả năng bảo trì, đáng tin cậy, và được người sử dụng chấp nhận. Cụ thể hơn, chất lượng của một phần mềm được thể hiện qua các tính chất sau đây :

❖ Tính đúng đắn

Tính chất này được thể hiện ở chỗ sản phẩm đó thực hiện *đầy đủ và chính xác* các yêu cầu của người dùng.

Theo nghĩa rộng, tính đúng đắn cần hiểu là chương trình *cần phải thực hiện được* trong cả những trường hợp mà *dữ liệu đầu vào là không hợp lệ*.

Ví dụ, nếu một trong các chức năng của phần mềm là sắp xếp tăng hoặc giảm một tập tin có số lượng mẫu tin tùy ý theo một cột bất kỳ, thì những trường hợp sau là vi phạm tính đúng đắn của chương trình khi:

- Không thể sắp xếp theo chiều tăng hay giảm dần ...
- Không thể thực hiện được (treo máy) khi tập không có mẫu tin nào.
- Không thể thực hiện, hoặc thực hiện nhưng cho kết quả sai, khi có quá nhiều mẫu tin hoặc các mẫu tin có quá nhiều cột.

Tính đúng đắn của một phần mềm được xác định dựa trên cơ sở sau đây:

- Tính đúng đắn của giải pháp xử lý hoặc thuật toán cơ sở,
- Tính đúng đắn của chương trình có thể được chứng minh trực tiếp trong tập mã lệnh hoặc nội dung của chương trình,
- Tính đúng đắn có thể được khẳng định dàn qua việc kiểm thử, việc áp dụng chương trình trong một khoảng thời gian dài trên diện rộng và với tần suất sử dụng cao (*liên quan đến hiệu năng của chương trình*),
- Tính tương đương của chương trình với thuật toán, do thuật toán có thể đúng nhưng chương trình lập ra nếu không tương đương với thuật toán, thì sẽ cho kết quả sai khi thực hiện.

❖ Tính tiền hóa

Tính chất này nhấn mạnh về *khả năng* cho phép:

- Sản phẩm có thể mở rộng, tăng cường về mặt chức năng dễ dàng,
- Người dùng khai báo thay đổi về quy định với phần mềm tùy theo thay đổi trong thế giới thực liên quan (như thay công thức tiền phạt ...)

❖ Tính hiệu quả

Tính chất này được xác định qua các tiêu chuẩn sau:

- Hiệu quả *kinh tế*, *ý nghĩa*, *giá trị* thu được do áp dụng sản phẩm đó.
- Hiệu quả *sử dụng* (tốc độ xử lý của phần mềm ...)
- Hiệu quả *kỹ thuật* (sử dụng tối ưu tài nguyên của máy tính: CPU, bộ nhớ, không gian xử lý ...)

❖ Tính tiện dụng

Tính chất này của phần mềm dựa trên những yếu tố sau đây:

- Tính cơ động và linh hoạt của sản phẩm
- Cảm nhận (về mặt tâm lý) của người dùng về:
 - Sản phẩm dễ học, có giao diện trực quan tự nhiên.
 - Các *chức năng* của sản phẩm dễ thao tác ...

❖ Tính tương thích

Tính chất này được căn cứ trên *khả năng trao đổi dữ liệu* với các *phần mềm khác* có liên quan (ví dụ: nhận danh sách nhân viên từ tập tin Excel ...), gồm Giao tiếp nội bộ và Giao tiếp bên ngoài.

❖ Tính tái sử dụng

Tính chất này được xác định khi phần mềm *có thể áp dụng* cho *nhiều lĩnh vực* theo *nhiều chế độ làm việc* khác nhau, như áp dụng về mặt kỹ thuật hay phối hợp sử dụng/liên kết, hoặc áp dụng thực hiện các phần mềm cùng lớp hay khác lớp.

1.1.3 Công nghệ Phần mềm

❖ Sự ra đời

Trong thập niên 1950, máy tính điện tử ra đời, được dùng trong các phòng thí nghiệm và bắt đầu được ứng dụng trong hoạt động xã hội. Các chương trình điều khiển (còn gọi là phần mềm) cho các máy tính cũng được chế tạo với số lượng rất ít, chủ yếu dùng cho việc tính toán đặc biệt trong lĩnh vực quốc phòng.

Đến thập niên 1960, phần mềm được chế tạo nhiều hơn và được ứng dụng rộng rãi trong nhiều lĩnh vực. Vào năm 1968, vấn đề “cuộc khủng hoảng phần mềm”¹ đã phát sinh trên thế giới, thể hiện qua 2 yếu tố chính:

- Số lượng các phần mềm tăng vọt, do sự phát triển của phần cứng làm tăng khả năng xử lý và giảm giá thành chế tạo,
- Các phần mềm được dùng đã mắc nhiều khuyết điểm như:
 - Không đáp ứng đúng yêu cầu, thiếu chính xác, không ổn định ...
 - Thời gian bảo trì, nâng cấp lâu với chi phí cao mà hiệu quả thấp.
 - Khó sử dụng và thực hiện chậm, Khó chuyển đổi dữ liệu giữa các phần mềm ...

Để giải quyết vấn đề trên, một hội nghị đã được triệu tập để bàn về cách giải quyết. Hội nghị này đã tiến hành xem xét, phân tích và xác định nguyên nhân gây ra cuộc khủng hoảng phần mềm, và đưa ra kết luận như sau:

- Việc tăng vọt các số lượng phần mềm là điều hợp lý, điều này sẽ còn tiếp diễn trong tương lai.
- Các khuyết điểm của phần mềm có nguồn gốc chủ yếu từ phương pháp và cách thức tiến hành xây dựng phần mềm, như là:
 - Cảm tính: mỗi người/mỗi bộ phận theo một cách thức riêng.

¹ Tham khảo thêm tại: http://en.wikipedia.org/wiki/Software_engineering

- Thô sơ, đơn giản: chỉ tập trung vào việc lập trình mà ít quan tâm đến các việc quan trọng cần làm khác ở giai đoạn trước khi lập trình (khảo sát hiện trạng, phân tích yêu cầu, thiết kế ...).
- Thủ công: công cụ hỗ trợ chính khi xây dựng phần mềm chỉ là trình biên dịch (*compiler*)

Từ kết luận trên, hội nghị này đã đề xuất khai sinh ngành khoa học mới là **Công nghệ Phần mềm** với *nhiệm vụ chính* là *nghiên cứu về các phương pháp tiến hành xây dựng phần mềm*.

❖ Định nghĩa

Công nghệ Phần mềm là một lĩnh vực nghiên cứu của tin học nhằm để xuất các nguyên lý, phương pháp, công cụ, cách tiếp cận phục vụ cho việc thiết kế, hiện thực các sản phẩm phần mềm đạt được đầy đủ các yêu cầu về chất lượng phần mềm.

Do quá trình tiến hóa của ngành Công nghệ Phần mềm, nên khái niệm về nó cũng thay đổi theo thời gian. Hơn nữa, đây là một lĩnh vực mới nên các khái niệm vẫn còn phụ thuộc rất nhiều vào quan điểm chủ quan của từng người khác nhau. Cụ thể như sau:

- Theo tác giả *Bauer [1969]*: việc thiết lập và sử dụng các nguyên lý công nghệ đúng đắn để thu được phần mềm một cách kinh tế vừa tin cậy vừa làm việc hiệu quả trên các máy thực.
- Theo tác giả *Ghezzi [1991]*: là một lĩnh vực của khoa học máy tính liên quan đến việc xây dựng các phần mềm vừa lớn vừa phức tạp bởi một hay một số nhóm kỹ sư.
- Theo *IEEE [1993]*: việc áp dụng phương pháp tiếp cận có hệ thống, bài bản, được lượng hóa, vận hành, bảo trì phần mềm.
- Theo tác giả *Sommerville [1995]*: là lĩnh vực liên quan đến lý thuyết, phương pháp và công cụ dùng cho phát triển phần mềm.
- Theo tác giả *Kawamura [1995]*: là lĩnh vực học vấn về các kỹ thuật, phương pháp luận công nghệ học (lý luận-kỹ thuật được hiện thực hóa trên các nguyên lý,

nguyên tắc xác định) trong toàn bộ quy trình phát triển phần mềm nhằm nâng cao chất và lượng của sản xuất phần mềm.

- Theo nhóm tác giả *Pressman [1995]*: là bộ môn tích hợp cả quy trình, các phương pháp, các công cụ để phát triển phần mềm máy tính.

Ta có thể tóm tắt về khái niệm Công nghệ Phần mềm như sau: **Công nghệ phần mềm là một nghành khoa học nghiên cứu về việc xây dựng các phần mềm có chất lượng trong khoảng thời gian và chi phí hợp lý.**

Mục tiêu nghiên cứu của ngành Công nghệ Phần mềm gồm:

- Xây dựng phần mềm *có chất lượng*.
- Xây dựng phần mềm trong thời gian và chi phí hợp lý.

❖ **Đối tượng nghiên cứu**

Hướng đến việc xây dựng phần mềm có chất lượng như nêu trên, ngành Công nghệ Phần mềm đưa ra 3 đối tượng nghiên cứu chính là:

- **Quy trình công nghệ phần mềm:** là *hệ thống các giai đoạn mà quá trình phát triển phần mềm phải trải qua*; với mỗi giai đoạn cần xác định rõ mục tiêu, kết quả nhận từ giai đoạn trước đó cũng chính là kết quả chuyển giao cho giai đoạn kế tiếp.
- **Phương pháp phát triển phần mềm:** là *hệ thống các hướng dẫn cho phép từng bước thực hiện một giai đoạn nào đó trong quy trình công nghệ phần mềm*.
- **Công cụ và môi trường phát triển phần mềm:** là *hệ thống các phần mềm trợ giúp chính trong lĩnh vực xây dựng phần mềm*. Các phần mềm này sẽ hỗ trợ các chuyên viên tin học trong các bước xây dựng phần mềm theo một phương pháp nào đó với một quy trình được chọn trước.

❖ **Các so sánh**

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

Khoa học máy tính đề cập đến lý thuyết và những vấn đề cơ bản; công nghệ phần mềm đề cập đến các hoạt động xây dựng và đưa ra một phần mềm hữu ích. Khi phần

mềm càng được phát triển mạnh thì các lý thuyết của khoa học máy tính vẫn không đủ để đóng vai trò là nền tảng hoàn thiện cho công nghệ phần mềm.

Sự khác biệt giữa công nghệ phần mềm và công nghệ hệ thống

Công nghệ hệ thống liên quan tới tất cả khía cạnh của quá trình phát triển hệ thống dựa máy tính bao gồm: phần cứng, phần mềm, và công nghệ xử lý. Kỹ sư hệ thống phải thực hiện việc đặc tả hệ thống, thiết kế kiến trúc hệ thống, tích hợp và triển khai. Công nghệ phần mềm là một phần của quy trình này, có liên quan tới việc phát triển hạ tầng phần mềm, ứng dụng và cơ sở dữ liệu trong hệ thống.

1.2 QUY TRÌNH CÔNG NGHỆ PHẦN MỀM

Quá trình xây dựng được phần mềm có chất lượng cần trải qua *nhiều giai đoạn*. Mỗi giai đoạn có *mục tiêu* và *kết quả chuyển giao* xác định. Trình tự thực hiện các giai đoạn này chính là *chu kỳ sống của một phần mềm*.

Tổng quát, **chu kỳ sống của một phần mềm** là *khoảng thời gian* mà trong đó một sản phẩm phần mềm được *phát triển, sử dụng và mở rộng* cho đến khi sản phẩm phần mềm đó *không còn được sử dụng* nữa.

Chu kỳ sống của một phần mềm được phân chia thành các pha chính như: *Xác định, Phát triển, Kiểm thử, Bảo trì / Vận hành*. Phạm vi và thứ tự các pha khác nhau tùy theo từng mô hình cụ thể.

Các bước cơ bản trong xây dựng phần mềm:

1.2.1 Bước Xác định

Đây là bước *hình thành dự án*, khi đó ta (*nhóm phân tích thiết kế*) cần:

- Biết được vai trò của phần mềm cần phát triển trong hệ thống, đồng thời phải ước lượng công việc, lập lịch biểu và phân chia công việc.
- Biết được yêu cầu của khách hàng. Các yêu cầu đó cần phải được thu thập đầy đủ, được phân tích theo diện rộng và sâu.

Công cụ sử dụng chủ yếu ở giai đoạn này là các *lược đồ, sơ đồ phản ánh rõ các thành phần* của hệ thống và *mối liên quan giữa chúng*.

1.2.2 Bước Phát triển

Trong bước này, dựa vào kết quả của bước trên, ta (*nhóm phát triển phần mềm*) thực hiện công đoạn **Thiết kế**, trong đó ta dùng *ngôn ngữ đặc tả hình thức* (dựa trên các kiến trúc toán học) hoặc *phi hình thức* (tựa ngôn ngữ tự nhiên) hoặc *kết hợp cả hai* để mô tả những yếu tố sau đây của chương trình:

- Giá trị nhập, giá trị xuất; Các phép biến đổi,
- Các yêu cầu cần đạt được ở mỗi điểm của chương trình.

Phần đặc tả chỉ quan tâm chủ yếu đến *giá trị vào/ra* chứ không quan tâm đến cấu trúc và nội dung các thao tác cần thực hiện.

Sau đó là công đoạn **Xây dựng** để chuyển các đặc tả chương trình thành sản phẩm phần mềm dựa trên ngôn ngữ lập trình cụ thể. Trong công đoạn này ta (*nhóm lập trình viên*) tiến hành hiện thực (hay là thi công, cài đặt, lập trình hoặc phát triển) các thao tác cần thiết để thực hiện đúng các yêu cầu đã được đặc tả.

Công đoạn cuối cùng **Kiểm thử**, trong đó ta (*nhóm kiểm tra chất lượng*) cần chứng minh tính đúng đắn của chương trình sau khi đã tiến hành hiện thực. Thông thường, ở công đoạn này ta xem các chương trình như những *hộp đen*. Vấn đề đặt ra là ta cần xây dựng một cách có chủ đích các tập dữ liệu thử nghiệm khác nhau để nhập cho chương trình thực hiện rồi dựa vào kết quả thu được để đánh giá chương trình. Công việc như trên được gọi là *kiểm thử chương trình*. Công việc kiểm thử nhằm vào các mục tiêu sau:

- Kiểm tra để *phát hiện lỗi* của chương trình. Lưu ý rằng kiểm thử không đảm bảo tuyệt đối tính đúng đắn của chương trình do bản chất quy nạp không hoàn toàn của cách làm.
- Kiểm tra để xác định tính *ổn định, hiệu quả* cũng như *hiệu năng tối đa* của chương trình.

Tùy theo mục đích mà người ta thiết kế các tập dữ liệu thử nghiệm sao cho có thể bao phủ hết các trường hợp cần quan tâm.

1.2.3 Bước Bảo trì (Vận hành)

Công tác quản lý việc triển khai và sử dụng phần mềm là vấn đề cần được quan tâm trong quy trình phát triển phần mềm. Trong quá trình xây dựng phần mềm, tất cả kết quả phân tích, thiết kế, hiện thực và hồ sơ liên quan cần phải *được lưu trữ và quản lý cẩn thận*, nhằm đảm bảo cho công việc được tiến hành một cách hiệu quả nhất và phục vụ cho công việc bảo trì phần mềm về sau. Công tác quản lý không chỉ dừng lại trong quá trình xây dựng phần mềm mà còn phải được tiến hành liên tục trong suốt quá trình sống của nó.

1.3 MỘT SỐ MÔ HÌNH TRIỂN KHAI

Có nhiều dạng mô hình khác nhau để triển khai các bước cơ bản trong quá trình phát triển phần mềm. Mỗi mô hình sẽ chia vòng đời của phần mềm theo cách khác nhau, để đảm bảo quy trình phát triển phần mềm sẽ thành công.

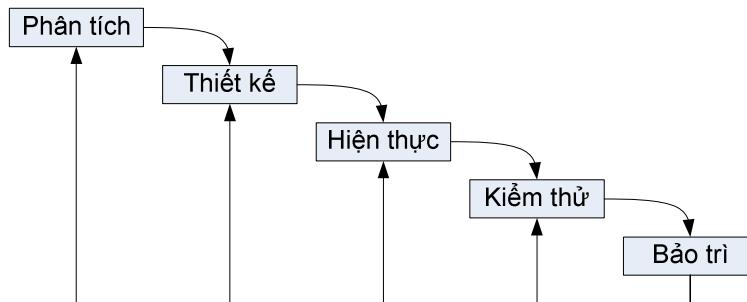
1.3.1 Mô hình Thác nước

Đây là một trong các mô hình đầu tiên và phổ biến, được áp dụng trong quá trình phát triển phần mềm. Mô hình này chia quá trình phát triển phần mềm thành những giai đoạn tuần tự nối tiếp. Mỗi giai đoạn có một mục đích nhất định. Kết quả của giai đoạn trước là thông tin đầu vào cho giai đoạn tiếp theo sau. Dạng phức tạp nhất của mô hình thác nước có 5 giai đoạn gồm:

- *Xác định yêu cầu*: Được tiến hành khi có nhu cầu xây dựng phần mềm.
 - Mục tiêu: xác định chính xác các yêu cầu cho phần mềm cần có.
 - Kết quả nhận: thông tin về hoạt động của thế giới thực.
 - Kết quả chuyển giao: danh sách các yêu cầu (công việc sẽ thực hiện trên máy tính) cùng với các thông tin miêu tả chi tiết về các yêu cầu (cách thức thực hiện trong thế giới thực)
- *Phân tích*: được tiến hành ngay sau khi kết thúc việc xác định yêu cầu.
 - Mục tiêu: mô tả lại thế giới thực bằng mô hình trước khi thiết kế.
 - Kết quả nhận: danh sách các yêu cầu cùng thông tin liên quan.

- Kết quả chuyển giao:
 - Mô hình xử lý (hệ thống các công việc trong thế giới thực cùng với quan hệ giữa chúng),
 - Mô hình dữ liệu (hệ thống các loại thông tin được sử dụng trong thế giới thực cùng với quan hệ giữa chúng),
 - Mô hình khác (không gian, thời gian, con người...) nếu có.
- *Thiết kế*: được tiến hành ngay sau khi kết thúc việc phân tích.
 - Mục tiêu: mô tả các thành phần của phần mềm (mô hình của phần mềm) trước khi tiến hành cài đặt.
 - Kết quả nhận: mô hình thế giới thực.
 - Kết quả chuyển giao:
 - Mô tả thành phần giao diện: các hàm/cấu trúc dữ liệu nhập/xuất.
 - Mô tả thành phần xử lý: các hàm kiểm tra xử lý.
 - Mô tả thành phần dữ liệu: các hàm đọc/ghi, tổ chức lưu trữ trên bộ nhớ phụ.
- *Hiện thực*: được tiến hành ngay sau khi kết thúc việc thiết kế.
 - Mục tiêu: tạo lập phần mềm theo yêu cầu.
 - Kết quả nhận: mô hình phần mềm.
 - Kết quả chuyển giao: chương trình nguồn của phần mềm với cấu trúc dữ liệu tương ứng và chương trình thực hiện trên máy tính.
- *Kiểm thử*: được tiến hành ngay khi có kết quả của việc lập trình.
 - Mục tiêu: tăng độ tin cậy của phần mềm.
 - Kết quả nhận: các yêu cầu, mô hình phần mềm, phần mềm.
 - Kết quả chuyển giao: phần mềm có độ tin cậy cao (đã sửa lỗi).
- *Bảo trì*: Công việc của giai đoạn bao gồm việc cài đặt và vận hành phần mềm trong thực tế.
 - Mục tiêu: đảm bảo phần mềm vận hành tốt

- Kết quả nhận: phần mềm đã hoàn thành
- Kết quả chuyển giao: các phản ánh của khách hàng trong quá trình sử dụng phần mềm.



Hình 1.1: Mô hình thác nước

❖ Nhận xét

Mô hình Thác nước giúp ta có thể dễ dàng phân chia quá trình xây dựng phần mềm thành những giai đoạn hoàn toàn độc lập nhau.

Một số hạn chế của mô hình này:

- Các dự án lớn ít khi tuân theo dòng chảy tuần tự của mô hình, vì chúng thường phải lặp lại các bước để nâng cao chất lượng, hơn nữa khách hàng ít khi tuyên bố hết các yêu cầu trong giai đoạn phân tích.
- Ta rất khó thực hiện các thay đổi khi đã thực hiện xong một giai đoạn nào đó, điều này làm cho việc xây dựng phần mềm rất khó thay đổi các yêu cầu theo ý muốn của khách hàng. Do đó, phương pháp này chỉ thích hợp cho những trường hợp mà ta đã hiểu rõ các yêu cầu của khách hàng.
- Mô hình này chỉ thích hợp khi các yêu cầu đã được tìm hiểu rõ ràng, những thay đổi sẽ được giới hạn rõ ràng trong suốt quá trình thiết kế.

❖ Chú ý

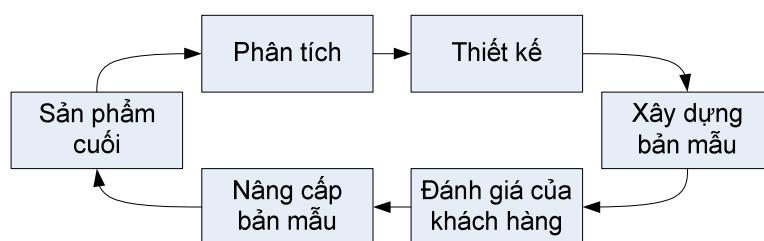
Mô hình thác nước có thể được cải tiến bằng cách cho phép quay lui khi phát hiện lỗi trong giai đoạn phía trước.

1.3.2 Mô hình Bản mẫu Phần mềm

Mô hình Bản mẫu tương tự như mô hình thác nước với việc bổ sung vào *giai đoạn thực hiện phần mềm mẫu* (ngay khi xác định yêu cầu nhằm mục tiêu phát hiện nhanh các sai sót về yêu cầu). Các giai đoạn trong mô hình bản mẫu có thể tiến hành lặp lại mà không nhất thiết theo trình tự nhất định.

Ngay sau khi giai đoạn Xác định yêu cầu, ta (*nhóm phát triển phần mềm*) đưa ra một bản thiết kế sơ bộ và tiến hành hiện thực bản mẫu đầu tiên, rồi chuyển chúng cho người sử dụng. Bản mẫu này chỉ nhằm để mô tả cách thức phần mềm hoạt động cũng như cách người dùng tương tác với hệ thống.

Người dùng sau khi xem xét sẽ phản hồi thông tin cần thiết lại cho nhóm phát triển. Nếu người dùng đồng ý với bản mẫu đã đưa, nhóm phát triển sẽ tiến hành hiện thực thật sự. Ngược lại, cả hai phải quay lại giai đoạn xác định yêu cầu. Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

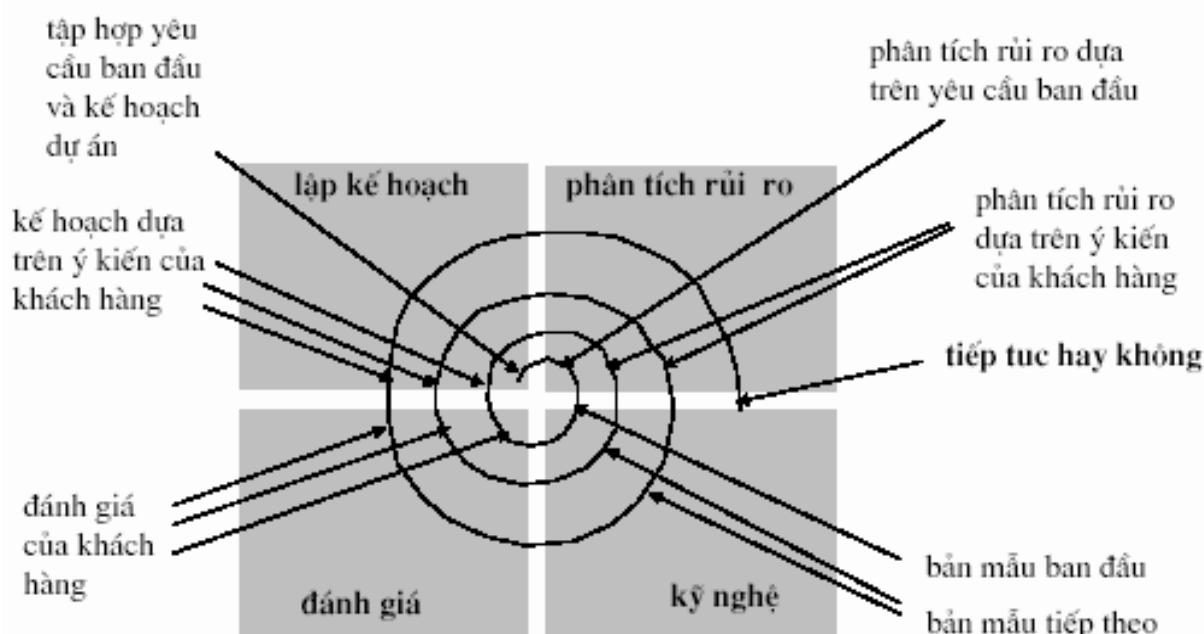


Hình 1.2: Mô hình bản mẫu

Như vậy đây là một hướng tiếp cận tốt khi các yêu cầu chưa rõ ràng và khó đánh giá được tính hiệu quả của các thuật toán. Tuy nhiên, mô hình này cũng có nhược điểm là tính cấu trúc không cao do đó khách hàng dễ mất tin tưởng, và thiếu tầm nhìn của cả quy trình; các hệ thống thường hướng cấu trúc nghèo nàn và yêu cầu các kỹ năng đặc biệt. Mô hình này chỉ nên áp dụng với những hệ thống có tương tác ở mức độ nhỏ hoặc vừa; trên một phần của những hệ thống lớn; hoặc những hệ thống có thời gian chu kỳ tồn tại ngắn.

1.3.3 Mô hình Xoắn ốc

Mô hình Xoắn ốc chính là *sự kết hợp của mô hình bản mẫu thiết kế và mô hình thác nước* được *lặp lại nhiều lần*. Ở lần lặp tiếp theo, hệ thống sẽ được tìm hiểu và xây dựng hoàn thiện hơn ở lần lặp trước đó, tức là yêu cầu của người dùng sẽ được hiểu ngày càng rõ ràng hơn, và các bản mẫu phần mềm cũng ngày một hoàn thiện hơn. Ngoài ra, ở cuối mỗi lần lặp sẽ có thêm công đoạn Phân tích mức độ rủi ro để quyết định xem có nên đi tiếp theo hướng này nữa hay không. Mô hình này phù hợp với các hệ thống phần mềm lớn do có khả năng kiểm soát rủi ro ở từng bước tiến hóa.



Mô hình xoắn ốc

Hình 1.3: Mô hình xoắn ốc

1.4 CÁC PHƯƠNG PHÁP XÂY DỰNG PHẦN MỀM

1.4.1 Khái niệm

Để tiến hành xây dựng một phần mềm, ta có thể áp dụng nhiều phương pháp khác nhau. Mỗi phương pháp sẽ có:

- Ưu khuyết điểm riêng, phù hợp riêng từng loại phần mềm cụ thể,

- Có các hướng dẫn cụ thể những công việc cần thực hiện trong từng giai đoạn trong quy trình xây dựng phần mềm,
- Quy định những cách thức khác nhau để trình bày các kết quả thu được trong quá trình xây dựng phần mềm; những quy định này có tính chất như là ngôn ngữ thống nhất để các thành viên tham gia xây dựng phần mềm có thể trao đổi thông tin trong việc xây dựng phần mềm.

1.4.2 Phân loại

Các phương pháp xây dựng phần mềm được chia làm 2 nhóm khác nhau dựa vào tính chất của công việc cần thực hiện, gồm có:

- Phương pháp xây dựng: Phương pháp hướng chức năng; hướng dữ liệu; hướng đối tượng
- Phương pháp tổ chức quản lý: Xây dựng phương án, Tổ chức nhân sự, Ước lượng rủi ro-chi phí, Lập và theo dõi kế hoạch triển khai.

1.4.3 Các phương pháp xây dựng phần mềm

❖ Cách tiếp cận

Tiếp cận từ trên xuống (top-down)

Đây là cách giải quyết vấn đề theo *hướng phân tích*. Khi tiến hành xây dựng phần mềm theo cách này, ta bắt đầu với những *thành phần chính* của hệ thống. Sau đó, các thành phần này sẽ được *phân tích thành các thành phần chi tiết và cụ thể hơn*. Quá trình phân tích sẽ kết thúc khi kết quả thu được có mức độ phức tạp đúng với ý muốn của nhóm phát triển phần mềm.

Tiếp cận từ dưới lên (bottom-up)

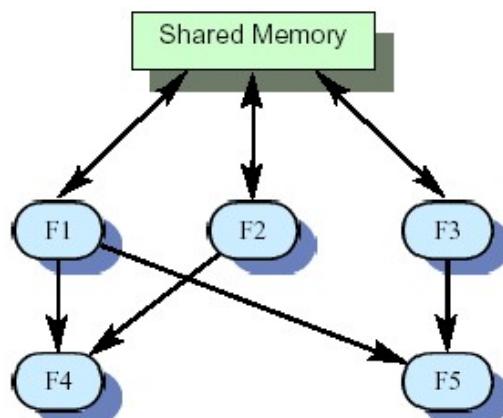
Đây là cách giải quyết vấn đề theo *hướng tổng hợp* (ngược lại với phương pháp từ trên xuống). Với phương pháp này, ta tiến hành xây dựng những *thành phần chi tiết* mà ta dự tính là sẽ có trong hệ thống. Sau đó, nhóm phát triển phần mềm sẽ tiến hành *kết hợp các thành phần chi tiết* này lại với nhau để tạo nên các *thành phần chính* mà hệ thống cần phải có.

❖ **Cách tiến hành**

Phương pháp Hướng Chức năng

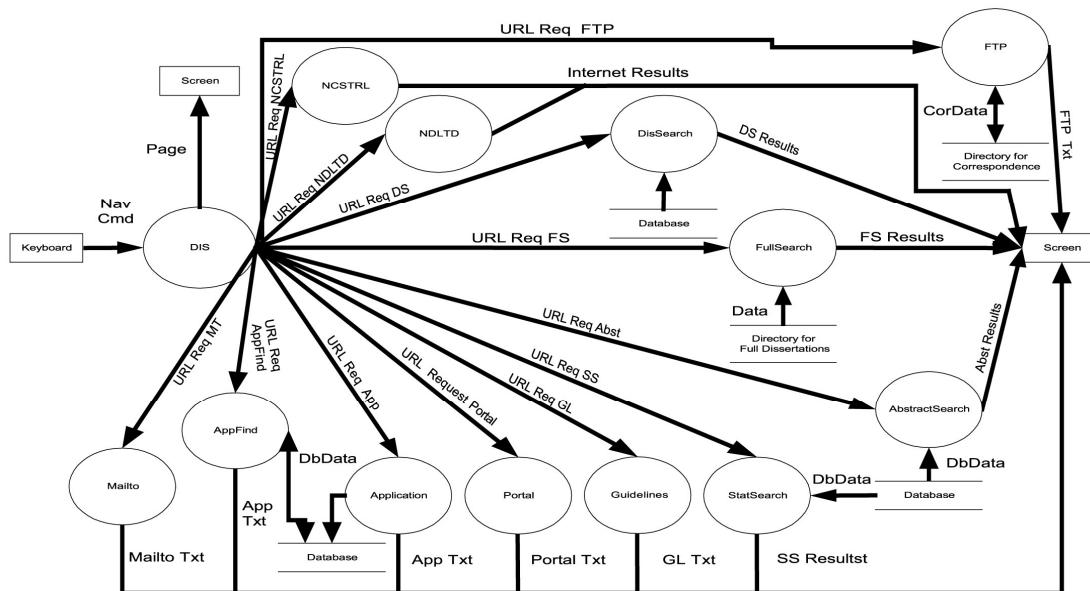
Với phương pháp này, việc xây dựng phần mềm được thực hiện *dựa trên các chức năng mà hệ thống cần thực hiện, chú trọng đến thành phần xử lý với các thao tác tính toán, thao tác phát sinh, thao tác biến đổi ...*

Phương pháp chung để giải quyết vấn đề là áp dụng nguyên lý “*chia để trị*”. Khi tiến hành xây dựng phần mềm theo phương pháp này, ta sẽ chia các công việc lớn (mà hệ thống cần thực hiện) thành các công việc nhỏ hơn và độc lập nhau. Việc phân chia các công việc được tiến hành cho đến khi các công việc thu được đủ nhỏ để ta có thể tiến hành xây dựng hoàn chỉnh (h1.4).



Hình 1.4: Minh họa Nguyên lý Chia để trị

Phương pháp Hướng chức năng *chú trọng đến cách giải quyết vấn đề nhưng không có khả năng che dấu các thông tin trạng thái của hệ thống*. Điều này dẫn đến việc các chức năng trong hệ thống sẽ không tương thích với nhau trong khi thực hiện thay đổi các thông tin. Vì vậy cách tiếp cận này chỉ *thích hợp* khi trong *hệ thống có rất ít thông tin cần phải quản lý* và *chia sẻ* giữa các chức năng với nhau. Để mô hình hóa cách xử lý thông tin, ta dùng **lược đồ dòng dữ liệu** (**DFD**, *Data Flow Diagrams*), đây là công cụ đơn giản và hữu ích để mô tả cách thức hoạt động của hệ thống (*xem thêm tài liệu về Phân tích Thiết kế Hệ thống Thông tin*)



Hình 1.5: Ví dụ minh họa về DFD (nguồn [5])

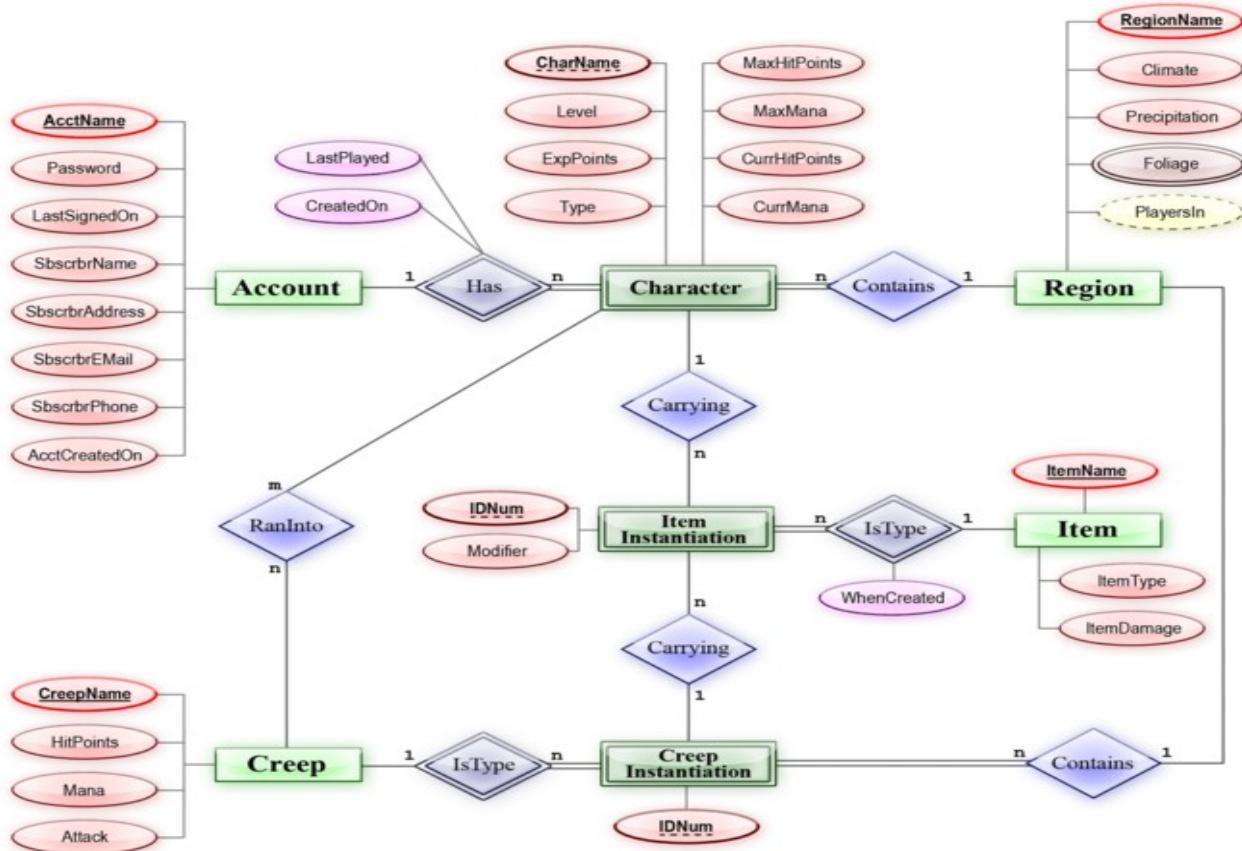
Phương pháp Hướng Dữ liệu

Ngược lại với phương pháp trên, phương pháp Hướng Dữ liệu *chú trọng nhiều đến thành phần dữ liệu* cần phải xử lý trong hệ thống là tổ chức dữ liệu, khối lượng lưu trữ, tốc độ truy xuất ...

Khi tiến hành thiết kế theo phương pháp này, ta *bắt đầu* với việc *thiết kế các cấu trúc dữ liệu* cần thiết có trong bài toán, sau đó mới tiến hành thiết kế các thao tác để vận hành trên cấu trúc dữ liệu đã thiết kế.

Phương pháp này chỉ *thích hợp* cho loại phần mềm có chức năng chính là *lưu trữ và thao tác* trên các loại *dữ liệu*. *Hạn chế* của nó là *không quan tâm* đến các *chức năng* mà hệ thống cần phải đáp ứng. Điều này dẫn đến việc có khả năng hệ thống sau khi thiết kế không có đầy đủ các chức năng cần thiết.

Kết quả thu được sau khi thiết kế theo phương pháp hướng dữ liệu là **mô hình thực thể kết hợp** (*Entity Relationship Diagram, ERD*) (xem thêm tài liệu về *Phân tích Thiết kế Hệ thống Thông tin*).



Hình 1.6: Minh họa về ERD (nguồn [5])

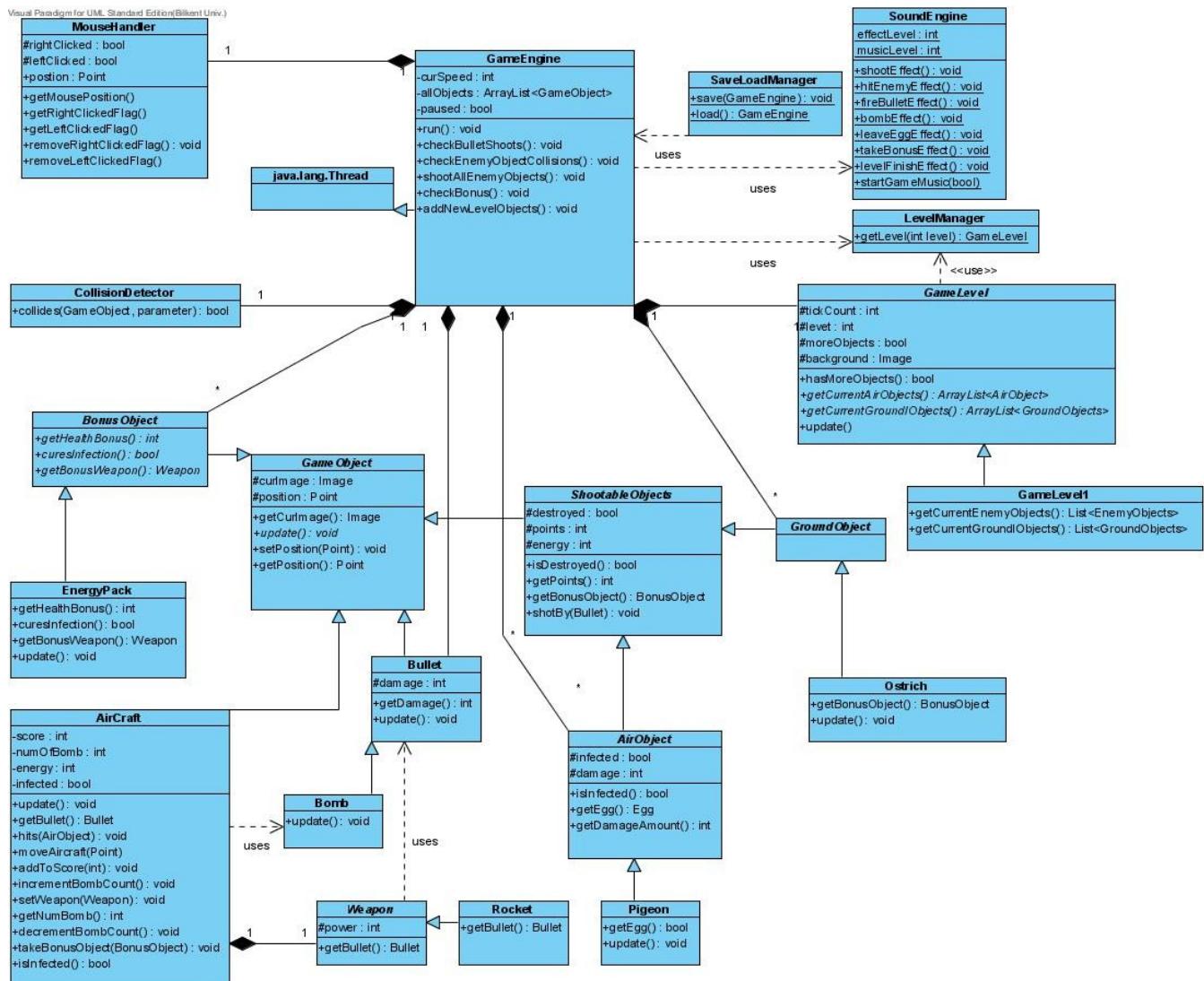
Phương pháp Hướng Đối tượng

Phương pháp thiết kế Hướng Đối tượng là sự kết hợp của phương pháp *Hướng Dữ liệu* và phương pháp *Hướng Chức năng*. Phương pháp này chú trọng đến cả thành phần dữ liệu và chức năng của hệ thống.

Theo phương pháp hướng đối tượng, một hệ thống phần mềm là tập hợp các *đối tượng* có khả năng tương tác với nhau. Các đối tượng chính là những sự vật và hiện tượng vật lý cũng như trừu tượng có trong thế giới thực. Mỗi đối tượng có dữ liệu riêng được che dấu với thế giới bên ngoài và các thao tác mà đối tượng có thể thực hiện trên các thành phần dữ liệu của đối tượng.

Các đối tượng liên lạc, trao đổi thông tin với nhau bằng cách gửi các thông điệp cho nhau. Các thông điệp mà mỗi đối tượng có thể xử lý được gọi là giao diện của đối tượng. Khi đó mọi thao tác liên quan đến các đối tượng được phải thực hiện thông qua giao diện của đối tượng. Điều này giúp ta đảm bảo rằng các thông tin bên trong các

đối tượng được bảo vệ một cách chắc chắn. (xem thêm tài liệu về Phân tích Thiết kế Hướng Đối tượng).



Hình 1.7: Minh họa sơ đồ lớp đối tượng (nguồn [5])

1.5 CÔNG CỤ & MÔI TRƯỜNG PHÁT TRIỂN PHẦN MỀM

1.5.1 Mở đầu

Các công cụ và môi trường phát triển phần mềm là các phần mềm hỗ trợ chính người phát triển trong quá trình xây dựng phần mềm. Các phần mềm này có tên gọi chung là công cụ **CASE** (*Computer Aided Software Engineering tool*). Chúng có thể hỗ trợ cụ thể cho một giai đoạn nào đó hay cũng có thể hỗ trợ một số giai đoạn, trong

trường hợp này tên gọi chung thường là **môi trường phát triển phần mềm (SDE, Software Development Environment)**, gồm 2 hình thức chính:

- Cho lưu / cập nhật trên kết quả chuyển giao với phương pháp nào đó.
- Cho phát sinh ra kết quả chuyển giao cho giao đoạn kế tiếp.

1.5.2 Phần mềm hỗ trợ thực hiện các giai đoạn

1.5.2.1 Phần mềm hỗ trợ phân tích

- Công việc hỗ trợ chính: Soạn thảo các mô hình thế giới thực, Ánh xạ vào mô hình luận lý
- Các phần mềm: WinA&D, Analyst Pro, Rational Rose ...

1.5.2.2 Phần mềm hỗ trợ thiết kế

- Công việc hỗ trợ chính: Soạn thảo các mô hình luận lý, Ánh xạ vào mô hình vật lý
- Các phần mềm: Power Designer, Oracle Designer, Rational Rose ...

1.5.2.3 Phần mềm hỗ trợ lập trình

- Công việc hỗ trợ chính: Quản lý các phiên bản (dữ liệu, chương trình nguồn, giao diện), biên dịch
- Các phần mềm: Visual Studio Net (Basic, C#, C++), Borland ...

1.5.2.4 Phần mềm hỗ trợ kiểm chứng

- Công việc hỗ trợ chính: Phát sinh tự động các bộ dữ liệu thử nghiệm, Phát hiện lỗi
- Các phần mềm: WinRuner, QuickTestPro ...

1.5.3 Phần mềm hỗ trợ tổ chức, quản lý việc triển khai

1.5.3.1 Xây dựng phương án

- Công việc hỗ trợ chính: tạo lập phương án, dự đoán rủi ro, tính chi phí
- Các phần mềm: MS Project, Visio, Rational Rose ...

1.5.3.2 Lập kế hoạch

- Công việc hỗ trợ chính: xác định các công việc, phân công, lập lịch biểu, theo dõi thực hiện
- Các phần mềm: MS Project, Visio

1.6 YÊU CẦU ĐỐI VỚI KỸ SƯ PHẦN MỀM

Quy trình xây dựng phần mềm được thực hiện trong một môi trường chuyên nghiệp và đòi hỏi tuân thủ các nguyên tắc một cách chính xác.

Những kỹ sư phần mềm phải coi công việc của họ là trách nhiệm to lớn, chứ không đơn thuần chỉ là việc ứng dụng kỹ thuật.

Kỹ sư phần mềm phải ứng xử trung thực và cách làm của họ phải rất chuyên nghiệp và đúng quy tắc.

Một số nguyên tắc cần thiết mà một kỹ sư phần mềm phải thực hiện.

- *Sự tin cẩn*: kỹ sư phần mềm phải tạo được sự tin cẩn từ phía nhân viên và khách hàng.
- *Năng lực*: kỹ sư phần mềm không nên trình bày sai khả năng của mình, không nên nhận những công việc vượt quá khả năng của mình.
- *Các quyền về tài sản trí tuệ*: kỹ sư phần mềm nên quan tâm về các tài sản trí tuệ được bảo hộ: bằng sáng chế, quyền tác giả ... để đảm bảo tất cả tài sản trí tuệ đều được bảo hộ.
- *Lạm dụng máy tính*: kỹ sư phần mềm không nên sử dụng các kỹ năng của mình để gây ảnh hưởng tới người khác. Lạm dụng máy tính có thể được hiểu là những việc tăm thường (như chơi điện tử trên máy tính của người khác) đến những vấn đề nghiêm trọng (như phát tán virus).

Vấn đề về tính chuyên nghiệp và đúng quy tắc đối với kỹ sư phần mềm quan trọng tới mức một số tổ chức ở Mỹ đã hợp tác để phát triển bản Code of Ethics gồm 8 quy tắc liên quan đến ứng xử và cách ra quyết định của các kỹ sư phần mềm chuyên nghiệp.

TÓM TẮT

Bài học này trình bày những khái niệm về Công nghệ Phần mềm như:

- Các khái niệm cơ bản
 - Phần mềm (khái niệm, phân loại, kiến trúc phần mềm)
 - Chất lượng phần mềm (tính đúng đắn, tính tiến hóa, tính hiệu quả, tính tiện dụng, tính tương thích, tính tái sử dụng)
 - Công nghệ phần mềm (nguồn gốc, định nghĩa ...)
- Quy trình công nghệ phần mềm
 - Các bước cơ bản trong xây dựng phần mềm (Xác định, Phát triển, Bảo trì /Vận hành)
 - Một số mô hình triển khai xây dựng phần mềm (Thác nước, Bản mẫu Phần mềm, Xoắn ốc)
- Các phương pháp xây dựng phần mềm: tổng quan (Khái niệm, Phân loại), phương pháp xây dựng phần mềm (Cách tiếp cận, tiến hành)
- Công cụ & môi trường phát triển phần mềm
 - Phần mềm hỗ trợ thực hiện các giai đoạn (phân tích, thiết kế, lập trình, kiểm chứng)
 - Phần mềm hỗ trợ tổ chức, quản lý việc triển khai (Xây dựng phương án, Lập kế hoạch)
- Yêu cầu đối với kỹ sư phần mềm với các tiêu chí về sự tin cẩn, năng lực, quyển tài sản trí tuệ.

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 2: TÁC VỤ PHÂN TÍCH & ĐẶC TẢ YÊU CẦU

Học xong bài này người học sẽ:

- *Hiểu được các khái niệm cơ bản về mô hình hóa yêu cầu hệ thống.*
- *Biết được các dạng thức và quy trình phân tích yêu cầu, từ đó có thể thực hiện mô hình hóa phần mềm dựa trên yêu cầu ban đầu.*

2.1 TỔNG QUAN

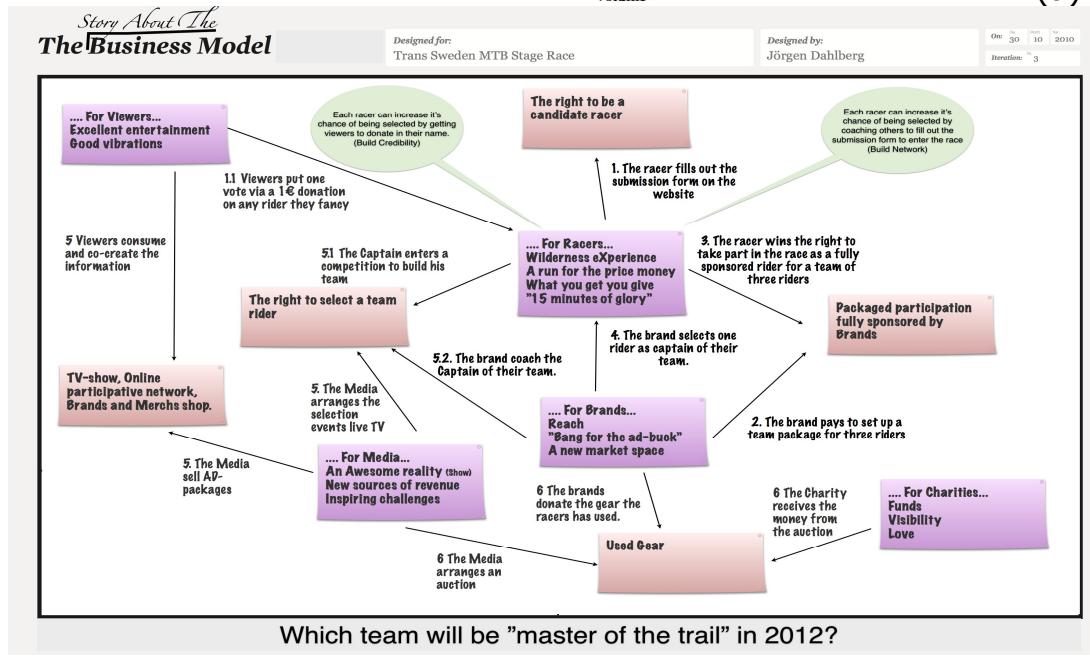
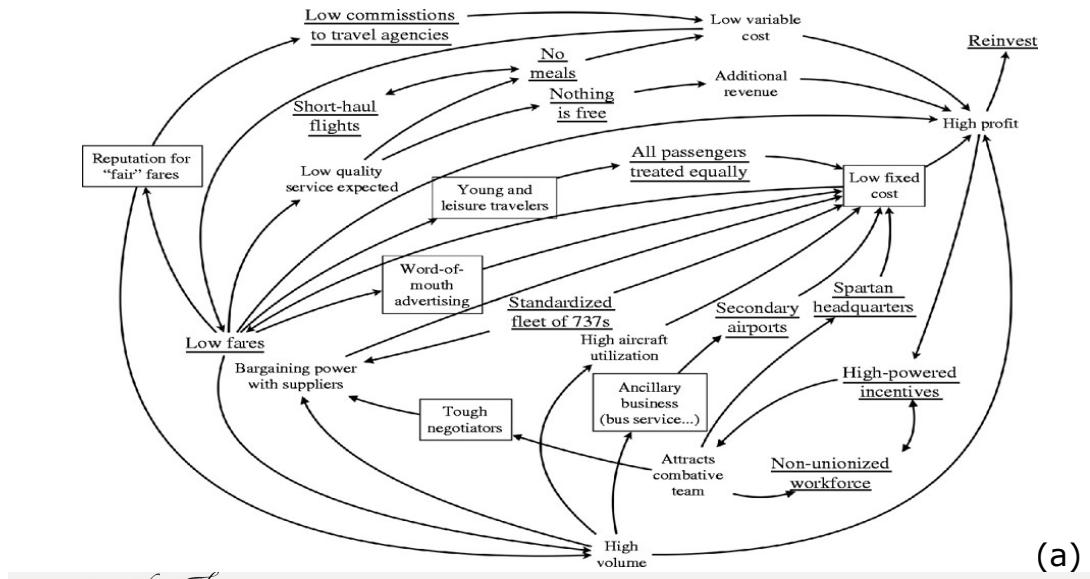
Tác vụ Phân tích Yêu cầu gồm các công đoạn nhỏ là nghiên cứu khả thi, phân tích mô hình hóa, đặc tả thẩm định yêu cầu. Bước này được phối hợp tiến hành giữa nhóm phát triển phần mềm và khách hàng, đồng thời đóng vai trò đặc biệt quan trọng trong tiến trình phát triển phần mềm. Đây là bước hình thành dự án phần mềm, trong đó trưởng nhóm thiết kế và người phân tích hệ thống phải biết được nhu cầu của người đặt hàng. Các yêu cầu cần được thu thập đầy đủ và được phân tích đủ rộng và sâu. Công cụ sử dụng chủ yếu ở giai đoạn này là các sơ đồ phản ánh rõ các đối tượng của hệ thống: lưu đồ, sơ đồ dòng dữ liệu, mạng, thực thể-quan hệ, sơ đồ cấu trúc phân cấp, mạng ngữ nghĩa. (Xem thêm Phụ lục C – Phần B)

2.2 QUÁ TRÌNH PHÂN TÍCH

2.2.1 Phân tích Phạm vi dự án

Nhóm phân tích hệ thống thường dùng thuật ngữ *phạm vi* để chỉ trách nhiệm *dự án phải thực thi*. Ta xác định phạm vi dự án bằng cách định rõ quá trình nghiệp vụ mà phần mềm sẽ xử lý. Một giải pháp nghiệp vụ luôn có:

- Phần *triển khai* phần mềm: trong đó yêu cầu nghiệp vụ của khách hàng được hiện thực hóa thành phần mềm cụ thể,
 - Phần *thực hiện* bởi con người hay chương trình: là giai đoạn vận hành sử dụng hệ thống.



Hình 2.1: Minh họa phân tích phạm vi dự án (nguồn [5])

Việc định ra ranh giới giữa hai phần này chính là công tác xác định quy trình trách nhiệm. Khi đã định nghĩa được trách nhiệm của dự án thì ta cũng:

- Chia trách nhiệm thành những nhiệm vụ con để đưa ra ý tưởng cho chính mình về bao nhiêu mô-đun chương trình khác nhau yêu cầu,

- Xác định bao nhiêu vùng địa lý liên quan (chi nhánh văn phòng),
- Ước lượng số người dùng phần mềm, thời gian phần mềm được duy trì,
- Biết được tính chính xác của yêu cầu phần mềm,
- Hiểu khách hàng mong đợi dự án được triển khai.

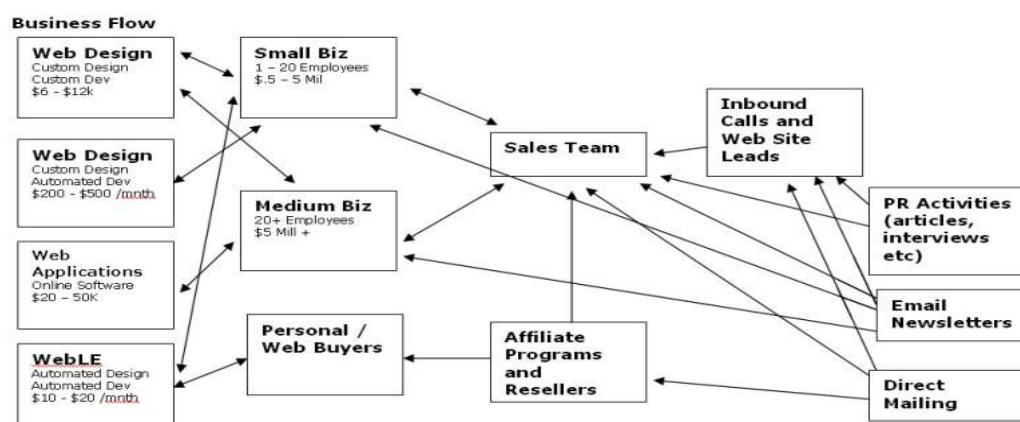
Tại thời điểm đó, ta sẽ có ý tưởng về phạm vi dự án.

Ta cần cân nhắc *độ lớn* của dự án đối với *thời gian* và ràng buộc *ngân sách* để đảm bảo tính khả thi của việc triển khai phần mềm. Nếu dự án quá lớn về thời gian và chi phí, ta cần trao đổi với khách hàng để thương (thời gian hơn, chi phí, phạm vi dự ...). Nếu phân tích được tất cả tình huống ở giai đoạn đầu của dự án, ta có thể đảm bảo nhiều hơn cho sự thành công của dự án.

2.2.2 Phân tích Mở rộng Yêu cầu Nghiệp vụ

❖ Xác định yêu cầu nghiệp vụ

Mỗi dự án sẽ có một hay nhiều yêu cầu nghiệp vụ, gọi là *tác vụ*, liên quan đến việc mô tả *công việc cụ thể* trong nghiệp vụ của khách hàng bằng ngôn ngữ phù hợp để giúp người dùng có thể *kiểm tra các ràng buộc nghiệp vụ* một cách chính xác. Một tác vụ cần được chia nhỏ thành nhiều phần cho đến khi mỗi phần đủ để mô tả được công việc chính xác. Khi mức độ của thành phần chia nhỏ dưới mức tối thiểu, chúng sẽ được xác định lại trình tự thành phần.



Hình 2.2: Minh họa luồng nghiệp vụ (nguồn [5])

❖ Xác định yêu cầu chất lượng

Mỗi dự án phần mềm có thể có các yêu cầu liên quan đến khả năng đáp ứng nhanh, bảo mật, phụ thuộc, dễ dùng ... Thực tế, ràng buộc thời gian và tài chính làm cho chương trình dự án khó có thể diễn ra hoàn tất trọn vẹn. Thay vào đó, điều quan trọng để quyết định sự hoàn tất trọn vẹn là dựa trên mức độ chấp nhận của chất lượng thỏa mãn nhu cầu của khách hàng.

❖ Phân tích cơ sở hạ tầng hiện hành

Giải pháp phần mềm được đưa vào - nếu phù hợp với cơ sở hạ tầng - sẽ phù hợp hơn là thay thế hệ thống hiện hành (trừ khi khách hàng có yêu cầu thay thế, hoặc hệ thống hiện hành đã quá lạc hậu với nhiều lỗi kỹ thuật hay hạn chế lớn). Khi đưa giải pháp, ta cần nhớ là việc *tương thích với cơ sở hạ tầng hiện hành sẽ đảm bảo khả thi cho giải pháp* của ta, vì dự án cần làm việc trên phần cứng và phần mềm mà người dùng hiện có. Nếu ta biết được hệ điều hành đang được cài trên máy của người dùng, loại mạng đang sử dụng, và những phần mềm không tương thích với chương trình mới hơn, hoặc biết được thông tin cấu hình của máy chủ hay hệ điều hành, phần mềm đang chạy trên đó, sẽ giúp việc phân tích sẽ chính xác và hiệu quả hơn.

❖ Phân tích ảnh hưởng kỹ thuật

Nếu cần mở rộng chức năng cho hệ thống hiện hành, thường ta mong muốn thay đổi cả hệ thống cũ, nhưng ta nên lưu ý đến việc cải thiện hệ thống cũ và tích hợp dễ dàng hơn hệ thống mới.

Ví dụ: chức năng của chương trình kế toán lưu trữ dữ liệu nhỏ trên cơ sở dữ liệu MS Access. Để tạo dữ liệu truy xuất hiệu quả hơn và thỏa mãn yêu cầu của giải pháp mới, ta mới chuyển toàn bộ dữ liệu sang hệ quản trị dữ liệu SQL Server. Việc định hướng trước như vậy sẽ giúp ta tiết kiệm thời gian sau đó như: trải qua thời gian tìm hiểu sự khác biệt về giao tác, bảo mật, và những chức năng khác giữa kỹ thuật cũ và giải pháp mới.

Ta nên tìm hiểu thủ tục chuyển đổi dữ liệu từ kỹ thuật cũ sang kỹ thuật mới, cần đảm bảo thực nghiệm được các thủ tục này, có kế hoạch bảo lưu khi việc thực hiện này bị lỗi. Ta cần đảm bảo những tác động sẽ chuyển đổi trên mọi thành phần của hệ thống mà không chỉ thay đổi phần tử gần nhất.

2.2.3 Phân tích Yêu cầu Bảo mật

Khi hệ thống cung cấp cho người dùng khả năng lưu trữ, truy xuất dữ liệu cá nhân (như thông tin nhân sự, thẻ tín dụng, doanh số bán ...) hay thông tin riêng tư, ta cần có biện pháp đảm bảo an toàn những dữ liệu này.

❖ Xác định các vai trò người dùng trong phần mềm

Toàn bộ phần mềm không chỉ có một mức độ bảo mật, ví dụ như:

- Người dùng cuối (*end-user*) chỉ cần quyền truy xuất giới hạn.
- Quản trị hệ thống, người thao tác viên cập nhật, và người dùng có quyền truy cập cao hơn ở mọi cấp độ.

Xử lý bảo mật phần mềm dựa trên vai trò người dùng (*authorization*) là kỹ thuật dùng để cấp quyền sử dụng với mức độ bảo mật khác nhau tương ứng quyền hạn và độ chuyên nghiệp của mỗi người dùng trong hệ thống.

Lưu ý: Ta cần nhận biết những lớp vai trò (*role*) chính yếu của những người dùng cần truy cập đến phần mềm, sau đó gán tên vai trò cho mỗi lớp người dùng và gán mức độ tối thiểu có thể truy xuất đến mỗi vai trò. Mỗi lớp vai trò nên có đủ (và vừa đủ) mức quyền truy xuất đến công việc của họ.

❖ Xác định môi trường bảo mật phần mềm

Khi người dùng muốn sử dụng các chức năng liên quan, phần mềm đòi hỏi họ thực hiện đăng nhập vào phần mềm, nhằm để kiểm soát tài nguyên chia sẻ như tập tin, dịch vụ hệ thống, cơ sở dữ liệu. Mức độ kiểm soát của phần mềm được gọi là ngữ cảnh bảo mật. Ta cần làm việc với các người dùng khác (như người quản trị mạng) để cấp quyền truy xuất phù hợp với chức năng của phần mềm hay chia sẻ tài nguyên. Độ bảo mật của phần mềm không bị giới hạn bởi người dùng.

❖ Xác định ảnh hưởng bảo mật

Nếu tổ chức của khách hàng có sẵn cơ chế bảo mật, thì hệ thống của ta nên điều chỉnh cho phù hợp với cơ chế đã có. Nếu ta đang thực hiện hệ thống bảo mật mới, cần phân tích tác động của hệ thống trên hệ thống hiện tại:

- Hệ thống mới có làm hỏng chức năng của phần mềm hiện tại?

- Hệ thống mới có đòi hỏi phải hỗ trợ thêm đăng nhập mở rộng?
- Hệ thống mới có hạn chế một số người dùng trên những tập tin hay những tài nguyên mà họ được quyền truy cập trước đây?

❖ Kế hoạch vận hành

Khi tổ chức của khách hàng thay đổi nhân sự, người mới được thêm vào, người cũ được cập nhật và bỏ đi. Những thao tác này đòi hỏi ta hiệu chỉnh cơ sở dữ liệu bảo mật (nơi lưu thông tin quyền truy cập của người dùng).

Nếu người dùng có vị trí (địa lý, luận lý ...) khác nhau, ta cần lên kế hoạch tái tạo cơ sở dữ liệu bảo mật. Việc tái tạo đó là sự thay đổi hệ thống dữ liệu tại nơi này, sao chép đến nơi khác, để tất cả thông tin bảo mật được lưu giữ mỗi nơi. Việc đó cần được lên triển khai nhằm tạo thuận lợi là người dùng có thể đăng nhập bằng thông tin được lưu ở vị trí gần hơn so với vị trí gốc.

Lưu ý: Ta cần lập kế hoạch cho điều kiện khẩn cấp để thực hiện nếu cơ sở dữ liệu bảo mật bị ngắt hay nếu việc tạo bản sao bị hỏng.

❖ Kế hoạch kiểm soát và đăng nhập

Một hệ thống bảo mật tốt sẽ không thực hiện theo cơ chế xử lý thụ động. Thay vào đó, chúng có chức năng trợ giúp kiểm soát hoạt động của hệ thống cho vần đề bảo mật, thể hiện ở dạng nhật ký (*log file*). Tất cả thao tác của hệ thống có thể được ghi nhận với hầu hết các sự kiện liên quan đến bảo mật hệ thống. Chức năng đó có thể ghi nhận mỗi khi người dùng đăng nhập hay truy xuất đến mọi tài nguyên, nhưng điều này hiếm khi hiệu quả; thường chúng sẽ ghi nhận một số thông tin đặc biệt (như việc cố gắng đăng nhập lối).

Lưu ý: Nếu ta chỉ đơn thuần lưu trữ nhật ký hệ thống thì điều đó không có ý nghĩa. Ta cần lập kế hoạch kiểm soát nhật ký thường xuyên bởi vì ta có thể phân tích và phát hiện những nghi ngờ thông tin nhật ký hoạt động đó và đưa ra những đề nghị nếu có bất kỳ điều nghi ngờ.

❖ Xác định mức độ yêu cầu bảo mật

Việc triển khai bảo mật cho phần mềm với mức độ nhiều ít cần được cân nhắc dựa trên tính hiệu quả và chi phí hao tốn liên quan. Nếu hệ thống không lưu trữ dữ liệu có

tính nhạy cảm cao, cách tốt nhất để triển khai việc bảo mật là lưu trữ thông tin về "sự xác thực của người dùng". Nếu ta lưu trữ các thông tin cần cho bảo mật, chi phí cho việc bảo mật thông tin đặc biệt cũng cần được kiểm chứng.

Trong thực tế, ít có hệ thống nào đạt được 100% bảo mật. Ta cần xác định được *độ rủi ro bảo mật* (là tỉ lệ % tương ứng khả năng bảo mật mà hệ thống khó đạt được) chấp nhận được dựa trên dữ liệu nhạy cảm của hệ thống.

❖ **Rà soát bảo mật hiện tại**

Ta cần tuân thủ yêu cầu bảo mật của phần mềm khi phân tích chính sách bảo mật hiện tại của một tổ chức, để xác định được việc bảo mật có đạt đến những nhu cầu của hệ thống hay không và thảo luận vấn đề với người phụ trách hệ thống bảo mật của tổ chức đó để tìm ra giải pháp mang lại lợi ích cao từ đó triển khai mở rộng bảo mật.

2.2.4 Phân tích Yêu cầu Tốc độ

Tốc độ xử lý của phần mềm có thể là yêu cầu khó khăn cho nhóm phát triển phần mềm. Nếu phần mềm được thiết kế tốt, chúng có thể thực thi nhanh hơn.

Thuật ngữ tốc độ thường được dùng đồng nghĩa với sự phản hồi liên quan đến thời gian xử lý để phản hồi lại hành động của người dùng (*response time*). *Thời gian phản hồi trung bình* của phần mềm là đặc tính quan trọng, cần được kết hợp chặt chẽ với các yêu cầu khác trong giai đoạn thiết kế.

Lưu ý:

- Trong phần mềm dạng phân tán, khi nói về tốc độ, ta cần nhấn mạnh sự khác biệt quan trọng khi phần mềm được có nhu cầu cao hay trung bình. Tại một số thời điểm, như buổi tối hay cuối tuần, vì phần mềm chỉ phục vụ số lượng nhỏ người dùng, thì tốc độ nó sẽ trên trung bình. Ở thời điểm khác khi số lượng người dùng tăng cao, tốc độ của phần mềm sẽ được thay đổi cho phép người dùng thực thi ở mức vừa đủ. Vì vậy, trong phần mềm phân tán, mục tiêu tốc độ gồm cả tốc độ trung bình và cao.
- Nhiều phần mềm chạy chậm bởi thiết kế thiếu sót, có thể do nhiều nguyên nhân, như tương thích giữa phần cứng và các yếu tố khác.

Ta cần nhận biết rõ ràng về các yêu cầu tốc độ của phần mềm trước khi bắt đầu quy trình thiết kế dựa theo các tiêu chí sau:

- *Tần suất giao dịch*: việc cung cấp dịch vụ phụ thuộc vào số người dùng trong một khoảng thời gian. Số giao tác mỗi phút (*transactions per minute*, TPM) là độ đo tốc độ của hệ thống cơ sở dữ liệu.
- *Băng thông*: sự phản hồi nhanh (hay chậm) của phần mềm xác định mức băng thông mạng (độ rộng của đường truyền mạng) cao (hay thấp). Băng thông thường được đo bằng megabit/giây.
- *Khả năng chứa*: mức lưu trữ (cả phần bộ nhớ chính và phụ) luôn sẵn sàng đáp ứng đổi với phần mềm là vấn đề lưu tâm quan trọng cho tốc độ chung của phần mềm. Mức độ đòi hỏi sử dụng bộ nhớ của phần mềm sẽ gây ra những khác biệt lớn cho tốc độ của chúng.
- *Nút thắt*: mỗi hệ thống đều có phần giới hạn tốc độ. Ví dụ nếu CPU có tốc độ nhanh cũng không cải thiện được gì nếu phải chờ dữ liệu từ ổ cứng thực thi quá chậm. Lúc này, tốc độ ổ cứng là nút thắt của hệ thống. Ta cần nhận biết nút thắt của hệ thống để cải thiện chúng nhằm nâng cao tốc độ. Việc nhận biết nút thắt có thể thực hiện bằng công cụ báo cáo hệ thống *Windows NT Performance Monitor*.

2.2.5 Phân tích Yêu cầu Vận hành

Khi vận hành phần mềm, chi phí triển khai và chi phí vận hành là vấn đề quan trọng cần được xem xét cẩn thận với hướng giải quyết như sau:

- Chi phí triển khai có thể được giảm bớt bằng cách phân phối trực tuyến hay phần mềm những thủ tục tự động cài đặt, và thao tác vận hành có thể tự động hóa bằng các quy trình tin học.
- Chi phí vận hành có thể được tiết kiệm theo nhiều cách, nhưng cách tốt nhất là đảm bảo chương trình được kiểm thử và chạy kiểm tra (*debug*) đầy đủ trước khi đưa vào triển khai, điều đó sẽ giúp tăng chất lượng của phần mềm và giảm lỗi hay hỏng hóc xảy ra trong lúc vận hành nên giảm chi phí bảo trì giai đoạn vận hành.
- Ngoài ra, trong trường hợp phần cứng, phần mềm là thành phần được mua chứ không được phát triển, ta có thể nhận sự chấp thuận vận hành từ nhà xưởng hay

người ủy thác của sản phẩm. Vận hành sản phẩm trung gian tiết kiệm chi phí thuê nhân viên mới hay huấn luyện lại nhân viên cũ để duy trì một hay nhiều thành phần của hệ thống.

Giảm chi phí vận hành đòi hỏi sự tự thỏa mãn lợi nhuận trong thời ngắn đối với những lợi ích trong tương lai. Giảm chi phí vận hành lâu dài thường đòi hỏi đầu tư đón đầu trong tự động hóa phần cứng và phần mềm.

2.2.6 Phân tích Khả năng Mở rộng Yêu cầu

Theo diễn tiến thời gian, những yêu cầu của giải pháp ban đầu sẽ dần thay đổi. Người dùng cần những chức năng mới, các quy luật đặt ra ban đầu sẽ bị sửa đổi, dẫn đến cả phần cứng phần mềm cũng cần thay đổi theo.

Phần mềm thiết kế tốt là phần mềm có khả năng mở rộng được và có thể uyển chuyển cải thiện mà không phải xây dựng lại hoàn toàn. Khả năng mở rộng của phần mềm tỉ lệ nghịch với lượng công việc cần hoàn thành để thêm những đặc trưng mới và có thể đạt được thông qua những ý nghĩa khác nhau.

Có ba cách đạt đến những khả năng hạn định là:

- Lưu trữ thông tin và quy tắc nghiệp vụ vào cơ sở dữ liệu hơn là lập trình biểu diễn chúng trong đối tượng nghiệp vụ. Theo đó, nếu số chức năng hay thủ tục của phần mềm ban đầu cần thay đổi, nó có thể thay đổi trong cơ sở dữ liệu mà không cần thay đổi mã nguồn chương trình.
- Đặt mã nguồn vào trong đoạn mã kịch bản (script) được làm rõ hơn khi biên dịch chương trình; đoạn script có thể bị thay đổi một cách dễ dàng không đòi hỏi bất kỳ biên dịch hay cài đặt lại tập tin nhị phân.
- Chia phần mềm thành những đối tượng thành phần với từng nhiệm vụ riêng lẻ. Nếu những yêu cầu của các nhiệm vụ đặc biệt thay đổi, đối tượng tương ứng có thể bị thay đổi và biên dịch lại mà không gây ảnh hưởng bất kỳ đối tượng khác cũng như được thêm vào dễ dàng. Phương pháp này với các đối tượng nghiệp vụ ưu điểm nhiều hơn hai phương pháp trên trong khi vẫn đảm bảo tốt khả năng mở rộng.

2.2.7 Phân tích Yêu cầu Sẵn dùng

Các phần mềm luôn được thiết kế để thực thi đều đặn hàng ngày. Chúng rất cần thiết cho sự thành công của doanh nghiệp và cần có mức độ sẵn sàng cao để tránh được các bảo trì, sửa chữa, phát sinh không theo kế hoạch. Với những phần mềm đã có tính sẵn sàng, chúng không được gây ra lỗi. Bất kỳ thành phần phần mềm nào bị hỏng hay không sẵn sàng thì ta nên khởi động lại ngay khi có thể.

Việc bảo trì có kế hoạch cũng tác động đến tính sẵn sàng của phần mềm. Một máy chủ chứa các phần mềm lý tưởng luôn có bản sao lưu có thể khởi động khi máy chủ bảo trì. Phần mềm có độ sẵn sàng cao cần có cách luân phiên để kết nối mạng trong trường hợp mạng WAN, LAN ngưng hoạt động

Lưu ý: Tính sẵn sàng liên quan đến nghiệp vụ. Tính sẵn sàng của phần mềm càng cao, giá trị của phần mềm càng cao. Với những hệ thống trọng yếu, giá trị đối với tổ chức ở bất kỳ thời điểm nào hoàn toàn phù hợp và dẫn đến điều chỉnh chi phí thiết kế 100% cho vấn đề phần mềm sẵn sàng. Phần mềm khác đơn giản cần trở nên sẵn sàng hầu hết mọi lúc.

2.2.8 Phân tích Yếu tố Con người

Phần quan trọng trong thiết kế phần mềm chính là phần liên quan đến yếu tố con người. Ta nên xác định những kinh nghiệm sử dụng phần mềm mà người dùng cần có. Với bất cứ phần mềm nào, để có kinh nghiệm người dùng càng tốt thì chi phí chế tạo càng cao.

Ta bắt đầu với việc định nghĩa mục tiêu của người dùng, qua đó xác định được các dạng người dùng với những nhu cầu đặc biệt liên quan, qua đó ta có thể sửa đổi phần mềm thích ứng những nhu cầu đó.

2.2.9 Phân tích Yêu cầu Tích hợp

Trong các giải pháp giao tiếp với phần mềm kế thừa, để thực hiện việc truy xuất cơ sở dữ liệu hiện có, hay chuyển đổi dữ liệu cũ sang khuôn dạng mới, ta cần đưa kế hoạch tích hợp giữa phần mềm mới với hệ thống hiện hành thông qua những công cụ có sẵn (ví dụ ODBC) hay xây dựng tiện ích chuyển đổi riêng biệt.

Khi nhu cầu phát sinh lớn hơn, cơ sở dữ liệu cần được thiết kế lại dựa trên những kỹ thuật thiết kế cơ sở dữ liệu mới để đáp ứng nhu cầu nâng cấp cơ sở dữ liệu trong hệ thống phần mềm. Vấn đề này cần được thực hiện cẩn thận vì chúng có thể phá vỡ tất cả mã nguồn của cơ sở dữ liệu hiện tại. Trước khi cải tiến cấu trúc dữ liệu, ta cần đảm bảo các mã nguồn hiện tại có thể truy xuất đến cơ sở dữ liệu. Tất cả mã nguồn hiện tại phải được soát lại, có thể viết lại.

2.2.10 Phân tích Thực tiễn Nghiệp vụ tồn tại

Phần định nghĩa trong quy tắc nghiệp vụ (*business rule*) có liên quan đến sự hiểu biết ngữ cảnh của ta (*nhóm phân tích*) trong thao tác của những quy tắc đó. Khi hiểu được những nghiệp vụ thực tế (liên quan đến phần mềm) của doanh nghiệp, ta sẽ tránh được sai sót đồng thời và có thể tìm được cách tốt hơn, hiệu quả hơn cho việc hiện thực tiễn trình nghiệp vụ. Việc ta hiểu được vấn đề thực tế trong mỗi tiến trình sẽ giúp ngăn ngừa các lỗi tầm thường.

Ngoài ra, việc hiểu được cấu trúc tổ chức và sơ đồ nghiệp vụ của doanh nghiệp là yếu tố quyết định sự thành công của bước phân tích yêu cầu và xây dựng phần mềm. Nếu ta không hiểu rõ sơ đồ tổ chức, ta không có được các thiết kế phù hợp cho phần mềm hay cho quá trình hiện thực đúng. Sơ đồ tổ chức cũng giúp ta tìm kiếm được nhiều thông tin hữu ích liên quan.

Để có được phần mềm tốt từ giai đoạn phát triển đến lúc sử dụng, ta cần biết chi tiết về hệ thống mạng và chính sách hạ tầng của khách hàng để:

- Biết được người chịu trách nhiệm bảo trì, bảo mật, tính toàn vẹn, khả năng phản hồi tương tác trên mạng,
- Học những tiến trình, chính sách liên quan chạy trên phần mềm mới,
- Tìm ra cách kiểm soát chất lượng và chuẩn bị dịch vụ kiểm thử (sẵn sàng cho việc áp dụng vào phần mềm).

Ngoài ra, ta có thể triển khai phương pháp đặc biệt (thiết kế, triển khai thực tế) nhằm đảm bảo kế hoạch hiện thực phù hợp với ngân sách phát triển.

Cuối cùng, ta cần tuân thủ nguyên tắc cốt lõi là *Nhận biết nhu cầu khách hàng và cố gắng thực hiện chúng*, đôi khi việc này sẽ khó khăn (khi khách hàng không biết

được nhu cầu của họ là gì) nhưng đó là cách giúp ta xây dựng thành công phần mềm phần mềm.

2.2.11 Phân tích Yêu cầu Khả năng và Quy mô

Sự thành công của phần mềm sẽ thu hút người dùng. Đặc biệt, nếu phần mềm thực thi trên môi trường mạng Internet thì sự thành công của phần mềm sẽ đồng nghĩa với việc tăng số lượng người dùng và nhu cầu người dùng. Để đáp ứng việc tăng quy mô và nhu cầu của người dùng, khi thiết kế phần mềm ta phải chú ý đến tính chất quy mô của phần mềm qua đó có thể hỗ trợ nâng cấp cho phép phần mềm có thể phục vụ nhiều người hơn.

Để nâng cao khả năng phục vụ của phần mềm, giải pháp đơn giản là nâng cấp cơ sở hạ tầng (mua CPU nhanh hơn, nhiều RAM, kết nối mạng tốt hơn) cả phía người dùng và phía hệ thống phục vụ (các máy chủ). Một giải pháp khác là phát triển phần mềm ở dạng phân tán để có thể hoạt động trên nhiều máy tính và máy chủ cùng lúc, từ đó giúp ta kiểm soát và đáp ứng việc phân phối tài nguyên và thời gian xử lý được hợp lý hơn. Tuy nhiên, điều này làm gia tăng đáng kể tính phức tạp của hệ thống, nên chiến lược thiết kế cần được cân nhắc theo hướng đáp ứng khả năng cao và quy mô lớn cho phần mềm.

2.3 XÁC ĐỊNH YÊU CẦU

❖ Mục tiêu của việc xác định yêu cầu

Ta cần xác định thật *chính xác* và *đầy đủ* các yêu cầu đặt ra cho phần mềm sẽ được xây dựng.

❖ Kết quả nhận được sau giai đoạn xác định yêu cầu

- Danh sách các công việc (liên quan đến những chức năng của phần mềm) sẽ được thực hiện trên máy tính,
- Những mô tả chi tiết về các công việc này khi được triển khai vận hành trong thế giới thực,
- Thông tin khái quát về các hoạt động trong thế giới thực.

2.3.1 Yêu cầu và Mô tả yêu cầu

- *Yêu cầu* (hay yêu cầu phần mềm) là công việc (trong phần mềm) muốn thực hiện trên máy tính liên quan. Những công việc này phải xuất phát từ thực tế chứ không thuần túy tin học.
- *Mô tả yêu cầu* là mô tả đầy đủ các thông tin liên quan đến công việc tương ứng. Các mô tả này dùng làm cơ sở để nghiệm thu và đánh giá phần mềm khi được chuyển giao.

Các yêu cầu của phần mềm (với các thông tin liên quan đến công việc tương ứng) cần được mô tả thật rõ ràng, cụ thể, đầy đủ và chính xác. Việc mô tả sơ sài, mơ hồ yêu cầu phần mềm sẽ dẫn đến việc hiểu nhầm giữa chuyên viên tin học (người thực hiện phần mềm) và khách hàng (người đặt hàng thực hiện phần mềm), gây nhiều hao tốn, lãng phí công sức và chi phí.

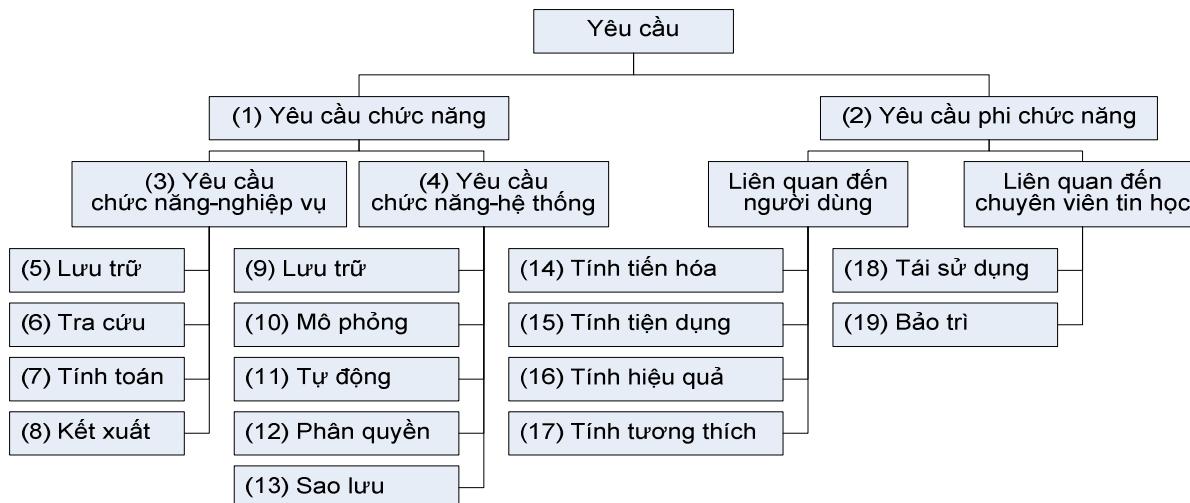
Khi xác định yêu cầu phần mềm, các thông tin quan trọng ta cần lưu ý là:

- *Tên công việc ứng với từng yêu cầu*: ta cần xác định cụ thể, tránh dùng các tên chung chung, mơ hồ. Ví dụ: tên công việc như “Quản lý độc giả” là chung chung, mơ hồ nên ta cần cụ thể hơn như “Đăng ký mượn sách”, “Gia hạn thẻ độc giả”, “Trả sách”.
- *Người hoặc bộ phận sẽ thực hiện công việc*: ta cần xác định chính xác người hoặc bộ phận sẽ thực hiện công việc trên máy tính (còn gọi là người dùng). Những người dùng có vai trò và công việc thực hiện tương tự nhau sẽ được xếp vào cùng loại người dùng (thông thường mỗi loại sẽ tương ứng với một bộ phận trong thế giới thực). Cùng một công việc có thể có nhiều loại người dùng khác nhau; ngược lại một loại người dùng có thể thực hiện nhiều công việc khác nhau.
- *Thời gian và Địa điểm thực hiện công việc*: ta cần xác định chính xác địa điểm, thời điểm tiến hành công việc. Các thông tin này sẽ có ý nghĩa nhất định trong một số trường hợp đặc thù.
- *Cách thức tiến hành công việc cùng với các quy định liên quan*: đây là phần chính yếu khi tiến hành mô tả yêu cầu. Đối với loại thông tin này cần đặc biệt quan tâm đến một số yếu tố sau:

- Các quy định cần kiểm tra khi thực hiện việc ghi nhận thông tin. Ví dụ quy định: chỉ cho mượn sách đối với những độc giả có thẻ độc giả còn hạn, số sách đang mượn chưa đến 2 và không có sách mượn quá hạn.
- Các quy định, công thức tính toán khi thực hiện việc tính toán. Ví dụ: quy định: mỗi ngày trả trễ phạt 1500 đồng/ngày. Từ ngày trả trễ thứ 10 trở đi sẽ phạt 5000 đồng/ngày, thu hồi thẻ độc giả 2 tuần.

2.3.2 Phân loại yêu cầu

Các yêu cầu được phân loại như thể hiện trên sơ đồ sau:



Hình 2.3: Phân loại các yêu cầu

❖ Đặc tả chi tiết từng loại yêu cầu:

(1) *Yêu cầu chức năng*: là danh sách các công việc sẽ được thực hiện trên máy tính cùng với các thông tin mô tả tương ứng.

(2) *Yêu cầu phi chức năng*: là các yêu cầu liên quan đến chất lượng phần mềm, là sự ràng buộc cách thức thực hiện các yêu cầu chức năng.

(3) *Yêu cầu chức năng nghiệp vụ*: là các chức năng phần mềm tương ứng với công việc có thật trong thế giới thực.

(4) *Yêu cầu chức năng hệ thống*: là các chức năng phần mềm phát sinh thêm khi thực hiện công việc trên máy tính thay vì trong thế giới thực hoặc các chức năng không tương ứng với bất kỳ công việc nào trong thế giới thực.

(5) *Chức năng lưu trữ*: Tương ứng với công việc ghi chép thông tin trên sổ sách (kèm theo các quy định khi ghi chép). Ví dụ: Ghi nhận việc cho mượn sách của một thư viện theo quy định mượn ...

(6) *Chức năng tra cứu*: tương ứng với công việc tìm kiếm, theo dõi hoạt động và xem thông tin về một đối tượng. Ví dụ: Tìm và xem tình trạng sách

(7) *Chức năng tính toán*: liên quan việc tính toán (quy định/ công thức). Ví dụ: Tính tiền phạt trả sách trễ theo quy định phạt của thư viện.

(8) *Chức năng kết xuất*: tương ứng với việc lập báo cáo (theo biểu mẫu). Ví dụ: Lập báo cáo thống kê về số lượt mượn sách theo loại trong năm.

(9) *Chức năng môi trường*: định cấu hình thiết bị, thời gian, nhân sự ... Ví dụ: Số nhân công, chọn loại máy in, khổ giấy, niên khóa hiện hành ...

(10) *Chức năng mô phỏng*: mô phỏng hoạt động của thế giới thực. Ví dụ: Mô phỏng quá trình hoạt động của chức năng hay thao tác xử lý

(11) *Chức năng tự động*: tự động thông báo, nhắc nhở người dùng. Ví dụ: Nhắc nhở thủ thư gửi giấy báo đòi sách mượn quá hạn.

(12) *Chức năng phân quyền*: phân quyền sử dụng cho các loại người dùng. Ví dụ: Phân quyền cho loại người dùng: (i) Quản trị hệ thống: có quyền sử dụng tất cả các chức năng; (ii) Thủ thư: chỉ dùng các chức năng liên quan đến việc cho mượn và trả sách; (iii) Độc giả: chỉ sử dụng chức năng tra cứu.

(13) *Chức năng sao lưu*: Sao lưu, phục hồi dữ liệu. Ví dụ: Sao lưu thông tin học sinh đã ra trường, phục hồi lại khi cần thiết

(14) *Tính tiến hóa*: đây là các yêu cầu liên quan đến việc cho phép người dùng thay đổi lại cách mô tả của yêu cầu chức năng (quy định, quy tắc tính toán), biểu mẫu nào đó khi đang dùng phần mềm đã được chuyển giao. Điều này đòi hỏi phải có dự kiến về các thay đổi trên thành phần dữ liệu và xử lý. Ví dụ: Cho thay đổi quy định về sổ sách mượn tối đa, mức phạt khi trả trễ.

(15) *Tính tiện dụng*: là các yêu cầu liên quan đến hình thức giao diện của phần mềm, thể hiện ở sự tự nhiên, dễ sử dụng, dễ học, đầy đủ thông tin ... Ví dụ: Giao diện

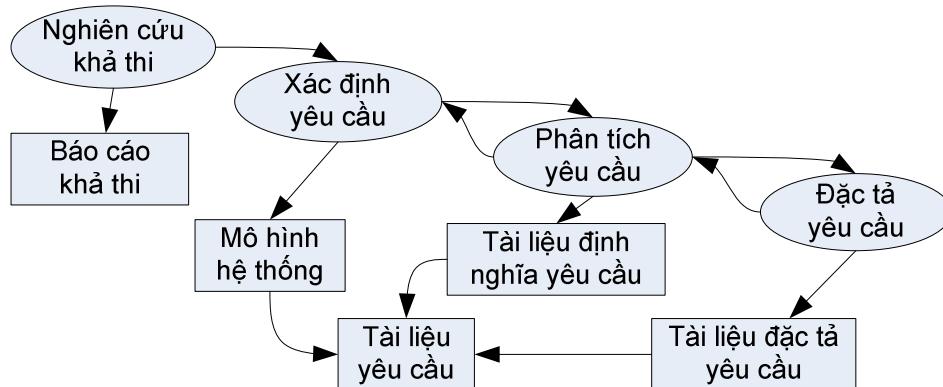
nhập hóa phiếu mượn hay trả sách dạng form, dòng nhập thể hiện bằng ô sáng và báo lỗi khi số liệu nhập sai.

(16) *Tính hiệu quả*: đây là yêu cầu liên quan đến thời gian thực hiện các chức năng phần mềm, dung lượng lưu trữ, chi phí sử dụng tài nguyên hệ thống như sử dụng tối ưu các không gian, thao tác thực hiện nhanh ... Ví dụ: Thời gian tra cứu sách, tra cứu độc giả không quá 10 giây.

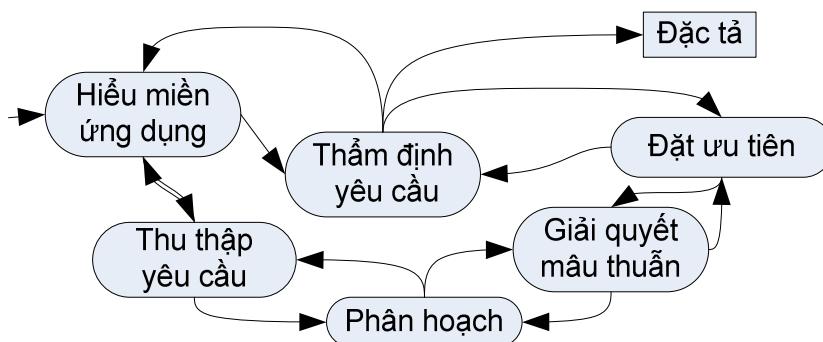
(17) *Tính tương thích*: là các yêu cầu liên quan đến việc chuyển đổi dữ liệu giữa phần mềm đang xét và các phần mềm khác, sự nhất quán giữa các màn hình trong hệ thống. Ví dụ: Cho nhập thông tin sách mới từ tập tin hay từ thiết bị đọc mã vạch.

(18) *Tính tái sử dụng*: (do chuyên viên tin học đảm trách)

(19) *Tính bảo trì*: (do chuyên viên tin học đảm trách) là các yêu cầu cho phép thay đổi mà không làm ảnh hưởng đến phần mềm



Hình 2.4: Quy trình Xây dựng Yêu cầu



Hình 2.5: Quy trình Phân tích Yêu cầu

2.3.3 Các bước xác định yêu cầu

❖ Quá trình thực hiện xác định yêu cầu

Gồm 2 bước chính là:

- Bước 1: Khảo sát hiện trạng, kết quả nhận được là báo cáo hiện trạng.
- Bước 2: *Lập danh sách các yêu cầu*, kết quả nhận được là *danh sách các yêu cầu* sẽ được thực hiện trên máy tính.

❖ Đối tượng tham gia xác định yêu cầu

Gồm 2 nhóm người luôn phối hợp chặt chẽ để có thể xác định đầy đủ và chính xác các yêu cầu là:

- *Chuyên viên tin học*: là những người hiểu rõ về khả năng của máy tính. Họ phải tìm hiểu thật chi tiết về công việc của chuyên gia nhằm tránh sự hiểu nhầm cho những bước phân tích sau này.
- *Chuyên gia*: là những người hiểu rõ về công việc của mình. Họ cần lắng nghe ý kiến của các chuyên viên tin học để đảm bảo các yêu cầu của họ là có thể thực hiện được với chi phí và thời gian hợp lý.

Sau đây ta sẽ phân tích chi tiết từng bước quy trình thực hiện.

2.3.3.1 Khảo sát hiện trạng

Các chuyên viên tin học sẽ tìm hiểu hiện trạng công việc của chuyên gia.

❖ Các hình thức thực hiện phổ biến

- *Quan sát*: theo dõi các hoạt động diễn ra ở thế giới thực có liên quan, có thể ghi âm, ghi hình đối với những tình huống mang tính phức tạp, quan trọng, cần sự chính xác cao. Ví dụ: Quan sát thao tác cho mượn sách của một thủ thư tại một thư viện,
- *Phỏng vấn trực tiếp*: tổ chức phỏng vấn bắt đầu từ cấp lãnh đạo dẫn xuống các vị trí công việc. Có thể sử dụng các bảng câu hỏi có sẵn các câu trả lời cho đối tượng được phỏng vấn lựa chọn ...

- *Thu thập thông tin, tài liệu*: các công thức tính toán, quy định; các bảng biểu, mẫu giấy tờ có ít nhiều liên quan. Ví dụ: Phiếu mượn sách tại thư viện của trường đại học.

❖ Quy trình thực hiện

- *Tìm hiểu tổng quan về thế giới thực*: bao gồm Quy mô hoạt động và Các hoạt động mà đơn vị có tham gia.
- *Tìm hiểu hiện trạng tổ chức* (cơ cấu tổ chức): Người tiến hành khảo sát hiện trạng cần hiểu rõ cơ cấu tổ chức các bộ phận của thế giới thực, đặc biệt là 2 yếu tố: trách nhiệm và quyền hạn. Việc hiểu rõ cơ cấu tổ chức sẽ giúp xác định bộ phận nào sẽ sử dụng phần mềm để có thể lên kế hoạch tiếp tục khảo sát chi tiết hơn bộ phận đó. Cơ cấu tổ chức bao gồm: Đối nội, Đối ngoại, Các chức danh (Ví dụ: nhân viên nhập liệu, thủ thư, ...). Ta cần sử dụng các sơ đồ để vẽ lại cơ cấu tổ chức.
- *Tìm hiểu hiện trạng nghiệp vụ*: Hiện trạng này thường xảy ra tại các vị trí công việc. Với bộ phận được chọn khảo sát chi tiết, người thực hiện khảo sát cần lập danh sách các công việc mà bộ phận này phụ trách, sau đó tìm hiểu các thông tin chi tiết cho từng công việc (thông tin mô tả yêu cầu phần mềm). Việc tìm hiểu dựa trên các ý sau: Thông tin đầu vào, Quá trình xử lý, Thông tin kết xuất.
- *Tiến hành xếp loại nghiệp vụ* vào 4 loại sau nhằm tránh thiếu sót khi tìm hiểu các thông tin: lưu trữ, tra cứu, tính toán, tổng hợp/thống kê.

2.3.3.2 Lập danh sách các yêu cầu

Để có được danh sách đầy đủ và chính xác các, quá trình lập danh sách các yêu cầu cần theo các bước sau:

❖ Xác định yêu cầu chức năng nghiệp vụ

Cách tiến hành

Chuyên gia đề xuất và chuyên viên tin học sẽ xem xét lại

Bước tiến hành

- Xác định bộ phận (người dùng) sẽ sử dụng phần mềm

- Xác định các công việc mà người dùng sẽ thực hiện trên phần mềm theo từng loại công việc sau: Lưu trữ, Tra cứu, Tính toán, Kết xuất
- Lần lượt lập bảng yêu cầu chức năng nghiệp vụ, bảng quy định/công thức và các biểu mẫu.

Mẫu 1: Bảng yêu cầu chức năng nghiệp vụ

Bộ phận (người thực hiện): Mã số:

STT	Công việc	Loại công việc	Quy định/công thức liên quan	Biểu mẫu liên quan	Ghi chú
1					
2					

Mẫu 2: Bảng Quy định/ Công thức liên quan

STT	Mã số	Tên Quy định/ Công thức	Mô tả chi tiết	Ghi chú
1	QĐ 1			
2	QĐ 2			

Các biểu mẫu được mô tả chi tiết ngay sau bảng quy định/Công thức

Ví dụ: Xét phần mềm quản lý thư viện

(i) Bảng yêu cầu chức năng

Bộ phận: Thủ thư. Mã số: TT

STT	Công việc	Loại công việc	Quy định/Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Cho mượn sách	Lưu trữ	TT_QĐ 1	TT_BM 1	
2	Nhận trả sách	Lưu trữ	Chỉ nhận lại những sách đã cho mượn	TT_BM 1	
3	Tính tiền phạt	Tính toán	Mỗi ngày trả trễ phạt: - 1000 đồng/ngày: từ ngày thứ nhất đến ngày thứ 5 - 3000 đồng/ngày: từ ngày thứ 6 trở đi.		
4	Tính tiền đèn	Tính toán	Tiền đèn cho sách bị mất dựa trên giá thị trường tại thời điểm hiện hành.		
5.	Tra cứu sách	Tra cứu	Việc tìm sách dựa trên các thông tin: tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
6.	Gửi giấy báo đòi sách	Kết xuất	Sách mượn quá hạn 3 ngày sẽ tự động gửi giấy báo cho đến khi sách được trả hoặc đã tính xong tiền đèn sách	TT_BM 2	

(ii) Bảng yêu cầu chức năng nghiệp vụ

STT	Mã số	Tên Quy định/Công thức	Mô tả chi tiết	Ghi chú
1	QĐ 1	Quy định cho mượn sách	Chỉ cho mượn sách khi: - Thẻ độc giả còn hạn - Độc giả chưa mượn hết số sách quy định - Độc giả không có sách mượn quá hạn - Sách hiện không có người mượn	Độc giả mượn sách sẽ phải gửi lại thẻ độc giả tại bộ phận bạn đọc, nhận phiếu mượn sách (TT_BM 1, tìm kiếm mã số sách mượn và điền các sách cần mượn vào phiếu, xong gửi cho thủ thư)

(iii) Bảng Quy định/ Công thức liên quan

TT_BM 1:**PHIẾU MUỢN SÁCH**

Số thẻ: Số phiếu mượn:

Họ và tên: Ngày mượn:

[] Mượn về nhà [] Đọc tại chỗ

STT	Mã sách	Tên sách	Tác giả	Mã loại
1				
2				

Ngày ... tháng ... năm

TT_BM 2:**GIẤY BÁO MUỢN SÁCH QUÁ HẠN**

Thân gửi: Địa chỉ:

Chúng tôi xin thông báo rằng, anh (chị) đã mượn của thư viện chúng tôi những quyển sách sau:

STT	Mã sách	Tên sách	Ngày mượn	Đến hôm nay đã quá hạn
1				
2				

Vậy thông báo anh(chị) vui lòng đem sách đến trả. Và mang theo số tiền đồng để trả phí sách trễ.

Bộ phận: Độc giả. Mã số: ĐG

STT	Công việc	Loại công việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Tìm sách	Tra cứu	Việc tìm sách dựa trên các thông tin: tên sách, tên tác giả, nhà xuất bản, năm xuất bản		
2	Đăng ký mượn sách	Lưu trữ	Độc giả phải có thẻ độc giả.	TT_BM 1	Mọi độc giả có thẻ mượn sách đều có thể đăng ký mượn sách. Tuy nhiên, hệ thống sẽ thông báo khi thẻ mượn sách của độc giả đã hết hạn sử dụng.

Bộ phận: Quản lý độc giả. Mã số: QLDG

STT	Công việc	Loại việc	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1	Làm thẻ độc giả mới	Lưu trữ	Chỉ cấp thẻ độc giả có độ tuổi từ 18 trở lên và có chứng minh thư. Lệ phí làm thẻ độc giả là 5000 đồng/thẻ. Một số chứng minh thư chỉ có thẻ duy nhất một thẻ độc giả	QLDGBM1 QLDGBM2	Độc giả có yêu cầu làm thẻ mượn sách sẽ được nhận phiếu đăng ký để điền thông tin vào (QLD_G_BM 1), sau đó bộ phận quản lý độc giả tiến hành cấp thẻ và thu lệ phí theo quy định (QLD_G_BM 2)
2	Gia hạn thẻ độc giả	Lưu trữ	Gia hạn thẻ theo yêu cầu của độc giả và thời gian quá hạn không được quá 3 tháng. Sau thời gian 3 tháng, những thẻ hết hạn sẽ bị hủy.		
3	Hủy thẻ độc giả	Lưu trữ	Hủy bỏ các thẻ độc giả đã quá hạn đăng ký 3 tháng.		

QLDG_BM 1:**PHIẾU ĐĂNG KÝ LÀM THẺ MUỢN SÁCH**

Họ và tên: Năm sinh:

Địa chỉ thường trú: Nghề nghiệp:

Ngày đăng ký:

QLDG_BM 2:**THẺ ĐỘC GIẢ**

Họ và tên:

Địa chỉ:

Ngày tháng năm:

Bộ phận: Quản lý sách. Mã số: QLS

STT	Công việc	Loại	Quy định/ Công thức liên quan	Biểu mẫu liên quan	Ghi chú
1.	Nhận sách mới vào kho sách.	Lưu trữ		QLSBM 1	Khi có sách mới nhập về, bộ phận quản lý sách có trách nhiệm rà xét xem số sách đó đã có hay chưa, nếu chưa thì lập thẻ quản lý sách và định mã số sách mới. Nếu có rồi thì gọi lại thẻ cũ để cập nhật bổ sung số lượng.
2.	Thanh lý sách cũ	Lưu trữ	Các sách hư, không đọc được		
3.	Lập báo cáo các sách cần thanh lý	Kết xuất		QLS_BM 2	
4.	Lập báo cáo sách mượn	Kết xuất		QLS_BM 3	

QLS_BM 1:

THẺ QUẢN LÝ SÁCH

Tên sách:

Tập: Số trang: Số lượng:

Năm xuất bản: Mã ngôn ngữ: Ngôn ngữ:

Mã nhà xuất bản: Nhà xuất bản:

Mã phân loại: Phân loại:

Mã tác giả: Tác giả:

Mã vị trí: Khu: Kệ: Ngăn:

QLS_BM 2:

DANH SÁCH CÁC SÁCH CẦN THANH LÝ

STT	Mã sách	Tên sách	Tác giả	Năm sản xuất	Ngày nhập kho	Tình trạng
1						
2						

Ngày lập báo cáo: Người lập:

QLS_BM 3:

BÁO CÁO THỐNG KÊ SÁCH MUỢN

Từ ngày: Đến ngày:

STT	Mã sách	Tên sách	Tác giả	Số lượt mượn
1.				
2.				

Ngày lập báo cáo: Người lập:

❖ Xác định yêu cầu chức năng hệ thống và yêu cầu chất lượng**Cách tiến hành**

Chuyên viên tin học và chuyên gia cùng đề xuất và xem xét các yêu cầu.

Bước tiến hành

- Bước 1: Xem xét các yêu cầu chức năng hệ thống cơ bản, thông dụng (yêu cầu phát sinh thêm do thực hiện các công việc trên máy tính): phân quyền, sao lưu, phục hồi, định cấu hình hệ thống, ...
- Bước 2: Xem xét các yêu cầu chức năng hệ thống chuyên biệt (yêu cầu về công việc mới, chỉ có thể tiến hành khi thực hiện trên máy tính)
- Bước 3: Xem xét các yêu cầu về chất lượng theo từng loại tiêu chuẩn sau: Tiết kiệm, Tiện dụng, Hiệu quả, Tương thích.

Sau đó lập bảng yêu cầu tương ứng theo mẫu sau:

Mẫu 3: Bảng yêu cầu chức năng hệ thống.

STT	Nội dung	Mô tả chi tiết	Ghi chú
1.			
2			

Mẫu 4: Bảng yêu cầu về chất lượng

STT	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1.				
2.				

Ví dụ: Xét phần mềm quản lý thư viện (được xây dựng nhằm phục vụ cho 4 bộ phận là: độc giả, thủ thư, ban giám đốc và quản trị hệ thống).

(i) Bảng yêu cầu chức năng hệ thống:

STT	Nội dung	Mô tả chi tiết	Ghi chú
1	Phân quyền sử dụng	<ul style="list-style-type: none"> - Người quản trị: được phép sử dụng tất cả các chức năng - Độc giả: chỉ tra cứu sách và đăng ký mượn sách - Ban giám đốc: chỉ tra cứu sách và lập các báo cáo thống kê - Thủ thư: tất cả các chức năng, ngoại trừ chức năng phân quyền, sao lưu và phục hồi dữ liệu 	

(ii) Bảng yêu cầu về chất lượng hệ thống:

STT	Nội dung	Tiêu chuẩn	Mô tả chi tiết	Ghi chú
1	Cho phép thay đổi quy định tính tiền phạt	Tiến hóa	Người dùng phần mềm có thể thay đổi đơn giá phạt và biên các mức phạt.	
2	Hình thức tra cứu thật tiện dụng, tự nhiên, trực quan. Dễ sử dụng cho cả những người không chuyên tin học.	Tiện dụng	Hỗ trợ khả năng tra cứu gần đúng, tra cứu theo nội dung ...	
3	Cho phép nhập sách mới từ tập tin Excel có sẵn. Các màn hình có sự nhất quán chung	Tương thích	Có thể nhập trực tiếp danh sách các sách mới có trước trên tập tin Excel với cấu trúc hợp lý.	
4	Tốc độ thực hiện việc cho mượn và tra cứu sách nhanh	Hiệu quả	<ul style="list-style-type: none"> Tối đa 30 giây cho mỗi phiếu mượn sách. Hỗ trợ thiết bị đọc mã vạch. Tối đa 10 giây phải có kết quả tra cứu. 	

2.4 MÔ HÌNH HOÁ YÊU CẦU HỆ THỐNG

Các mô tả yêu cầu trong giai đoạn xác định yêu cầu chỉ trình bày chủ yếu các thông tin liên quan đến việc thực hiện các nghiệp vụ trong thế giới thực và chưa thể hiện rõ nét việc thực hiện các nghiệp vụ này trên máy tính. *Mô tả thông qua các văn bản dễ gây ra nhầm lẫn và không trực quan.*

Ví dụ: Xét yêu cầu lập phiếu mượn sách, yêu cầu này chỉ mô tả biểu mẫu và quy định lập phiếu và chưa thể hiện cách thức lập phiếu trên máy tính.

Khi thực hiện mô hình hóa yêu cầu hệ thống, ta cần quan tâm đến:

- *Mục tiêu*: hiểu chi tiết hơn về ngữ cảnh vấn đề cần giải quyết một cách trực quan và bản chất nhất về các thông tin cốt lõi trong yêu cầu.
- *Kết quả*: thu được mô hình mô tả toàn bộ hoạt động của hệ thống.
- *Kỹ thuật phân tích*: cách tiến hành để thu thập được những yêu cầu của người sử dụng, từ đó trình bày lại nhu cầu đó trên mô hình rồi chi tiết hóa chúng bằng đặc tả chức năng và đặc tả dữ liệu thông qua phân tích góc nhìn, đối tượng, dữ liệu thu thập được ở các bước trên.

Trước khi đi vào tìm hiểu các phương pháp biểu diễn bằng mô hình, ta hãy xem qua một số nguyên lý phân tích.

2.4.1 Các Nguyên lý Mô hình hóa

❖ Nguyên lý Phân tích 1: *Mô hình hóa Miền thông tin*

Ta phải *hiểu và biểu diễn* được miền thông tin liên quan:

- Định danh dữ liệu (đối tượng, thực thể), Định nghĩa các thuộc tính
- Thiết lập các mối quan hệ giữa các dữ liệu

❖ Nguyên lý Phân tích 2: *Mô hình hóa Chức năng*

Bản chất của phần mềm là biến đổi thông tin, nên ta cần:

- Định danh các chức năng (biến đổi thông tin)
- Xác định cách thức dữ liệu (thông tin) di chuyển trong hệ thống

- Xác định các tác nhân tạo dữ liệu và tác nhân tiêu thụ dữ liệu

❖ **Nguyên lý Phân tích 3: Mô hình hóa Hành vi**

Phần mềm (hệ thống) có trạng thái (hành vi), nên ta cần:

- Xác định các trạng thái hệ thống. Ví dụ: giao diện đồ họa
- Xác định các dữ liệu làm thay đổi hành vi hệ thống. Ví dụ: bàn phím, chuột, các cổng thông tin ...

❖ **Nguyên lý Phân tích 4: Phân hoạch Mô hình**

Ta cần làm mịn, phân hoạch, biểu diễn mô hình ở các mức khác nhau:

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

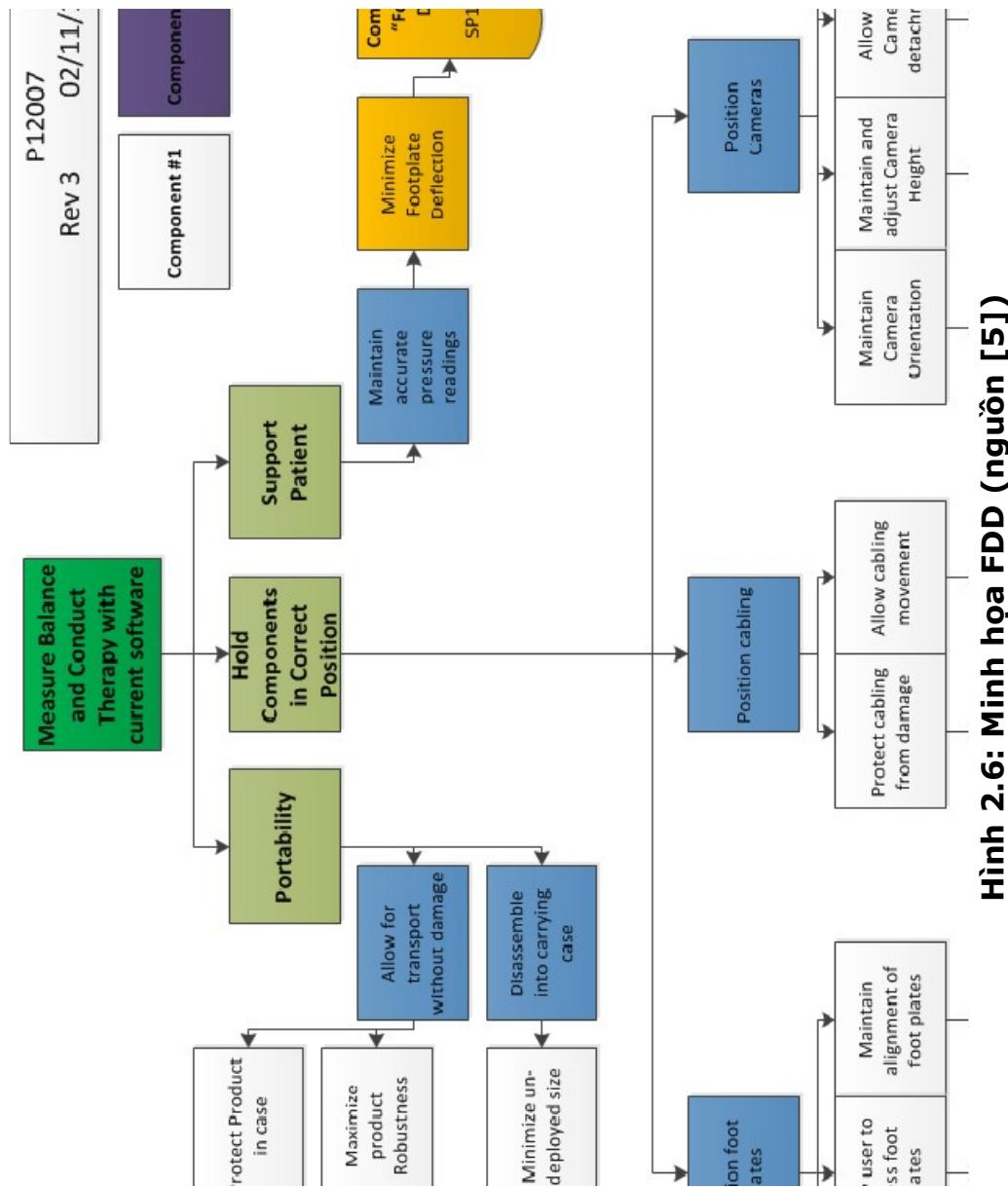
❖ **Nguyên lý Phân tích 5: Tìm hiểu Vấn đề bản chất**

Ta chỉ xét bản chất của yêu cầu và không quan tâm đến cách cài đặt.

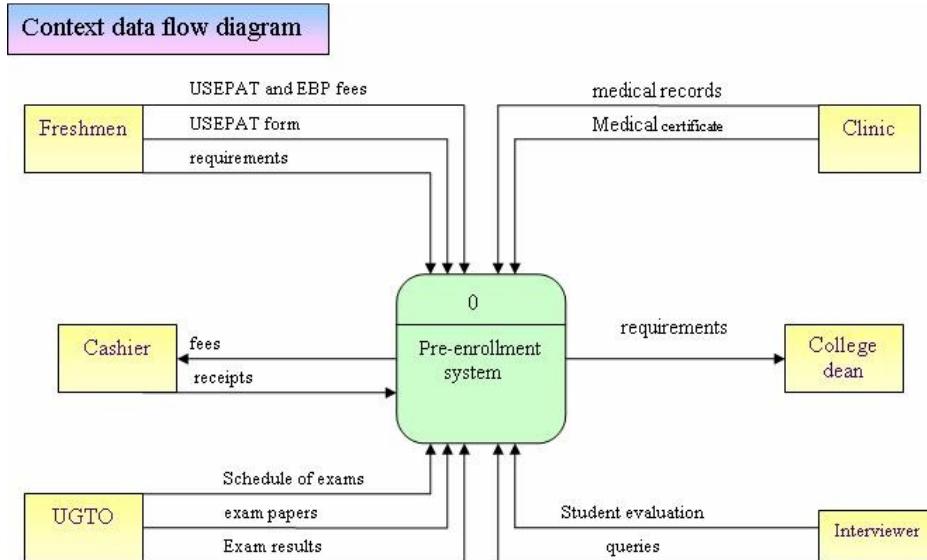
2.4.2 Sơ đồ phân rã chức năng (FDD)

Sơ đồ này (*FDD, Function Decomposition Diagram*) cho ta biểu diễn các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra, nhằm: (i) Giúp xác định được phạm vi của hệ thống, (ii) Giúp phân hoạch được hệ thống chức năng, (iii) Giúp tạo nền tảng cho thiết kế kiến trúc hệ thống

Ví dụ: Sơ đồ phân rã chức năng



Hình 2.6: Minh họa FDD (nguồn [5])



Hình 2.7: Minh họa sơ đồ ngữ cảnh gồm các chức năng

2.4.3 Mô hình bản mẫu (Prototype)

Khi xác định yêu cầu, nhóm phát triển phần mềm dựa trên các ý tưởng hay yêu cầu của khách hàng để đưa ra bản thiết kế sơ bộ với các màn hình giao diện, từ đó tiến hành mô phỏng hay giả lập sơ bộ một số chức năng. Đây là bước cài đặt bản mẫu đầu tiên để chuyển cho người sử dụng. Bản mẫu này chỉ nhằm để mô tả cách thức phần mềm hoạt động cũng như cách người dùng tương tác với hệ thống và giúp người dùng có thể hình dung được giao diện ban đầu của yêu cầu mà họ đặt ra. Mô hình này cũng cần có sự hỗ trợ giữa kỹ sư phân tích và kỹ sư thiết kế phần mềm khi phối hợp thực hiện.

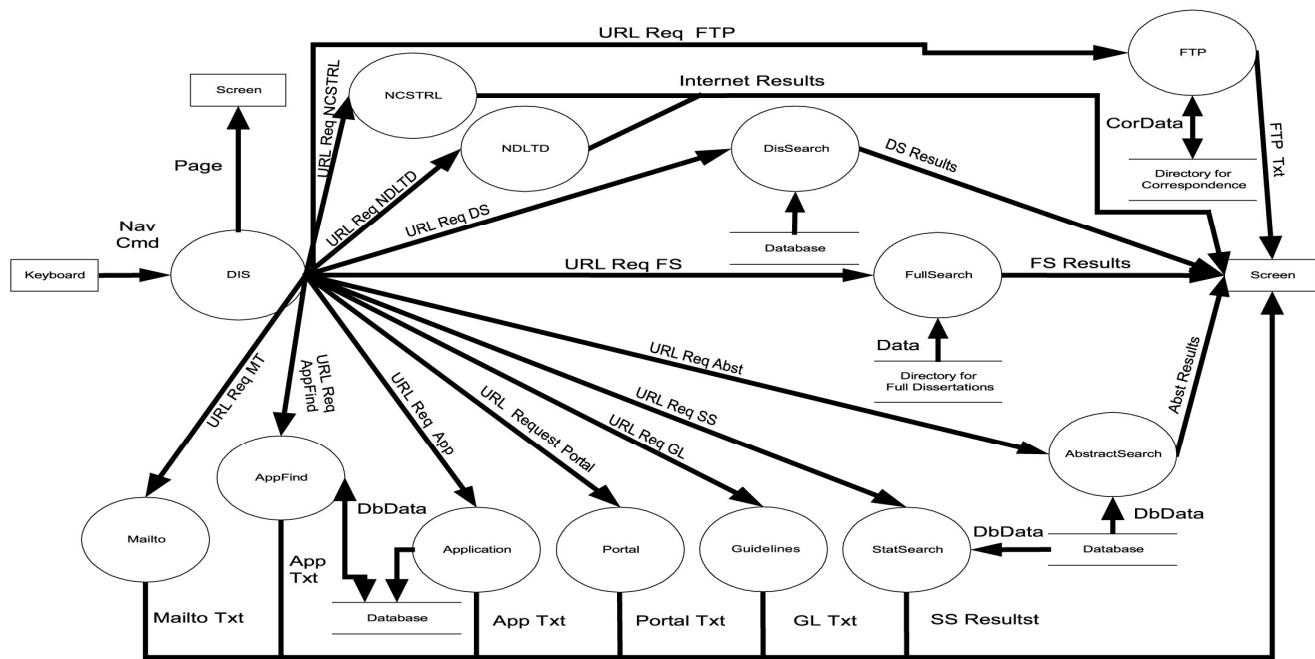
Người sử dụng khi xem xét bản mẫu sẽ đưa ra ý kiến đóng góp và phản hồi thông tin đồng ý hay không trên phương án thiết kế của bản mẫu hiện có.

- Nếu người sử dụng đồng ý với bản mẫu đã đưa thì người phát triển sẽ tiến hành cài đặt thực sự.
- Ngược lại cả hai phải quay lại giai đoạn xác định yêu cầu.

Công việc này được lặp lại liên tục cho đến khi người sử dụng đồng ý với các bản mẫu do nhà phát triển đưa ra.

2.4.4 Sơ đồ luồng dữ liệu (Data Flow Diagram, DFD)

Đây là mô hình cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống và cách thức dữ liệu được xử lý bên trong hệ thống với nhiều mức chi tiết khác nhau. Sơ đồ này có nhiều biến thể mở rộng khác nhau. (*tham khảo thêm tài liệu về Phân tích Thiết kế Hệ thống Thông tin*).



Hình 2.8: Minh họa sơ đồ DFD (nguồn [5])

2.4.5 Mô hình hướng đối tượng

Phương pháp phân tích hướng đối tượng (*Object Oriented Analysis, OOA*) hình thành giữa thập niên 80 dựa trên ý tưởng lập trình hướng đối tượng. Phương pháp này đã hoàn thiện và nay rất phổ dụng, dựa trên các khái niệm:

- **Đối tượng (Object):** gồm dữ liệu và thủ tục tác động lên dữ liệu này.
- **Đóng gói (Encapsulation):** khả năng hạn chế tác động trực tiếp lên dữ liệu của đối tượng mà phải thông qua các phương pháp trung gian.
- **Lớp (Class):** là nhóm các đối tượng có chung một cấu trúc dữ liệu và cùng một phương pháp.
- **Kế thừa (Heritage):** tính chất kế thừa là đặc tính cho phép định nghĩa một lớp mới từ các lớp đã có bằng cách thêm vào đó những dữ liệu mới, các phương pháp mới có thể kế thừa những đặc tính của lớp cũ.

❖ Mô hình nắm bắt yêu cầu hướng đối tượng bằng UML

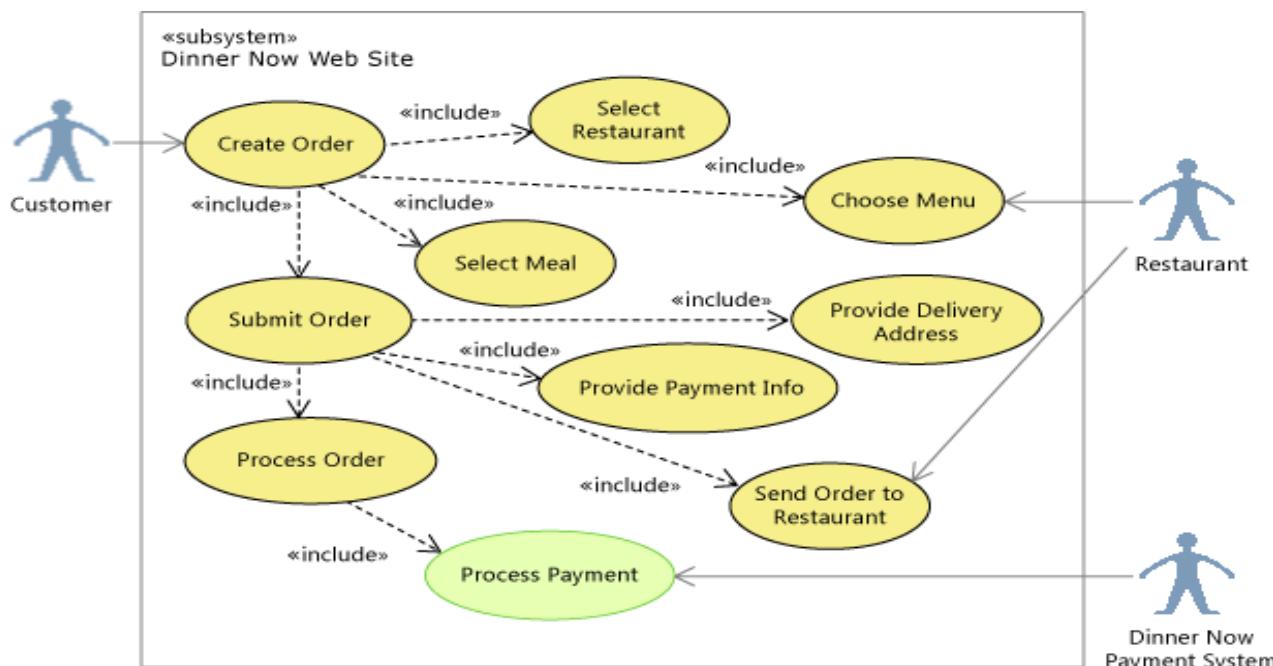
Mục đích của hoạt động nắm bắt yêu cầu là xây dựng mô hình hệ thống bằng cách sử dụng các ca sử dụng (*use case*). Các điểm bắt đầu cho hoạt động này khá đa dạng:

- Từ mô hình nghiệp vụ (*business model*) cho các phần mềm nghiệp vụ.
- Từ mô hình lĩnh vực (*domain model*) cho các phần mềm nhúng.
- Từ đặc tả yêu cầu của hệ thống nhúng được tạo bởi nhóm khác và hoặc dùng các phương pháp đặc tả khác (thí dụ hướng cấu trúc).
- Từ điểm nào đó nằm giữa các điểm xuất phát trên.

Mô hình ca sử dụng (Use Case Diagram)

Mô hình này gồm các thành phần:

- *Actor*: người/ hệ thống ngoài/ thiết bị ngoài tương tác với hệ thống
- *Use-case*: các chức năng có nghĩa của hệ thống cung cấp cho các actor (i) Luồng các sự kiện (flow of events), (ii) Các yêu cầu đặc biệt của use-case
- *Đặc tả kiến trúc*
- Các thiết kế mẫu giao diện người dùng



Hình 2.9: Minh họa sơ đồ ca sử dụng (nguồn [5])

❖ Mô hình phân tích hướng đối tượng với UML

Mục đích của hoạt động phân tích yêu cầu là xây dựng mô hình phân tích với các đặc điểm sau:

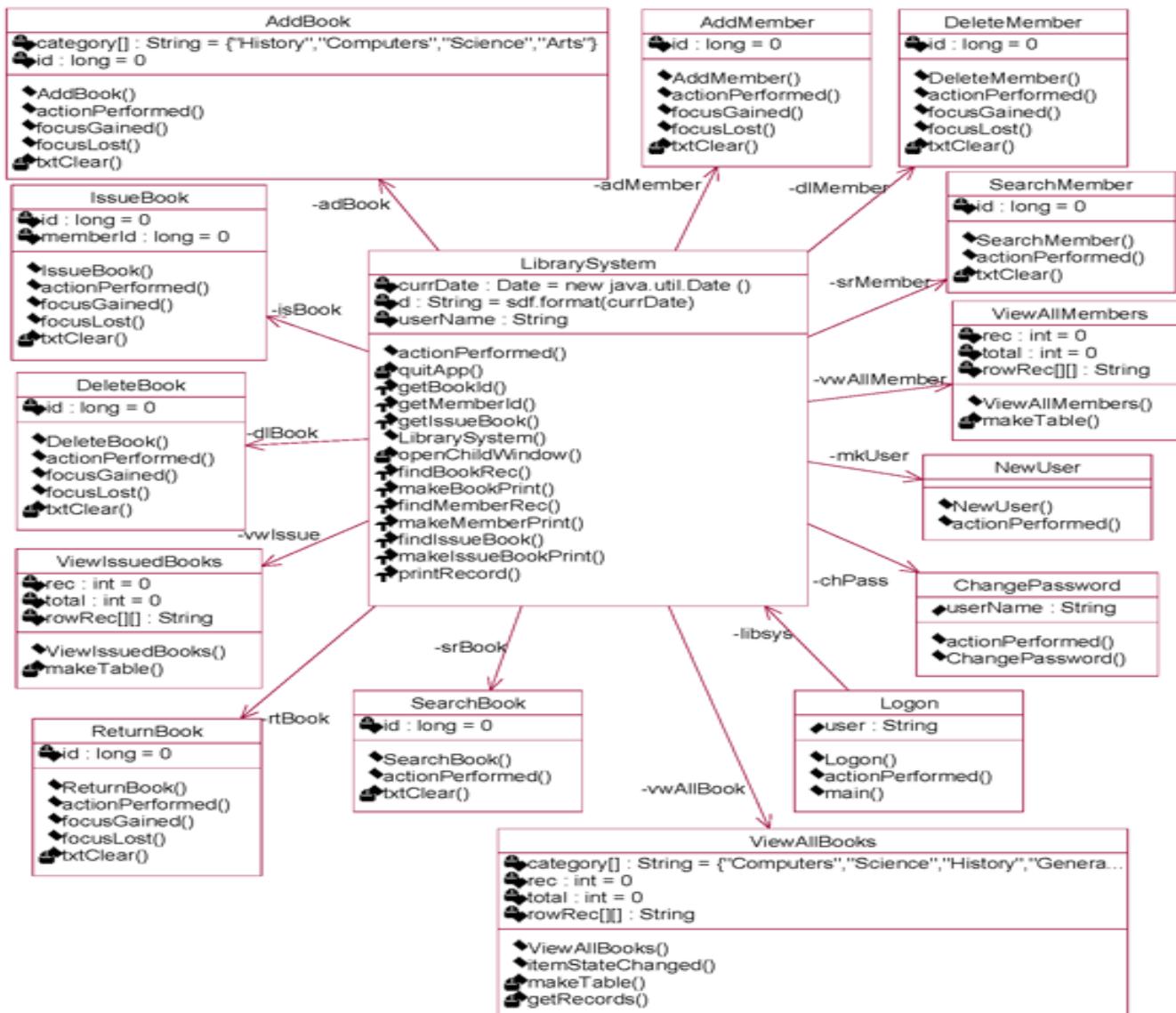
- Dùng ngôn ngữ của nhà phát triển để miêu tả mô hình
- Thể hiện gốc nhìn từ bên trong hệ thống
- Được cấu trúc từ các lớp phân tích và các package phân tích
- Được dùng chủ yếu cho nhà phát triển để hiểu cách tạo hình hệ thống
- Loại trừ mọi chi tiết dư thừa, không nhất quán
- Phát họa hiện thực các chất năng bên trong hệ thống
- Định nghĩa các dẫn xuất use-case, mỗi dẫn xuất use-case cấp phân tích miêu tả sự phân tích 1 use-case

Mô hình phân tích

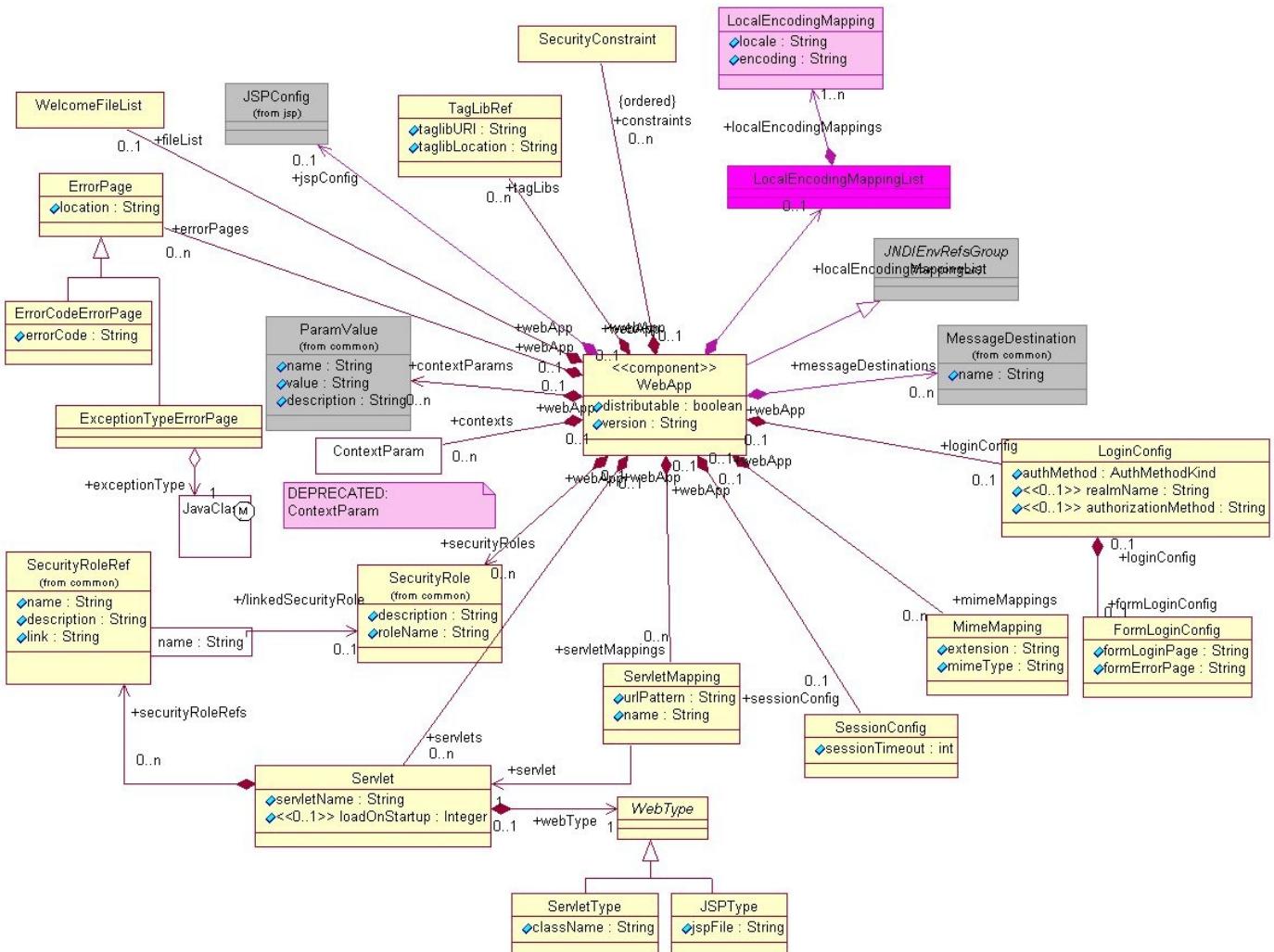
Mô hình này bao gồm:

- Các lớp (*class*) phân tích: lớp biên, lớp thực thể, lớp điều khiển
- Các dẫn xuất use case cấp phân tích: các lược đồ lớp phân tích, các lược đồ tương tác, luồng sự kiện, các yêu cầu đặc biệt của use-case
- Các gói (*package*) phân tích
- Đặc tả kiến trúc.

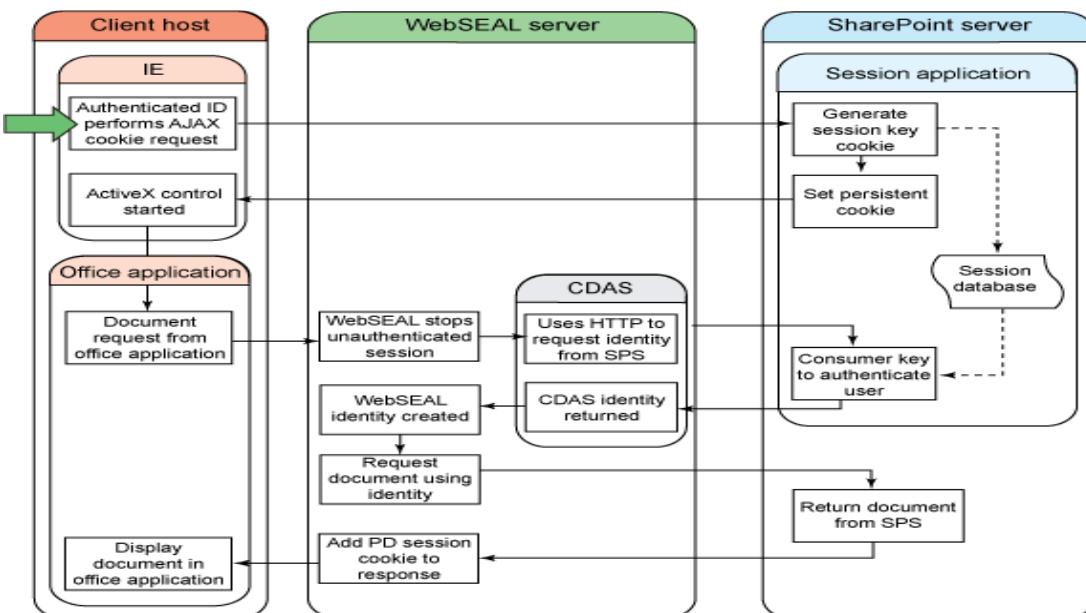
Ta cần tham khảo tài liệu môn Phân tích thiết kế hướng đối tượng cho các dạng mô hình hướng đối tượng cho từng giai đoạn phát triển phần mềm.



Hình 2.10: Minh họa sơ đồ lớp đối tượng mức phân tích (nguồn [5])



Hình 2.11: Minh họa sơ đồ lớp đối tượng mức thiết kế (nguồn [5])



Hình 2.12: Minh họa về thiết kế các thành phần (nguồn [5])

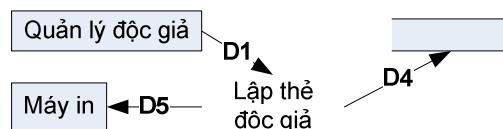
2.4.6 Ví dụ minh họa từ yêu cầu sang mô hình hóa

Ví dụ 1: Xét phần mềm quản lý thư viện với 4 yêu cầu: Lập thẻ độc giả, Nhận sách, Cho mượn sách, Trả sách.

❖ Giai đoạn 2: Mô hình hóa yêu cầu

Sơ đồ luồng dữ liệu cho công việc lập thẻ độc giả

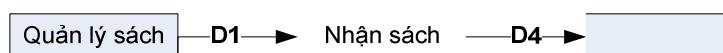
- D1: Thông tin về thẻ độc giả cần nhập
- D4: Thông tin về thẻ độc giả cần lưu trữ trên kho
- D5: Thông tin trên thẻ độc giả (trong thế giới thực)
- Xử lý thẻ độc giả: Kiểm tra tính hợp lệ của thẻ trước ghi nhận và in



Hình 2.13: Sơ đồ luồng dữ liệu cho công việc lập thẻ độc giả

Sơ đồ luồng dữ liệu cho công việc nhận sách

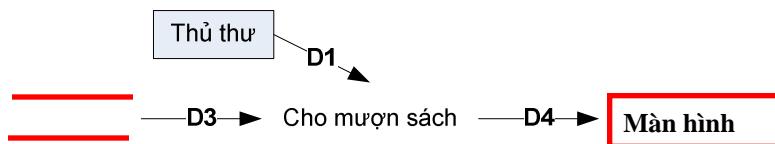
- D1: Thông tin về thẻ sách cần nhập
- D4: Thông tin về sách cần lưu trữ trên kho
- Xử lý nhận sách: Kiểm tra tính hợp lệ của sách trước khi lưu vào kho



Hình 2.14: Sơ đồ luồng dữ liệu cho công việc nhận sách

Sơ đồ luồng dữ liệu cho công việc cho mượn sách

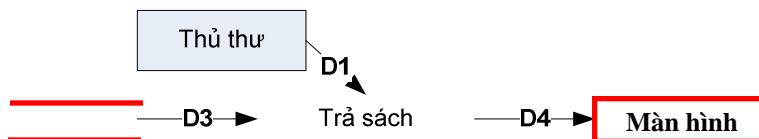
- D1: Thông tin về độc giả và sách muốn mượn
- D3: Thông tin được sử dụng cho việc kiểm tra quy định mượn sách
- D4: Thông tin về việc mượn sách
- Xử lý cho mượn sách: Kiểm tra tính hợp lệ của việc mượn, lưu vào kho



Hình 2.15: Sơ đồ luồng dữ liệu cho công việc cho mượn sách

Sơ đồ luồng dữ liệu cho công việc trả sách

- D1: Thông tin về độc giả và sách trả
- D3: Thông tin sử dụng cho việc kiểm tra quy định trả sách
- D4: Thông tin về việc trả sách
- Xử lý trả sách: Kiểm tra tính hợp lệ của việc trả sách, lưu vào kho.



Hình 2.16: Sơ đồ luồng dữ liệu cho công việc trả sách

TÓM TẮT

Bài học này trình bày những khái niệm về Công nghệ Phần mềm như:

- Quá trình Phân tích (Phạm vi dự án, Mở rộng Yêu cầu Nghiệp vụ, Yêu cầu Bảo mật, Yêu cầu Tốc độ, Yêu cầu Vận hành, Khả năng Mở rộng Yêu cầu, Yêu cầu Sẵn dùng, Yếu tố Con người, Yêu cầu Tích hợp, Thực tiễn Nghiệp vụ tồn tại, Khả năng và Quy mô)
- Xác định Yêu cầu (Mô tả yêu cầu, Phân loại yêu cầu, bước xác định)
- Các nguyên lý Mô hình hóa
- Sơ đồ phân rã chức năng (Function Decomposition Diagram, FDD)
- Mô hình bản mẫu (Prototype)
- Sơ đồ luồng dữ liệu (Data Flow Diagram, DFD)
- Mô hình hướng đối tượng

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 3: TÁC VỤ THIẾT KẾ PHẦN MỀM

Học xong bài này người học sẽ:

- *Hiểu được các khái niệm cơ bản thiết kế, kiến trúc phần mềm, phương pháp thiết kế phần mềm.*
- *Vận dụng được các kỹ thuật thiết kế (top-down, bottom-up ...) và phương pháp thiết kế (hướng chức năng, hướng đối tượng ...) để phát triển kiến trúc của phần mềm.*

3.1 TỔNG QUAN VỀ THIẾT KẾ

Mục tiêu của việc thiết kế là định hình hệ thống và tìm dạng thức của phần mềm (kể cả kiến trúc) để đáp ứng được mọi yêu cầu, kể cả yêu cầu phi chức năng và các ràng buộc khác, được đặt ra cho hệ thống đó. Kết quả thu được từ bước phân tích trước đó (là mô hình phân tích) chính là thông tin đầu vào (*input*) quan trọng cho công tác thiết kế. (Xem thêm Phụ lục C – Phần C)

Mục đích quan trọng của công tác thiết kế là:

- Hiểu biết sâu sắc về yêu cầu phi chức năng và những ràng buộc có liên quan tới ngôn ngữ lập trình, mức độ tái sử dụng của các thành phần, các thông tin kỹ thuật chuyên sâu (hệ điều hành, công nghệ phân tán, công nghệ cơ sở dữ liệu, công nghệ giao diện người dùng, công nghệ quản lý các giao dịch ...).
- Tạo ra một đầu vào thích hợp và điểm xuất phát cho các hoạt động hiện thực tiếp theo sau thông qua việc nắm bắt các yêu cầu về mỗi hệ thống cụ thể, các giao diện và các lớp.
- Có khả năng phân rã việc cài đặt thành các phần nhỏ dễ quản lý hơn để nhiều nhóm phát triển khác nhau có thể xử lý đồng thời. Điều này sẽ có ích khi mà ta không thể tiến hành sự phân rã giữa các kết quả thu được từ bước nắm bắt các yêu cầu hoặc phân tích.

- Nắm bắt sớm các giao diện cốt lõi tương tác giữa các hệ thống con trong vòng đời của phần mềm. Điều này sẽ có ích khi ta suy luận về kiến trúc và khi ta sử dụng các giao diện như những công cụ đồng bộ các nhóm phát triển khác nhau.
- Trực quan hóa và suy luận thiết kế bởi hệ thống ký pháp chung.
- Tạo ra một sự trừu tượng hóa liên tục cho việc hiện thực của hệ thống, tức là cài đặt sự làm mịn dần thiết kế bằng cách chi tiết hóa nó mà không thay đổi cấu trúc của nó.

Mục tiêu của bài học này là giới thiệu một số phương pháp và kỹ thuật chính trong công tác thiết kế, đối với việc triển khai một hệ thống thành nhiều hệ thống con và hệ thống con thành nhiều thành phần và quản lý những vấn đề liên quan đến cấu trúc nội tại của những thành phần hệ thống.

3.1.1 Kỹ thuật thiết kế

- Thiết kế đặc tả đi đến kỹ thuật cốt lõi của tiến trình của công nghệ phần mềm và được cung cấp xem xét những mô hình của tiến trình phần mềm được sử dụng.
- Thiết kế phần mềm là bước đầu tiên trong ba hoạt động kỹ thuật - thiết kế, phát sinh mã nguồn, và thử nghiệm – đó là những yêu cầu trong xây dựng và phát triển phần mềm.

Một trong những điểm mấu chốt chính đối với độ phức tạp của hệ thống phần mềm là sự trừu tượng. Có hai phương pháp chính:

3.1.1.1 Thiết kế trên xuống (Top-down)

- Thiết kế bắt đầu với việc phân tích những định nghĩa yêu cầu và không nên xem xét việc thực hiện chi tiết đầu tiên.
- Một dự án được triển khai thành những dự án nhỏ, thủ tục này phải được lặp lại cho đến khi những nhiệm vụ con trở nên đơn giản sao cho một thuật toán được tính toán và giải quyết.

3.1.1.2 Thiết kế từ dưới lên (Bottom-up)

- Ý tưởng nền tảng: Hiểu được phần cứng và tầng trên của nó như một cơ chế trừu tượng.
- Kỹ thuật: Thiết kế từ dưới lên bắt đầu được cho bởi máy cụ thể và liên tiếp phát triển một máy trừu tượng sau khi những máy khác được thêm vào những thuộc tính cần thiết cho đến khi một máy đã đạt được kết quả mà cung cấp những chức năng người dùng yêu cầu.

3.1.1.3 Thiết kế Hệ thống

Trong hệ thống lớn, tiến trình thiết kế bao gồm một số yếu tố thiết kế cho hệ thống, trong đó mỗi chức năng được phân chia thành những chức năng phần mềm và phần cứng.

Những thuận lợi của chức năng thực hiện trong phần cứng là thành phần phần cứng phân phối thực hiện tốt hơn đơn vị phần cứng. Nút thắt của hệ thống được xác định và thay thế bởi thành phần của phần cứng, như thế việc tối ưu phần mềm là hết sức tốn kém. Ngoài ra, việc cung cấp tốc độ phần cứng có nghĩa là thiết kế phần mềm có thể được cấu trúc cho khả năng thích ứng và khả năng xem xét thực thi cả chức năng.

3.1.1.4 Thiết kế Bản mẫu (Prototype)

Thiết kế bản mẫu là tạo ra các màn hình giao diện sơ bộ, hay các bản thiết kế phác thảo nháp cho người dùng tham khảo trước khi đi vào thiết kế chi tiết cho toàn phần mềm hay cho chức năng cụ thể. Các bản thiết kế này được soạn thảo dưới dạng tài liệu kỹ thuật (tài liệu sưu tập, hay tài liệu kỹ thuật) bằng một số phần mềm có khả năng thiết kế nhanh giao diện, như MS Visio, MS Visual Basic / C# / C++, MS Front Page / Visual Interdev ... Đây có thể là bước đệm cơ bản trước khi đi vào hiện thực chi tiết cho từng chương trình con hay mô-đun con ...

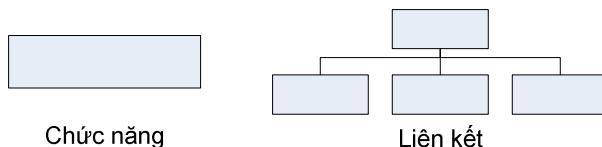
3.1.1.5 Phân rã Thiết kế

Tiến trình thiết kế không chỉ ảnh hưởng đến phương pháp thiết kế mà còn ảnh hưởng đến tiêu chuẩn được sử dụng để phân rã hệ thống. Việc phân rã giúp

hiện thực hóa từng phần bản thiết kế đến mức chi tiết đồng thời tác động đến phương pháp thiết kế. Các nhóm phương pháp phân rã gồm:

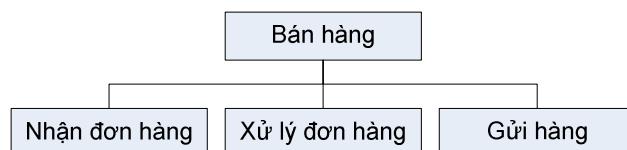
❖ Phân rã hướng chức năng

- Ta dựa trên những yêu cầu chức năng có trong các định nghĩa yêu cầu để phân rã hướng đến các tác nhiệm của toàn bộ hệ thống.
- Ta sử dụng *Sơ đồ phân rã chức năng (FDD)* để nêu lên các chức năng thông qua việc mô tả các tính chất của đầu vào và đầu ra, đồng thời:
 - Xác định phạm vi của hệ thống
 - Phân hoạch chức năng
 - Tạo nền tảng cho thiết kế kiến trúc hệ thống



Hình 3.1: Sơ đồ minh họa tổ chức hệ thống

Ví dụ: Sơ đồ phân rã chức năng



Hình 3.2: Minh họa sơ đồ chức năng

❖ Phân rã hướng dữ liệu

Tiến trình thiết kế tập trung trung khía cạnh hệ thống hướng đến dữ liệu. Chiến lược thiết kế hướng đến chính các đối tượng dữ liệu cần được thực hiện. Việc phân rã hệ thống dựa trên việc phân tích dữ liệu, bao gồm các phần:

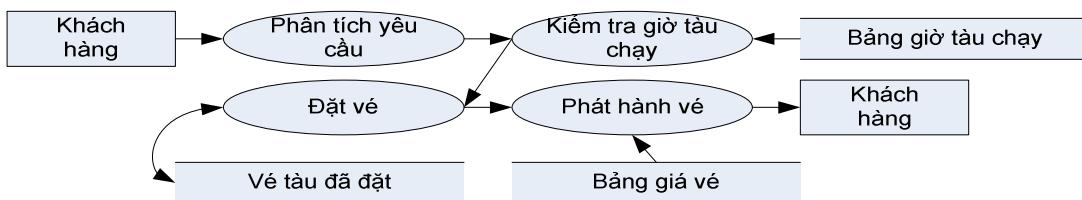
Sơ đồ luồng dữ liệu (Data flow diagram - DFD)

DFD cho phép xem toàn bộ sơ đồ luồng dữ liệu bên trong hệ thống với cách thức dữ liệu được xử lý bên trong hệ thống dựa theo nhiều mức chi tiết khác nhau và nhiều biến thể mở rộng khác nhau (*tham khảo tài liệu Phân tích thiết kế hệ thống thông tin*).

Ví dụ: DFD hệ thống bán vé



Hình 3.3: DFD mức 0



Hình 3.4: DFD mức 1

Các hướng tiếp cận chính để tạo lập sơ đồ luồng dữ liệu gồm:

Tiếp cận từ trên xuống (top-down)

Quá trình thực hiện theo hướng tiếp cận này như sau:

- Lập sơ đồ luồng dữ liệu cấp 0 (xem xét tất cả các luồng dữ liệu nhập xuất, tất cả các yêu cầu xử lý của phần mềm)
- Phân rã sơ đồ luồng dữ liệu cấp 0 thành các sơ đồ luồng dữ liệu cấp 1:
 - Phân rã các xử lý của phần mềm thành nhiều xử lý con và quyết định các luồng dữ liệu tương ứng trên các xử lý con này.
 - Phân rã các luồng dữ liệu nhập xuất thành nhiều luồng dữ liệu con và quyết định các xử lý tương ứng với những luồng con này.
- Quá trình kết thúc khi đạt đến các sơ đồ không thể tiếp tục phân rã được (sơ đồ lá).

Thông thường đây là sơ đồ tương ứng với công việc cụ thể của một chuyên gia trong thế giới thực.

Nhận xét: Cách tiếp cận này:

- Thích hợp với phần mềm có số lượng người dùng, số lượng các yêu cầu ít (nếu ngược lại sơ đồ cấp 0 sẽ rất phức tạp và khó lập chính xác).

- Đặc biệt thích hợp với các loại phần mềm mà vì lý do nào đó các hệ thống yêu cầu chưa được xác định rõ ngay từ đầu (ví dụ các phần mềm hệ thống). Và ít được sử dụng.

Hướng tiếp cận từ dưới lên (bottom-up)

Quá trình thực hiện theo hướng tiếp cận này như sau

- Lập sơ đồ luồng dữ liệu ở mức cao nhất. Các sơ đồ này không được tiến hành phân rã thành các sơ đồ có cấp lớn hơn (thường đây là sơ đồ ứng với công việc cụ thể của một người dùng trong thế giới thực)
- Tích hợp các sơ đồ này để tạo các sơ đồ có cấp nhỏ hơn (thông thường các sơ đồ được chọn tích hợp theo một tiêu chí cụ thể: cùng một người sử dụng, cùng một loại yêu cầu ...) theo cách:
 - Tích hợp các xử lý của các sơ đồ cấp k vào sơ đồ cấp k-1 và giữ nguyên các luồng dữ liệu của các sơ đồ cấp k
 - Tích hợp đồng thời các xử lý và các luồng dữ liệu của các sơ đồ cấp k để tạo lập sơ đồ cấp k-1.
 - Quá trình kết thúc khi đạt đến các sơ đồ cấp 0

Nhận xét: Cách tiếp cận này:

- Thích hợp với các phần mềm có hệ thống yêu cầu chi tiết, cụ thể và có quy mô yêu cầu (số lượng người dùng, số lượng yêu cầu) thuộc mức trung bình.
- Khó thực hiện nếu quy mô yêu cầu lớn và chưa thật rõ ràng chi tiết

Hướng tiếp cận phối hợp

Quá trình thực hiện theo hướng tiếp cận này như sau:

- Lập sơ đồ luồng dữ liệu cấp k theo một tiêu chí xác định (sơ đồ cho từng người dùng hay bộ phận, sơ đồ cho một loại yêu cầu ...).
- Phân rã sơ đồ cấp k thành nhiều sơ đồ cấp k+1 tiếp tục cho đến khi đạt được các sơ đồ lá.
- Tích hợp các sơ đồ cấp k thành các sơ đồ cấp k-1 tiếp tục cho đến khi đạt được sơ đồ cấp 0.

Nhận xét: Cách tiếp cận này:

- Thích hợp cho các phần mềm có quy mô yêu cầu lớn, phức tạp
- Được sử dụng rất thường xuyên trong thực tế.

Lập sơ đồ luồng dữ liệu cho từng công việc

Do các giới hạn đã nêu trên, việc lập các sơ đồ luồng dữ liệu cho toàn bộ phần mềm chỉ quy về việc lập sơ đồ luồng dữ liệu cho từng công việc (sau đó chỉ thực hiện đơn giản một bước tích hợp để có sơ đồ cấp 0)

Quá trình lập sơ đồ luồng dữ liệu cho một công việc được tiến hành qua các bước như sau

Bước 1: Xác định dữ liệu nhập

Dữ liệu nhập từ người dùng sử dụng được xác định dựa vào biểu mẫu có liên quan với các lưu ý sau:

- Không nhập vào các dữ liệu có thể tính toán được dựa trên quy định hay công thức đã có.
- Không nhập vào các dữ liệu đã được lưu trữ trước đó (qua việc khác).

Dữ liệu nhập từ thiết bị nhập (khác với bàn phím) chỉ được xem xét khi có yêu cầu đặc biệt trong một số phần mềm đặc biệt (hệ thống thời gian thực, hệ thống bản đồ, nhập thông qua sử dụng điện thoại tổng đài điện thoại trong quản lý khách sạn ...).

Dữ liệu nhập (đọc) từ bộ nhớ phụ được xác định dựa trên các quy định công thức liên quan với một số lưu ý:

- Chỉ đọc dữ liệu thật sự cần thiết cho việc thực hiện xử lý tương ứng (thông tin nhập chưa đủ để xử lý).
- Có thể đọc thêm các tham số phục vụ cho việc xử lý từ bộ nhớ phụ để cải tiến chất lượng phần mềm (đặc biệt là tính tiến hóa). Tuy nhiên trong giai đoạn này chỉ nên tập trung vào tính đúng đắn (các chất lượng khác được xem xét chi tiết trong giai đoạn thiết kế).

Bước 2: Xác định dữ liệu xuất

Dữ liệu xuất cho người dùng được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau:

- Các thông báo (về việc xử lý có thực hiện được hay không) luôn phải có và không cần thiết thể hiện trên sơ đồ (ví dụ thông báo việc mượn sách là không hợp lệ ...)
- Để tăng tính tiện dụng, trong tất cả các xử lý (kể cả xử lý lưu trữ, xử lý tính toán) ta đều phải xuất cho người dùng nhiều thông tin. *Tuy nhiên vấn đề này chỉ xem xét và thực hiện trong các giai đoạn sau, nếu chú ý quá sớm đến vấn đề này sẽ làm phức tạp sơ đồ và dễ phạm các sai lầm trong tính đúng đắn.*

Các dữ liệu được gửi ra các thiết bị xuất (khác với màn hình) thông thường là máy in. Để tăng tính tiện dụng ta có thể tuân theo nguyên tắc sau “*Tất cả dữ liệu xuất ra màn hình đều cho phép người dùng xuất ra máy in*” (có thể với cách trình bày khác). Tuy nhiên vấn đề này cũng có thể dời lại xem xét chi tiết trong giai đoạn thiết kế. Các loại thiết bị xuất khác chỉ có trong các loại phần mềm đặc biệt hoặc do yêu cầu tính tương thích.

Dữ liệu xuất (ghi) vào bộ nhớ phụ được xác định dựa trên biểu mẫu liên quan với một số lưu ý như sau:

- Ghi các dữ liệu kết quả mới tạo lập hoặc các dữ liệu đã có nhưng bị thay đổi trong quá trình thực hiện xử lý.
- Để tăng tính hiệu quả, ta có thể ghi các thông tin bổ sung có liên quan đến các yêu cầu khác. Tuy nhiên cách tốt nhất là vấn đề này cần được xem xét chi tiết trong giai đoạn thiết kế.

Bước 3: Mô tả xử lý

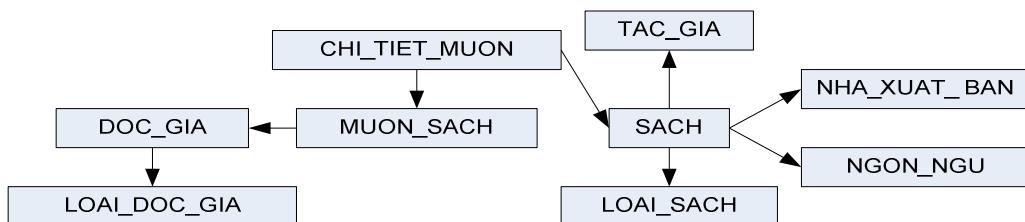
Khi mô tả quá trình xử lý dữ liệu nhập để tạo dữ liệu xuất, ta cần chú ý:

- Chỉ mô tả cách xử lý mà không cần chú ý đến cách thực hiện nhập xuất (hình thức nhập, lưu trữ trên bộ nhớ phụ, câu lệnh đọc, ghi ...).
- Mô tả chi tiết cách sử dụng dữ liệu nhập để tạo dữ liệu xuất, càng chi tiết thì việc thiết kế xử lý càng dễ dàng.

- Chỉ chú trọng đến tính đúng đắn mà không nên xem xét quá sớm các yêu cầu chất lượng khác.
- Mô tả chính xác thứ tự nhập/xuất (có thể đổi trong một số trường hợp).

Xây dựng mô hình thực thể liên kết (Entity Relation Diagram, ERD)

Mô hình ERD (như ví dụ trong hình sau) là dạng sơ đồ giúp thể hiện các đối tượng dữ liệu được đặc tả trong yêu cầu của phần mềm, tạo nền tảng cho việc thiết kế chi tiết cơ sở dữ liệu cho phần mềm (*xem thêm tài liệu Phân tích Thiết kế Hệ thống Thông tin*).



Hình 3.5: Biểu diễn các ký hiệu sử dụng trong ERD

❖ Phân rã hướng đối tượng

Tính chất hướng đối tượng của hệ thống được tập trung chủ yếu trong bản thiết kế với các thành phần được thể hiện thông qua những đối tượng khác nhau. Theo cách này, một hệ thống phần mềm được xem xét như tập hợp các đối tượng thông tin, mỗi đối tượng có cấu trúc dữ liệu ẩn và thao tác của chúng có thể được thực hiện trên cấu trúc này. Những điểm cơ bản của phân rã hướng đối tượng chính là nó hướng đến tính đồng nhất giữa dữ liệu và thao tác và dựa trên sự che dấu thông tin và dẫn xuất kẽ thửa.

3.1.2 Thiết kế giao diện người dùng

Thiết kế giao diện người dùng là một tác nhiệm trong giai đoạn thiết kế. Thiết kế giao diện được hỗ trợ một phần trong thiết kế dạng mô hình bản mẫu (prototype) nhằm làm sáng tỏ các yêu cầu từ người dùng, xác định đúng yêu cầu người dùng và đáp ứng các đòi hỏi về mặt thẩm mỹ, giao diện đẹp cho khách hàng. Nếu khách hàng đã đồng ý với bản mẫu đã đưa ra trong giai đoạn xác định yêu cầu, kỹ sư thiết kế sẽ hoàn chỉnh thêm giao diện để đảm bảo tính tiện dụng, đảm bảo chính xác yêu cầu người dùng, ngược lại cần sáng tạo thêm theo các tiêu chí về thẩm mỹ, tiện dụng và đầy đủ yêu cầu thông tin, liên quan đến các thành phần sau:

❖ Chế độ (*modes*)

Chế độ chương trình là trường hợp mà người dùng chỉ có thể thực hiện ở một số thao tác giới hạn. Kỹ thuật tạo/sử dụng cửa sổ có thể cung cấp dịch vụ có giá trị về biểu diễn chế độ chương trình, đồng thời trợ giúp người thực hiện các thao tác con tương ứng trong những cửa sổ khác nhau thể hiện bởi những chế độ chương trình khác nhau.

Ví dụ: phần mềm có chế độ chương trình dạng giao diện hội thoại (dialog) như MS Calculator, phần mềm đơn tài liệu (single document interface) như Notepad, đa tài liệu (multiple document interface) như MS Word.

❖ Thực đơn (*menu*)

Người dùng chọn vào thực đơn bằng con trỏ chuột để hiển thị tất cả lệnh và kích hoạt thao tác trên đó.

Có hai dạng thực đơn:

- Dạng *thực đơn bấm mở ra* (*Pop-up menu*): là dạng được thiết kế hiệu quả để xuất hiện được ở vị trí bất kỳ.
- Dạng *thực đơn bấm thả xuống* (*Pull-down menu*): là dạng cho phép cấu trúc tốt hơn việc mở rộng tập lệnh và dễ dàng sử dụng.

Ta có thể phân loại menu theo tập lệnh thao tác, tập lệnh thao tác với tham số, tập lệnh chuyển đổi chế độ người dùng.

❖ Cửa sổ hội thoại (*dialog window*)

Cửa sổ hội thoại là một dạng giao diện đơn giản chương trình giúp người dùng tương tác linh hoạt hơn.

Khi thiết kế cửa sổ hội thoại, chúng ta cần đảm bảo tính đồng nhất trong giao diện người dùng, tránh những giải thích dài dòng nên ngắn gọn cô động như cách đặt nhãn Label, Checkbox, Button, List box.

❖ Màu sắc (*color*)

Màu sắc chủ yếu chỉ được chúng ta dùng ở những nơi cần diễn đạt những yêu cầu nào đó, hay khi muốn nhấn mạnh ý nghĩa nào đó, hoặc dấu hiệu cảnh báo nguy hiểm, nhưng cần hạn chế lòe loẹt quá cho giao diện. Ví dụ màu chữ đen trên nền trắng

thường dễ đọc nhất cho khả năng làm việc hàng ngày, còn màu chữ trắng trên nền xanh thì khó đọc ...

❖ Âm Thanh (*sound*)

Âm thanh là cách tốt nhất tập trung sự chú ý của người dùng. Chúng được phần mềm phù hợp trong các tình huống xử lý lỗi, sự kiện không chắc chắn, tạm thời. Chúng ta nên tạo những âm thanh khác nhau với những sự kiện khác nhau, tránh dùng âm thanh gây ồn.

❖ Tính kiên định

Khi thiết kế giao diện, chúng ta cần lưu ý:

- Thực đơn lệnh với những chức năng giống nhau nên vị trí giống nhau (thậm chí ở những chương trình khác nhau).
- Phím nóng nên được phần mềm cho thực đơn lệnh và nên cố định.
- Nút lệnh với những chức năng tương tự (trong những cửa sổ hội thoại khác nhau) cần giống nhau về nhãn và vị trí liên hệ.

3.1.3 Thiết kế hướng chức năng

Thiết kế hướng chức năng có nghĩa là tập trung trên thuật toán để giải quyết vấn đề.

Chúng ta có thể xem một thuật toán như một hàm tính toán mà tính kết quả từ những tham số cơ bản được cho. Tại thời điểm bắt đầu giai đoạn thiết kế, thuật toán như là hộp đen mà nội dung thì không được biết. Chúng ta cần xây dựng những thuật toán để giải quyết những tác nhiệm khó hoặc phức tạp của phần mềm. Như vậy, việc thực hiện mô-đun hóa để phân rã những tác nhiệm thành tác nhiệm con độc lập nhau và nhờ những thuật toán xử lý các tác nhiệm con đó, tạo thành những hộp đen. Kết quả chung của những giải pháp trở thành mạng lưới những thuật toán con gộp lại.

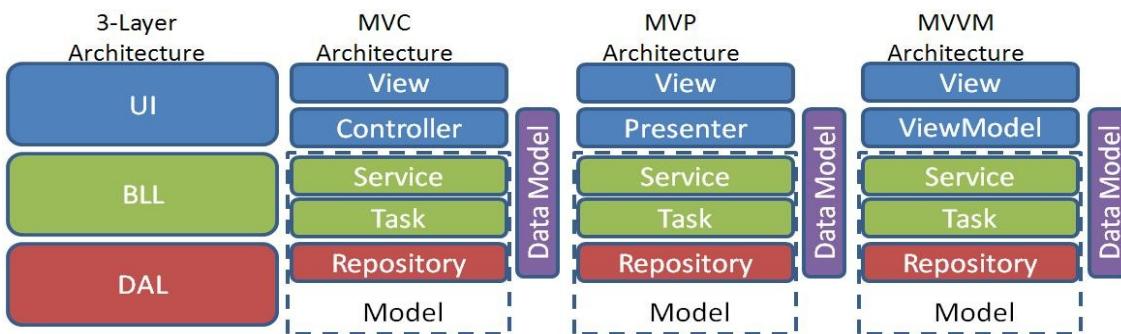
3.1.4 Thiết kế hướng đối tượng

Thiết kế hướng đối tượng là tổ chức thiết kế xoay quanh những đối tượng và mối liên hệ giữa chúng, bao gồm các loại:

- Thiết kế lớp đối tượng: mô tả lớp đối tượng cùng thuộc tính, hành vi.

- Thiết kế giao diện: mô tả giao diện của lớp đối tượng trong từng trách nhiệm của chúng (khái niệm giao diện này không phải là khái niệm giao diện người dùng hay giao diện đồ họa).
- Thiết kế *dữ liệu*: Mô tả cách thức tổ chức lưu trữ các đối tượng trên bộ nhớ phụ (chỉ có khi không sử dụng cơ sở dữ liệu hướng đối tượng)

Ngoài ra, khả năng tái sử dụng của các đối tượng đóng vai trò quan trọng trong lập trình hướng đối tượng (đối với chuyên viên tin học phải thực hiện nhiều phần mềm). Với tiếp cận hướng đối tượng, việc tái sử dụng sẽ rất dễ dàng, nhanh chóng, đồng thời tốn ít chi phí nhất vì các phần mềm trong cùng lớp phần mềm gồm các đối tượng tương tự như nhau, cách xây dựng đối tượng tương tự nhau cho các phần mềm khác nhau.



Hình 3.6: Minh họa các dạng kiến trúc phần mềm (nguồn [5])

3.2 KIẾN TRÚC PHẦN MỀM

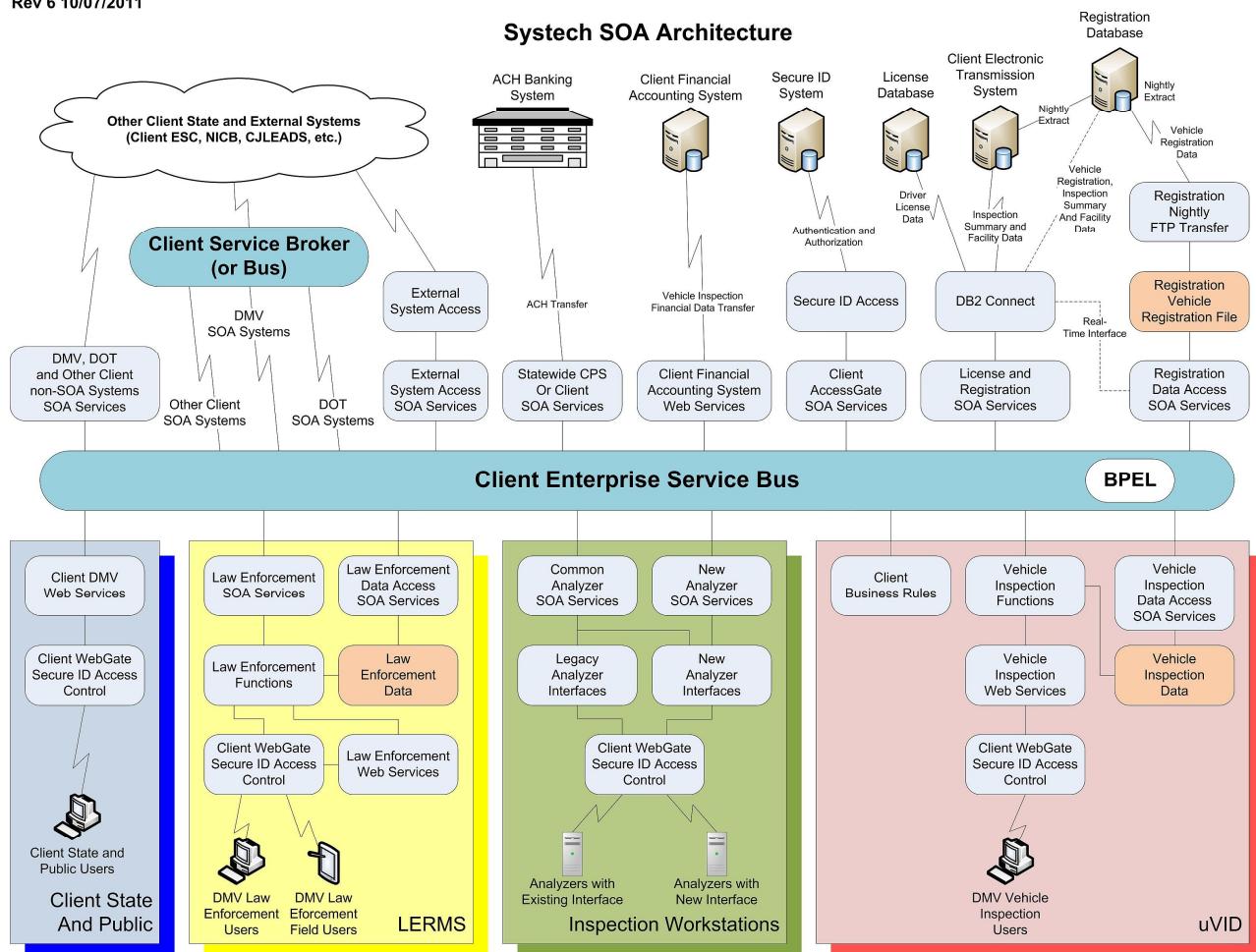
Kiến trúc phần mềm bao gồm các thành phần cơ bản: *Giao diện*, *Xử lý*, *Dữ liệu*. Khi thiết kế một phần mềm, nhóm thiết kế phải chọn lựa và ra quyết định về các “vật liệu” được dùng trong các thành phần. Khi đã quyết định xong, kết quả sẽ được đặc tả dưới dạng các bản vẽ phần mềm, dưới dạng tài liệu kỹ thuật, tạo thành các mô hình phần mềm chứa đầy đủ thông tin gồm:

- *Thành phần Giao diện*² với các thông tin sau:
 - Nội dung và hình thức trình bày của các màn hình giao tiếp

² Xem thêm phần Thiết kế Giao diện

- *Hệ thống thao tác* mà người dùng thực hiện trên màn hình giao tiếp và xử lý tương ứng.
- *Thành phần Xử lý* gồm các thông tin sau:
 - *Hệ thống các kiểu dữ liệu* được sử dụng trong phần mềm. Các kiểu dữ liệu này được mô tả về cách tổ chức lưu trữ dữ liệu trong bộ nhớ chính của phần mềm.
 - *Hệ thống các hàm* được sử dụng trong phần mềm. Các hàm này thể hiện tương ứng các tác nhiệm (trên máy tính) xử lý một công việc nào đó của thế giới thực (ví dụ kiểm tra tính hợp lệ việc cho mượn sách, ghi vào sổ việc cho mượn sách ...)
- *Thành phần Dữ liệu*³ bao gồm:
 - Các thông tin liên quan đến *cách tổ chức lưu trữ dữ liệu* (nội dung ghi chép vào sổ sách trong thế giới thực) trên bộ nhớ phụ,
 - *Dạng lưu trữ* của dữ liệu được sử dụng của phần mềm,
 - Hệ thống các thành phần lưu trữ cùng với quan hệ giữa chúng.

³ Xem thêm phần Thiết kế Dữ liệu



Hình 3.7: Minh họa về kiến trúc tổng quan phần mềm (nguồn [5])

3.3 PHƯƠNG PHÁP THIẾT KẾ PHẦN MỀM

Tùy thuộc vào quy trình được chọn khi thực hiện phần mềm, việc thiết kế có thể được tiến hành theo 2 phương pháp chính:

❖ Phương pháp trực tiếp

Phương pháp này được áp dụng khi thực hiện phần mềm không thông qua giai đoạn phân tích, trong đó việc thiết kế nhận kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu, từ đó mô hình phần mềm sẽ được xây dựng trực tiếp từ các yêu cầu. Cách tiếp cận này rất khó khăn cho người thực hiện với các phần mềm có quy mô lớn (nhiều yêu cầu hay yêu cầu phức tạp ...).

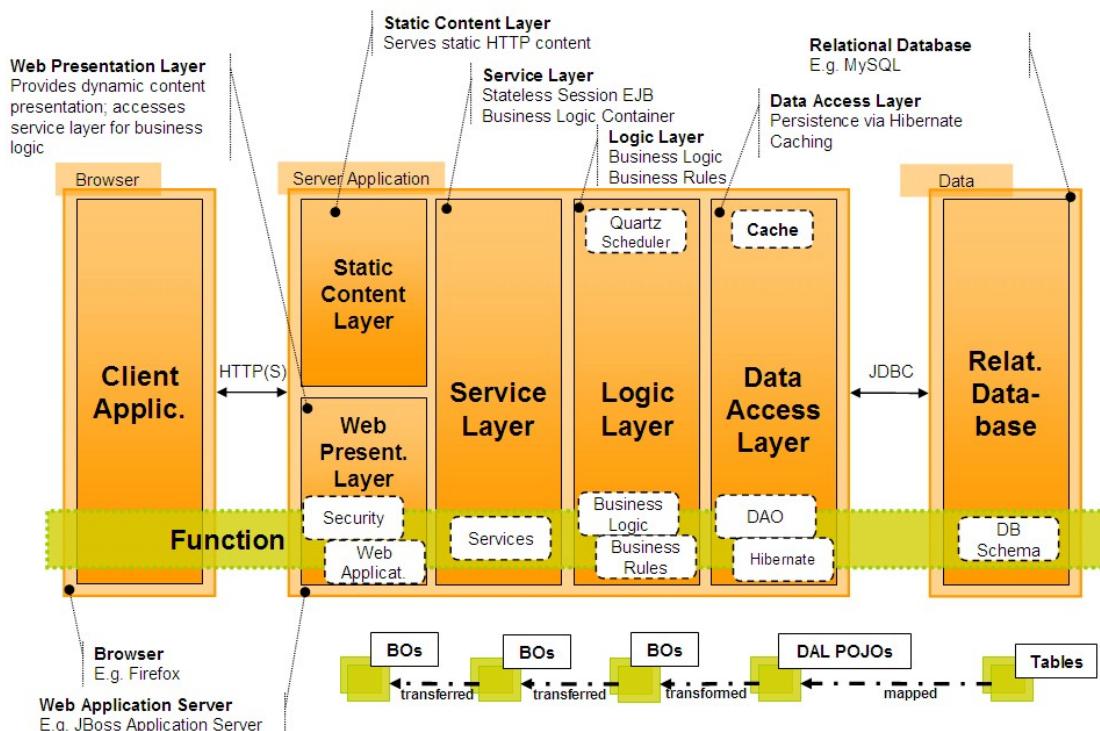
Với phương pháp trực tiếp, thiết kế phần mềm là quá trình cho phép chuyển đổi từ các yêu cầu (kết quả giai đoạn xác định yêu cầu) đến mô hình phần mềm tương ứng.

Mục tiêu chính của việc thiết kế là mô tả các thành phần của phần mềm (Giao diện, Xử lý, Dữ liệu) tương ứng với các yêu cầu của phần mềm (yêu cầu chức năng nghiệp vụ, yêu cầu chức năng hệ thống, yêu cầu phi chức năng).

❖ Phương pháp gián tiếp

Phương pháp này được áp dụng với các quy trình có giai đoạn phân tích, trong đó việc thiết kế sẽ chỉ nhận một phần các kết quả chuyển giao trực tiếp từ giai đoạn xác định yêu cầu, phần chính yếu sẽ được nhận được gián tiếp qua giai đoạn phân tích, từ đó mô hình phần mềm sẽ được xây dựng tương ứng theo các mô hình trong giai đoạn phân tích. Cách tiếp cận này sẽ rất thuận lợi trong đa số trường hợp với các phần mềm có quy mô lớn.

Với phương pháp gián tiếp, thiết kế phần mềm là quá trình cho phép chuyển từ mô hình thế giới thực (kết quả giai đoạn phân tích) đến mô hình phần mềm tương ứng. Mục tiêu chính của việc này là mô tả các thành phần của phần mềm (Giao diện, Xử lý, Dữ liệu) tương ứng với các mô hình của thế giới thực (mô hình xử lý, mô hình dữ liệu).



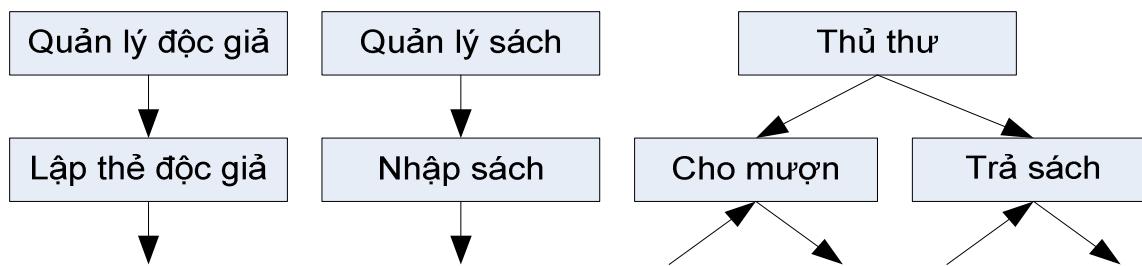
Hình 3.8: Minh họa về bản thiết kế phần mềm chi tiết (nguồn [5])

3.4 VÍ DỤ MINH HỌA

Ví dụ sau đây chỉ nhằm minh họa quá trình thiết kế phần mềm sau khi thực hiện giai đoạn mô hình hóa yêu cầu. Các kết quả chỉ chú trọng chủ yếu tính đúng đắn và bỏ qua các yêu cầu chất lượng khác (tiến hóa, hiệu quả, tiện dụng). Kết quả thực tế khi xem xét đầy đủ các yêu cầu chất lượng là quá phức tạp và không thích hợp cho việc minh họa.

Ví dụ: Xét phần mềm quản lý thư viện với 4 yêu cầu: Lập thẻ độc giả, Nhận sách, Cho mượn sách, Trả sách

❖ Mô hình hóa các yêu cầu



Hình 3.9: Sơ đồ chức năng

❖ Thiết kế phần mềm

Hệ thống các màn hình giao diện gồm:

Màn hình chính

- Nội dung: Thông tin về thư viện, Thông tin về các độc giả trong thư viện, Thông tin về các sách trong thư viện.
- Thao tác người dùng: Tra cứu và chọn độc giả, Tra cứu và chọn sách.

Màn hình Lập thẻ

- Nội dung: Thông tin về thẻ độc giả.
- Thao tác người dùng: Nhập thông tin về thẻ, Yêu cầu lập thẻ

Màn hình Cho mượn sách:

- Nội dung: Thông tin về thẻ độc giả (Ngày mượn sách, Danh mục sách).
- Thao tác người dùng: Nhập thông tin mượn sách, Yêu cầu mượn sách.

Màn hình Nhận sách:

- Nội dung: Ngày nhận sách, Danh mục sách nhận & thông tin liên quan.
- Thao tác người dùng: Nhập thông tin về việc cho nhận sách, Yêu cầu cho nhận sách.

Màn hình Trả sách:

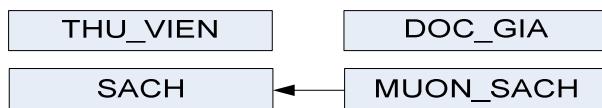
- Nội dung: Ngày trả sách, Thông tin về việc trả sách.
- Thao tác người dùng: Nhập thông tin trả sách, Yêu cầu trả sách.

❖ Hệ thống các hàm xử lý

- *Hàm Lập thẻ*: Kiểm tra tính hợp lệ và lưu thẻ vào kho
- *Hàm Tra cứu độc giả*: Tìm thẻ độc giả theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa thẻ
- *Hàm Xóa thẻ*: Xóa thẻ trong kho
- *Hàm Nhập sách*: Kiểm tra tính hợp lệ của sách và lưu sách vào kho
- *Hàm Xóa sách*: Xóa sách trong kho
- *Hàm Cho mượn sách*: Kiểm tra tính hợp lệ của việc cho mượn sách và ghi nhận các thông tin cho mượn sách vào kho
- *Hàm Tra cứu sách*: Tìm sách theo các tiêu chuẩn khác nhau để cho phép cập nhật hay xóa sách
- *Hàm Tính số lượng sách độc giả đang mượn*: Tính số lượng sách độc giả đang mượn và chưa trả
- *Hàm Kiểm tra độc giả có sách mượn quá hạn*: Kiểm tra độc giả có sách mượn quá hạn và trả về 1 nếu đúng, 0 nếu sai
- *Hàm Kiểm tra tình trạng sách*: Kiểm tra sách đang được mượn trả về 1 nếu đúng và 0 nếu sai
- *Hàm Tra cứu phiếu cho mượn sách*: Tra cứu các phiếu mượn sách theo nhiều tiêu chuẩn để cập nhật hay số phiếu cho mượn
- *Hàm Xóa phiếu cho mượn sách*: Xóa thông tin việc mượn sách trong kho

- *Hàm Trả sách*: Ghi nhận việc trả sách trong kho
- *Hàm Tính tiền phạt*: Tính tiền phạt khi độc giả trả sách trễ hạn

❖ **Hệ thống các bảng dữ liệu**



Hình 3.10: Sơ đồ ERD

- Bảng THU_VIEN: các thông tin về thư viện
- Bảng DOC_GIA: các thông tin về độc giả
- Bảng SACH: các thông tin về sách
- Bảng MUON_SACH: các thông tin về mượn trả sách

TÓM TẮT

Trong bài này, chúng ta quan tâm về các vấn đề liên quan đến phương pháp thiết kế như:

- *Kỹ thuật thiết kế*:
 - *Thiết kế từ trên xuống (Top-down), từ dưới lên (Bottom-up)*
 - *Thiết kế Hệ thống, Thiết kế Bản mẫu (Prototype)*
 - *Phân rã Thiết kế*
- *Thiết kế giao diện người dùng*
- *Thiết kế hướng chức năng và thiết kế hướng đối tượng*
- *Kiến trúc phần mềm*
- *Phương pháp thiết kế phần mềm*

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 4: TÁC VỤ THIẾT KẾ & TỔ CHỨC DỮ LIỆU

Học xong bài này người học sẽ:

- *Hiểu được các khái niệm cơ bản về thiết kế, kết quả của thiết kế, quá trình thiết kế, phương pháp thiết kế dữ liệu.*
- *Biết áp dụng các phương pháp thiết kế dữ liệu (trực tiếp hay gián tiếp) vào tác vụ xây dựng cơ sở dữ liệu cho hệ thống phần mềm.*

4.1 TỔNG QUAN

Mục tiêu chính của việc thiết kế dữ liệu là mô tả cách thức tổ chức lưu trữ các dữ liệu của phần mềm. Có hai dạng lưu trữ chính mà người thiết kế cần phải cân nhắc và lựa chọn là:

- Lưu trữ dưới *dạng tập tin*: Lưu trữ dưới dạng tập tin thường chỉ thích hợp với một số phần mềm đặc thù (từ điển, trò chơi ...) với đặc điểm chung của các phần mềm này là chú trọng rất nhiều vào xử lý, hình thức giao diện và không chú trọng nhiều đến việc lưu trữ lại các thông tin được tiếp nhận trong quá trình sử dụng phần mềm vì thông thường các thông tin này được tiếp nhận và xử lý ngay.
- Lưu trữ dưới *dạng cơ sở dữ liệu*: Trong thực tế, cách tiếp cận này rất thông dụng và đóng vai trò quan trọng trong việc thiết kế những phần mềm cở vừa và lớn hoặc dạng phân tán ...

4.2 KẾT QUẢ CỦA THIẾT KẾ

Cách thức tổ chức lưu trữ dữ liệu của phần mềm được mô tả thông qua 2 loại thông tin sau:

- Thông tin *tổng quát*: Cung cấp góc nhìn tổng quan về các thành phần lưu trữ, gồm có:
 - Danh sách các *bảng dữ liệu*: liên quan đến việc lưu trữ cụ các bảng dữ liệu cụ thể.
 - Danh sách các *liên kết*: liên quan đến việc hiện thực các mối liên kết dữ liệu giữa các bảng dữ liệu.
- Thông tin *chi tiết*:
 - Danh sách các thuộc tính của từng thành phần: liên quan đến các thông tin cần lưu trữ của thành phần
 - Danh sách các *miền giá trị toàn vẹn*: liên quan đến các quy định về tính hợp lệ của các thông tin được lưu trữ.

Có nhiều cách khác nhau về việc mô tả các thông tin trên, nhưng *sơ đồ luận lý* là cách khá tốt để biểu diễn thông tin tổng quát, và bảng thuộc tính cùng miền giá trị được dùng để mô tả chi tiết các thành phần trong sơ đồ đó.

Sơ đồ luận lý cho phép thể hiện hệ thống các *bảng dữ liệu* cùng với *quan hệ liên kết* giữa chúng (gần giống ERD), với các ký hiệu được dùng như sau:

- **Bảng:** hình chữ nhật
- **Liên kết:** (xác định duy nhất một): Mũi tên



Mũi tên hình trên có ngữ nghĩa: 1 phần tử A sẽ xác định duy nhất 1 phần tử B, ngược lại 1 phần tử B có thể tương ứng với nhiều phần tử A.

Ví dụ: Với phần mềm quản lý thư viện có sơ đồ luận lý sau:



Theo sơ đồ này việc lưu trữ các dữ liệu của phần mềm quản lý thư viện được tổ chức 3 bảng (DOCGIA, MUONSACH, SACH) với 2 liên kết giữa chúng. Tuy nhiên, sơ đồ trên chỉ là một trong nhiều cách thức tổ chức lưu trữ dữ liệu (*tham khảo thêm trong tài liệu Phân tích Thiết kế Hệ thống thông tin*).

Bảng thuộc tính cho phép mô tả chi tiết thành phần trong sơ đồ luận lý theo dạng như: Thành phần, Ý nghĩa ...

STT	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1					
2					
...					

Bảng miền giá trị cho phép mô tả các Miền giá trị giữa các thuộc tính cùng một thành phần hay nhiều thành phần khác nhau.

Mã Số	Mô tả miền giá trị	Thành phần liên quan	Ghi chú
RB1			
RB2			
...			

❖ Ghi chú

Bảng thuộc tính cho phép mô tả chi tiết thành phần cần lưu trữ và sẽ được dùng trong báo cáo về thiết kế dữ liệu của phần mềm. Tuy nhiên cách mô tả trên khá dài dòng, trong tài liệu này sẽ sử dụng một dạng trình bày cô đọng hơn theo dạng *lược đồ quan hệ*. Với dạng trình bày này gồm tên bảng và thuộc tính đi kèm, các thuộc tính khóa được gạch chân.

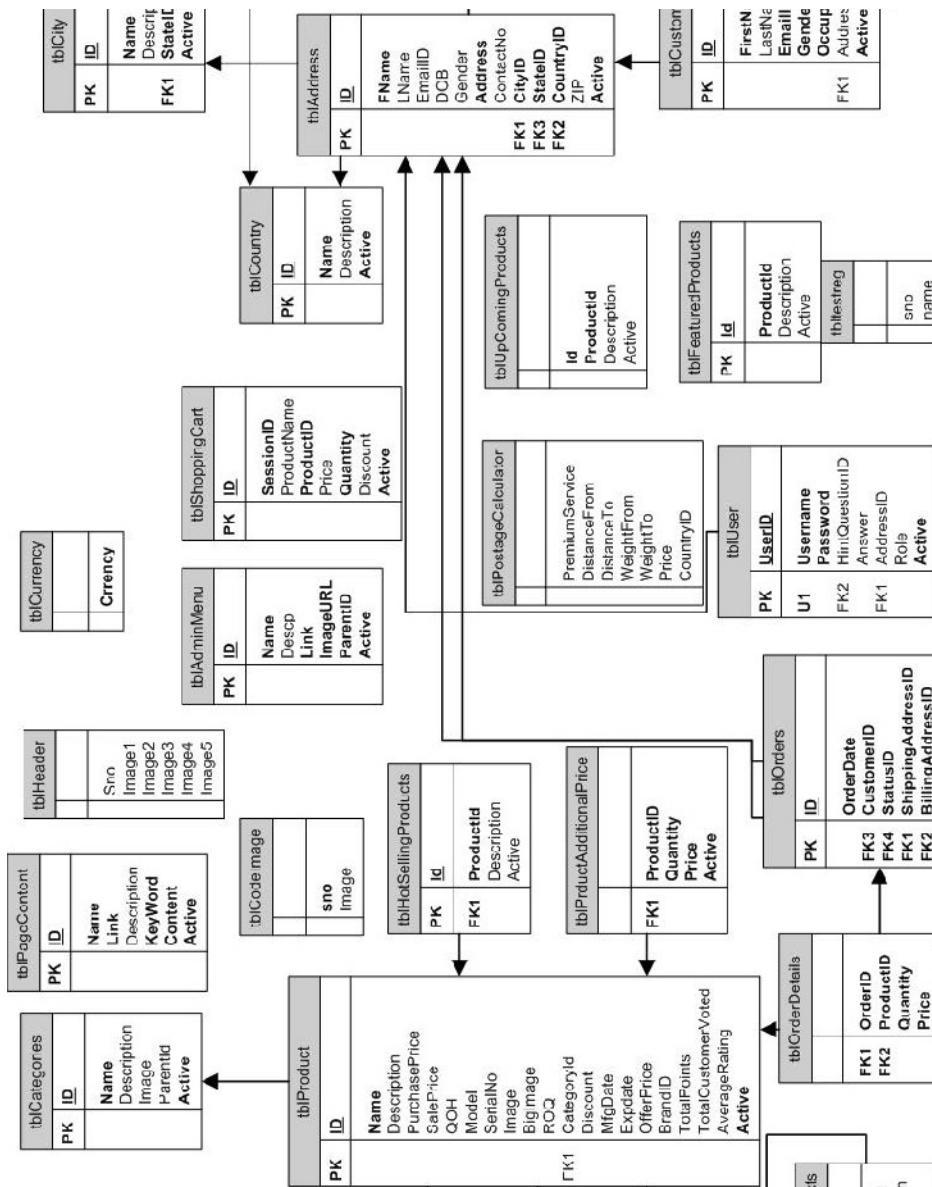
Ví dụ:

DOC_GIA(MDG, HoTen, LoaiDG, NgaySinh, NgayLapThe, DiaChi)

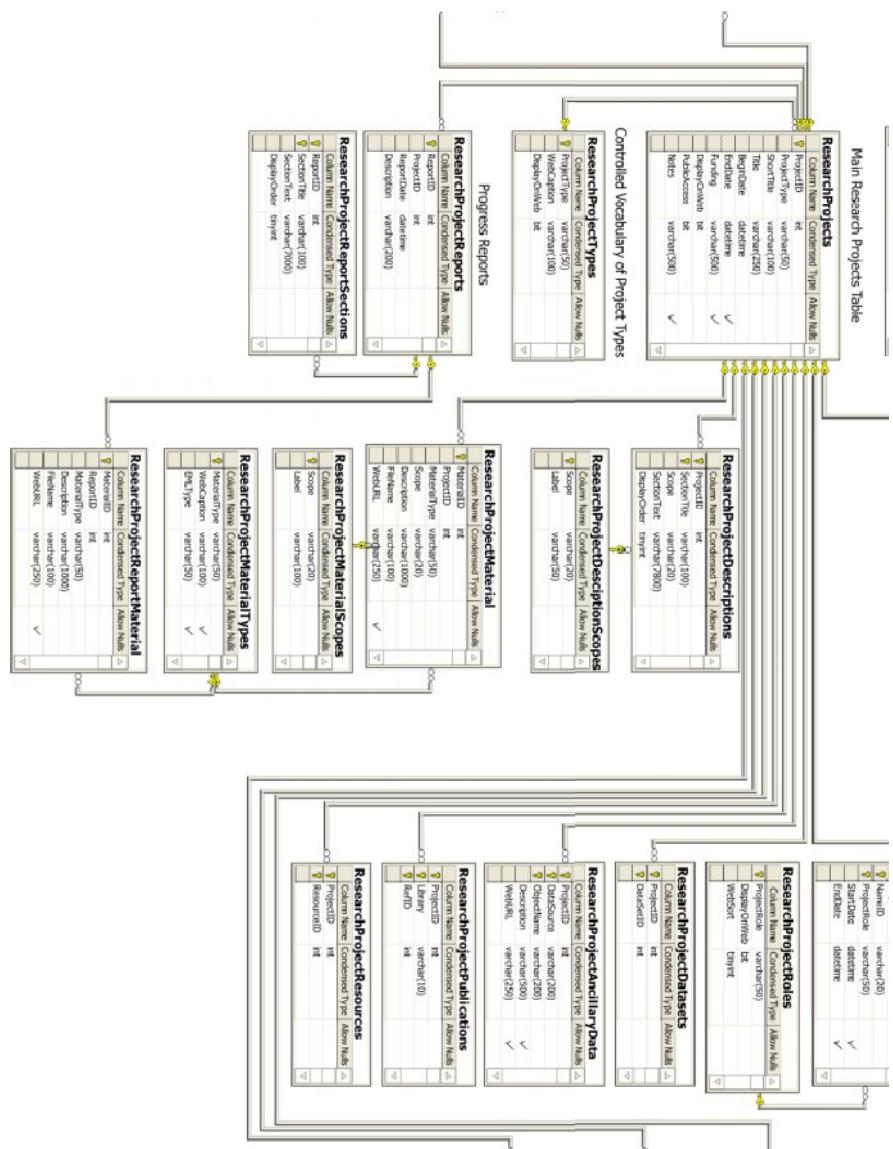
SACH(MSACH, TenSach, TheLoai, NgayNhap, TacGia, NhaXuatBan, NamXuatBan)

MUON(MDG, MSACH, NgayMuon, NgayTra)

❖ Ví dụ minh họa:



Hình 4.1: Minh họa sơ đồ ERD mức chi tiết (nguồn [5])



Hình 4.2: Minh họa sơ đồ thiết kế dữ liệu vật lý (nguồn [5])

4.3 QUÁ TRÌNH THIẾT KẾ

Có 2 cách tiếp cận chính để thiết kế dữ liệu, đó là:

- ❖ **Phương pháp trực tiếp**

Từ các yêu cầu đã xác định, ta tạo lập trực tiếp sơ đồ luận lý cùng với bảng thuộc tính và bảng miền giá trị. Cách tiếp cận này rất khó thực hiện đối với sơ đồ luận lý phức tạp.

- ❖ **Phương pháp gián tiếp**

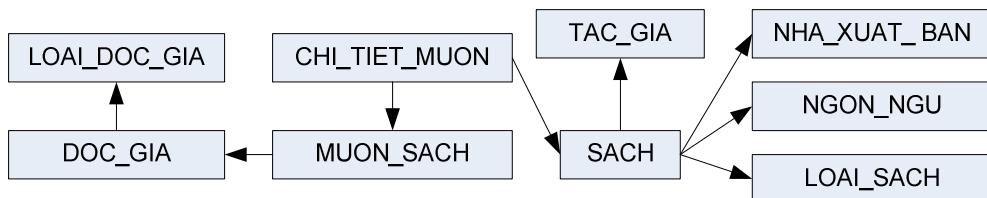
Từ các yêu cầu đã xác định, ta tạo lập mô hình quan niệm dữ liệu, sau đó tạo lập sơ đồ luân lý, bảng thuộc tính, bảng miền giá trị và đưa vào mô hình này. Cách tiếp cận này dễ thực hiện hơn vì mô hình quan niệm dữ liệu thường đơn giản (chứa các thành phần dữ liệu bản chất nhất của phần mềm).

Ứng với 3 yêu cầu của phần mềm, việc thiết kế dữ liệu gồm 3 bước lớn:

- Thiết kế với *tính đúng đắn*: cần đảm bảo đầy đủ và chính xác về mặt ngữ nghĩa các thông tin liên quan đến các công việc trong yêu cầu. Tuy nhiên, các thông tin phục vụ cho các yêu cầu chất lượng sẽ không được xét đến trong bước thiết kế này.
- Thiết kế với *yêu cầu chất lượng*: cần đảm bảo tính đúng đắn nhưng thỏa mãn thêm các yêu cầu chất lượng khác (tiến hóa, tốc độ nhanh, lưu trữ tối ưu) và bảo đảm tính đúng đắn khi cải tiến sơ đồ luân lý.
- Thiết kế với *yêu cầu hệ thống*: cần đảm bảo tính đúng đắn và các yêu cầu chất lượng khác nhưng thỏa mãn thêm các yêu cầu hệ thống (phân quyền, cấu hình phần cứng, môi trường phần mềm ...)

Ví dụ 1: Xét phần mềm quản lý thư viện, với phương pháp trực tiếp sẽ cho kết quả như sau:

Sơ đồ luân lý



Các bảng thuộc tính

DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoai)

SACH(MSACH, MTG, MNXB, MLSACH, MNN, TenSach, NgayMua, SoTrang)

PHIEU_MUON(MPHM, NgayMuon)

CHI_TIET_MUON(MPHM, MSACH, NgayTra)

LOAI_SACH(MLSACH, TenLS, GhiChu)

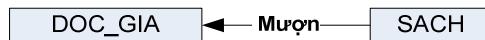
LOAI_DOC_GIA(MLDG, TenLoaiDocGia, GhiChu)

NHA_XUAT_BAN(MNXB, TenNhaXuatBan, GhiChu)

TAC_GIA(MTG, Ten, GhiChu)

NGON_NGU(MNN, Ten, GhiChu)

Với phương pháp gián tiếp, ngoài kết quả cuối cùng tương tự phương pháp trực tiếp, ta còn kết quả trung gian là mô hình quan niệm dữ liệu như sau:



Sơ đồ lớp đối tượng với 2 đối tượng chính Sách, Độc giả và 1 quan hệ mượn giữa 2 lớp đối tượng trên. Mô hình chi tiết các thành phần trong sơ đồ lớp: *Xem chi tiết ở phụ lục B*

Ví dụ 2: Xét phần mềm với 4 yêu cầu: Lập thẻ độc giả, Nhận sách, Cho mượn sách, Trả sách

Thiết kế dữ liệu với tính đúng đắn

- Sơ đồ luận lý



- Chi tiết các bảng

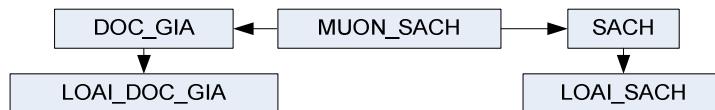
DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoai)

SACH(MSACH, MTG, MNXB, MLSACH, MNN, TenSach, NgayMua, SoTrang)

MUON_SACH(MDG, MSACH, NgayMuon, NgayTra, TienPhat)

Thiết kế dữ liệu với tính tiến hóa

- Sơ đồ luận lý



- Chi tiết các bảng:

DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoaiNguoiLapThe, NgayHetHan)

SACH(MSACH, Tensach, MTL, NgayNhap, TacGgia, NamXuatBan, NhaXuatBan)

MUON_SACH(MDG, MSACH, NgayMuon, NgayTra, TienPhat)

THE_LOAI(MTL, TenTheLoai, GhiChu)

LOAI_DOC_GIA(MLDG, TenLoaiDocGia, GhiChu)

Thiết kế với tính hiệu quả (truy xuất nhanh)

- **Sơ đồ luận lý**

Cùng sơ đồ luận lý như trên, nhưng ta có các bảng thuộc tính như sau.

- **Bảng thuộc tính**

DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoaiNguoiLapThe, NgayHetHan, SoSachMuon, TinhTrangTra)

SACH(MSACH, Tensach, MTL, NgayNhap, TacGia, NamXuatBan, NhaXuanBan, TinhTrangMuon)

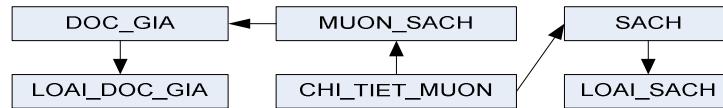
MUON_SACH(MDG, MSACH, NgayMuon, NgayTra, TienPhat)

THE_LOAI(MTL, TenTheLoai, GhiChu)

LOAI_DOC_GIA(MLDG, TenLoaiDocGia, GhiChu)

Thiết kế dữ liệu với tính hiệu quả (lưu trữ tối ưu)

- **Sơ đồ luận lý**



- **Chi tiết các bảng thuộc tính**

DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoaiNguoiLapThe, NgayHetHan, SoSachMuon, TinhTrangTra)

SACH(MSACH, Tensach, MTL, NgayNhap, TacGia, NamXuatBan, NhaXuanBan, TinhTrangMuon)

MUON_SACH(MDG, MSACH, NgayMuon, NgayTra, TienPhat)

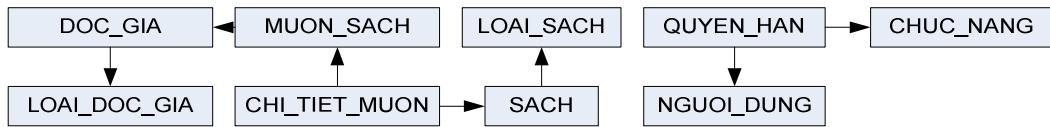
CHI_TIET_MUON(MMUON, MSACH, NgayTra, TienPhat)

THE_LOAI(MTL, TenTheLoai, GhiChu)

LOAI_DOC_GIA(MLDG, TenLoaiDocGia, GhiChu)

Thiết kế dữ liệu với yêu cầu phân quyền hệ thống

- **Sơ đồ luận lý**



- **Chi tiết các bảng**

DOC_GIA(MDG, MLDG, HoTen, NgaySinh, DiaChi, DienThoaiNguoiLapThe, NgayHetHan, SoSachMuon, TinhTrangTra)

SACH(MSACH, Tensach, MTL, NgayNhap, TacGia, NamXuatBan, NhaXuanBan, TinhTrangMuon)

MUON_SACH(MDG, MSACH, NgayMuon, NgayTra, TienPhat)

CHI_TIET_MUON(MMUON, MSACH, NgayTra, TienPhat)

THE_LOAI(MTL, TenTheLoai, GhiChu)

LOAI_DOC_GIA(MLDG, TenLoaiDocGia, GhiChu)

NGUOI_DUNG(MND, HoTen, Ghichu)

CHUC_NANG(MCN, Ten_ChucNang, GhiChu)

QUYEN_HAN(MND, MCN)

4.4 PHƯƠNG PHÁP THIẾT KẾ DỮ LIỆU

4.4.1 Phương pháp Trực tiếp

❖ Bước 1

- Lập sơ đồ với 1 thành phần duy nhất,
- Đánh giá tính đúng đắn so với yêu cầu, chuyển sang bước 2 nếu cần.

❖ Bước 2

- Tách 1 số thuộc tính để tạo ra các thành phần mới,
- Xác định liên kết giữa các thành phần,
- Đánh giá tính đúng đắn so với yêu cầu, lặp lại bước 2 nếu cần.

Ví dụ: phần mềm quản lý thư viện

Cách 1: Dùng thành phần SACH

MSach, Ten, TheLoai, NgayMua, TacGia, NhaXuatBan, NamXuatBan HoTenDocGia, LoaiDocGia, NgayLamThe, NgayMuon, NgayTra

Cách 2: Dùng thành phần SACH, DOC_GIA

- Cách 2.1: Chỉ lưu trữ lần mượn sách cuối cùng

SACH(MSACH, MADG, Ten, TheLoai, NgayMua, TacGia, NhaXuatBan, NamXuatBan, NgayMuon, NgayTra)

DOC_GIA(MDG, HoTen, LoaiDocGia, NgayLamThe)

- **Cách 2.2: Chỉ cho phép độc giả mượn tối đa 1 quyển sách**

SACH(MSACH, Ten, TheLoai, NgayMua, TacGia, NhaXB, NamXB, NgayMuon, NgayTra)

DOC_GIA(MDG, MSACH, HoTen, LoaiDocGia, NgayLamThe, NgayMuon)

- **Cách 3: Dùng thành phần SACH, DOC_GIA, MUON_SACH**

SACH(MSACH, Ten, TheLoai, NgayMua, TacGia, NhaXuatBan, NamXuatBan, NgayMuon, NgayTra)

DOC_GIA(MDG, HoTen, LoaiDocGia, NgayLamThe)

MUON_SACH(MMUON, MDG, MSACH, NgayMuon, NgayTra)

4.4.2 Phương pháp Gián tiếp

❖ Bước 1

- Lập sơ đồ lớp, Xác định các lớp đối tượng, Xác định quan hệ giữa các lớp đối tượng và lập sơ đồ.

❖ Bước 2

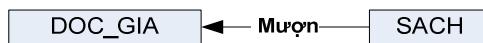
- Ánh xạ từ sơ đồ lớp vào sơ đồ luận lý, Ánh xạ các lớp đối tượng, Ánh xạ các quan hệ giữa các lớp đối tượng.

❖ Bước 3

- Hoàn chỉnh sơ đồ luận lý, Bổ sung các thành phần theo yêu cầu, Mô tả chi tiết các thuộc tính của các thành phần.

4.4.2.1 Lập Sơ đồ Lớp

Ví dụ: Với phần mềm quản lý thư viện, 2 đối tượng chính là Độc giả, Sách và quan hệ giữa chúng là quan hệ mượn sách.



4.4.2.2 Ánh xạ Sơ đồ Lớp

Ánh xạ lớp đối tượng sao cho mỗi đối tượng trong sơ đồ lớp tương ứng với 1 thành phần trong sơ đồ luận lý.



Sơ đồ luân lý:



4.4.2.3 Ánh xạ Quan hệ

- Quan hệ $1-n$: biểu diễn liên kết giữa 2 lớp đối tượng A và B (1 A với nhiều B) xác định duy nhất từ A sang B trong sơ đồ luân lý.
- Quan hệ $m-n$: biểu diễn liên kết giữa 2 lớp đối tượng A và B tương ứng với 1 thành phần C trong sơ đồ luân lý và xác định duy nhất với A, B.

Sơ đồ lớp:



Sơ đồ luân lý:



4.4.2.4 Hoàn chỉnh Sơ đồ Luận lý

❖ Bổ sung các thành phần

- *Đối tượng phụ*: tương ứng với 1 thành phần trong sơ đồ luân lý
- Các thành phần khác: Xét tính đúng đắn và bổ sung thêm nếu cần thiết.

❖ Mô tả chi tiết thuộc tính các thành phần

- Thuộc tính *khóa chính*:

- Mỗi thành phần ứng với đối tượng (chính hoặc phụ) cần 1 thuộc tính khóa riêng,
- Các thành phần còn lại, tùy theo ý nghĩa sử dụng, sẽ có thuộc tính khóa riêng hay dùng tổ hợp thuộc tính khóa của các thành phần khác.

Ví dụ: Các thành phần DOC_GIA, SACH, NHA_XUAT_BAN, TAC_GIA sẽ có thuộc tính khóa chính tương ứng là MDG, MSACH, MNXB, MTG. Thành phần MUON cũng sẽ có khóa chính là MMUON.

- Thuộc tính *khóa ngoại*: Thể hiện đúng liên kết giữa các thành phần trong sơ đồ luân lý: nếu A xác định duy nhất B thì A có thuộc tính là khóa chính của B (đó là khóa ngoại của A)

Ví dụ: Thành phần MUON có 2 khóa ngoại: MDG, MSACH; Thành phần SACH có 2 khóa ngoại: MNXB, MTG, MDG

- *Các thuộc tính khác:* Cần dựa vào yêu cầu lưu trữ, chú ý thuộc tính sau:

- Định danh: Tên
- Loại: Sự phân loại
- Thời gian: Ngày tháng
- Không gian: vị trí
- Định lượng: độ đo, tính chất ...

Ví dụ:

- DOC_GIA có thuộc tính khác như: HoTen (định danh) LoaiDG (loại) Ngaysinh (thời gian) Ngayhethan (thời gian) Diachi (không gian);
- SACH có thuộc tính khác như: TenSach (định danh) LoaiSach (loại) NgayMua (thời gian) GiaTien (định lượng)

❖ Thiết kế dữ liệu với tính đúng đắn

Các bước thực hiện:

- Bước 1: Chọn một yêu cầu và xác định sơ đồ luận lý cho yêu cầu đó
- Bước 2: Bổ sung thêm một yêu cầu và xem xét lại sơ đồ luận lý
 - Nếu sơ đồ luận lý vẫn đáp ứng được thì tiếp tục bước 3,
 - Nếu sơ đồ luận lý không đáp ứng được thì bổ sung vào sơ đồ thuộc tính mới (ưu tiên hoặc thành phần mới (ưu tiên 2) cùng với các thuộc tính và liên kết tương ứng.
- Bước 3: Quay lại bước 2 cho đến khi đã xem xét đầy đủ các yêu cầu dựa trên ghi chú:
 - Với mỗi yêu cầu cần xác định rõ cần lưu trữ các thông tin gì dựa vào luồng dữ liệu đọc/ghi trong sơ đồ luồng dữ liệu tương ứng) và tìm cách bổ sung các thuộc tính để lưu trữ các thông tin này,
 - Chỉ xem xét tính đúng đắn,

- Cần chọn các yêu cầu theo thứ tự từ đơn giản đến phức tạp (thông thường yêu cầu tra cứu là đơn giản nhất),
- Với yêu cầu phức tạp có thể phải bổ sung vào sơ đồ luận lý nhiều thành phần mới.

Thuộc tính *khóa* của các thành phần phải dựa trên ngữ nghĩa tương ứng trong thế giới thực.

❖ **Thiết kế dữ liệu và yêu cầu chất lượng**

Mục tiêu

Ta xem xét đánh giá sơ đồ luận lý theo các yêu cầu về chất lượng rồi tiến hành cập nhật lại sơ đồ để bảo đảm các tiêu chuẩn về chất lượng. Ngoài tính đúng đắn cần ưu tiên hàng đầu, ta cần chú ý sự hợp lý giữa các phần mềm, đó là mức độ thỏa mãn các tiêu chuẩn chất lượng còn lại (đặc biệt là tính tiến hóa).

Xem xét tính tiến hóa

Để bảo đảm tính tiến hóa, sơ đồ luận lý sẽ được bổ sung cập nhật lại nhiều thành phần qua các bước thiết kế chi tiết. Trong bước thiết kế dữ liệu, ta sẽ giới hạn xem xét đến các *thuộc tính có giá trị rời rạc*, tương ứng khi *miền giá trị* chỉ bao gồm *một số giá trị nhất định*. Các giá trị này thông thường thuộc về tập hợp có độ biến động rất ít trong quá trình sử dụng phần mềm.

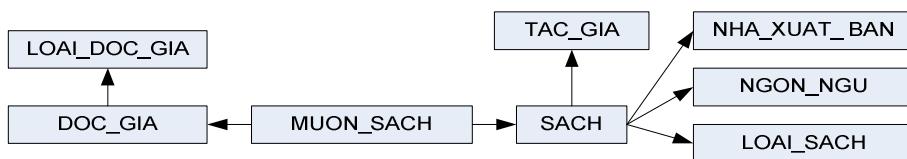
Ví dụ: LOAI_DOC_GIA (thành phần độc giả): thư viện hiện tại chỉ có 3 loại độc giả là 'A', 'B', 'C' và khả năng có thêm loại độc giả mới rất thấp. Ngôn ngữ (thành phần SACH): các sách trong thư viện hiện tại chỉ có 3 loại ngôn ngữ 'Việt', 'Anh', 'Pháp' và khả năng thêm sách thuộc ngôn ngữ mới rất thấp.

Tuy nhiên, ta cần lưu ý rằng khả năng biến động trên tập hợp giá trị của thuộc tính rời rạc là thấp nhưng không phải là không có. Khi xảy ra biến động (như thêm loại độc giả, thêm sách thuộc ngôn ngữ mới), nếu ta không chuẩn bị trước trong thiết kế thì người dùng sẽ không thể khai báo được các biến động này với phần mềm, khi đó có thể một số chức năng sẽ không thực hiện được (ví dụ người dùng sẽ không thể thêm sách mới với ngôn ngữ tiếng Hoa).

Như vậy, để chuẩn bị tốt cho biến động về sau (nếu có) trong tập hợp các giá trị của thuộc tính rời rạc, ta sẽ tách các thuộc tính này thành một thành phần trong sơ

đồ luận lý. Khi đó người dùng, trong quá trình sử dụng, hoàn toàn có thể cập nhật lại tập hợp các giá trị này tương ứng với các biến động thực tế trong thế giới thực.

Sơ đồ luận lý khi tách các thuộc tính rời rạc như sau:



Xem xét tính hiệu quả (tốc độ)

Phạm vi xem xét:

- Ta chỉ giới hạn xem xét việc tăng tốc độ thực hiện của phần mềm bằng cách bổ sung thêm các thuộc tính vào các bảng dùng lưu trữ các thông tin đã tính toán trước (theo quy tắc nào đó từ các thông tin gốc đã được lưu trữ). Ví dụ: số sách đang mượn của độc giả
- Các thông tin này phải được tự động cập nhật khi có bất kỳ thay đổi thông tin gốc liên quan. Ví dụ: độc giả mượn thêm hoặc trả sách

Các bước tiến hành:

- Bước 1: Chọn một yêu cầu và xem xét cần bổ sung thông tin gì trên bộ nhớ phụ để tăng tốc độ thực hiện của xử lý liên quan (các thông tin xử lý phải đọc mà không cần thực hiện việc tính toán)
- Bước 2: Làm lại bước 1 đến khi đã xét đầy đủ các yêu cầu theo ghi chú:
 - Sau mỗi bước, ta phải lập bảng danh sách các thuộc tính tính toán cùng với thông tin liên quan
 - Thông tin gốc
 - Xử lý tự động cập nhật thông tin gốc (chi tiết về các xử lý này sẽ được mô tả trong phần thiết kế xử lý)
 - Nếu thông tin gốc thường xuyên bị thay đổi, việc bổ sung thuộc tính tính toán để tăng tốc độ thực hiện sẽ mất ý nghĩa và hiệu ứng xấu.
 - Việc làm tăng tốc độ truy xuất sẽ dẫn đến việc lưu trữ không tối ưu

- Thứ tự xem xét các yêu cầu theo thứ tự từ đầu đến cuối (không cần chọn như các bước trong thiết kế dữ liệu)

Xem xét tính hiệu quả (lưu trữ)

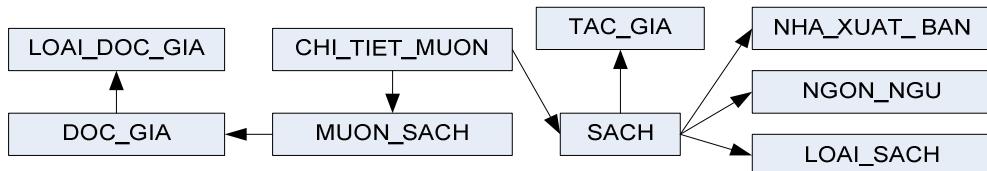
Tính hiệu quả trong thiết kế dữ liệu sẽ được xem xét dưới góc độ lưu trữ tối ưu. Vấn đề đặt ra là xây dựng sơ đồ luận lý sao cho vẫn đảm bảo đảm lưu trữ đầy đủ thông tin theo yêu cầu nhưng với dung lượng lưu trữ nhỏ nhất có thể có. Vấn đề này đặc biệt quan trọng với các phần mềm với hệ thống lưu trữ lớn và nhiều phát sinh thông tin cần lưu trữ theo thời gian. Khi đó ta cần đặc biệt quan tâm đến các thành phần mà dữ liệu tương ứng được phát sinh nhiều theo thời gian. Ta sẽ tìm cách bố trí lại sơ đồ luận lý sao cho vẫn đảm bảo thông tin mà dung lượng lưu trữ lại ít hơn.

Các bước tiến hành:

- Bước 1: Lập danh sách các bảng cần được xem xét để tối ưu hóa việc lưu trữ, cụ thể là:
 - Xem xét và xác định các công việc có tần suất thực hiện thường xuyên, và bổ sung chúng vào danh sách chọn các bảng được sử dụng tương ứng của công việc này,
 - Xem xét các bảng mà khóa của bảng bao gồm nhiều thuộc tính và bổ sung bảng này vào danh sách được chọn.
- Bước 2: Tối ưu hóa việc lưu trữ các bảng có khối lượng dữ liệu lưu trữ lớn thông qua việc tối ưu hóa lưu trữ từng thuộc tính trong bảng, gồm:
 - Xác định các thuộc tính mà việc lưu trữ chưa tối ưu và ưu tiên xem xét các thuộc tính có kiểu chuỗi,
 - Tối ưu hóa việc lưu trữ tùy theo từng trường hợp cụ thể,
 - Một trong các trường hợp thông dụng nhất là chuỗi có kích thước lớn và giá trị được sử dụng nhiều lần trong các mẫu tin khác nhau (ví dụ thuộc tính TacGia, NhaXuatBan trong bảng SACH của phần mềm),
 - Với trường hợp trên việc tối ưu hóa có thể thực hiện thông qua việc bổ sung các bảng mới (bảng TAC_GIA, NHA_XUAT_BAN) và tổ chức cấu trúc bảng SACH (thay thuộc tính TAC_GIA bằng MTG, thay thuộc tính NHA_XB bằng MNXB)

- Bước 3: Tối ưu hóa các bảng mà khóa của bảng bao gồm nhiều thuộc tính. Ta phân rã bảng đó thành hai bảng, trong đó một bảng chứa các thuộc tính mà giá trị được lặp lại nhiều lần trong cùng một lần thực hiện công việc tương ứng trong thế giới thực. Bảng này cần có khóa riêng (sẽ được bảng còn lại sử dụng để tham chiếu đến). Ta lưu ý là:
 - Việc phân rã giúp cho lưu trữ tối ưu, tuy nhiên:
 - Tốc độ truy xuất có thể sẽ chậm hơn,
 - Việc xử lý khó khăn hơn (thuật giải phức tạp hơn)
 - Cần cân nhắc trước khi thực hiện phân rã
 - Việc đánh giá khóa riêng cho bảng đã phân ra có thể cần kiểm tra thêm số phụ thuộc hàm

Ví dụ: Xét phần mềm quản lý thư viện.



Mô tả chi tiết các thuộc tính: **Xem chi tiết phụ lục B**

TÓM TẮT

Trong bài này, ta quan tâm đến những vấn đề liên quan đến kỹ thuật thiết kế dữ liệu như sau:

- Quá trình thiết kế
- Phương pháp thiết kế dữ liệu trực tiếp / gián tiếp: Lập sơ đồ lớp, Ánh xạ sơ đồ lớp, Ánh xạ quan hệ, Hoàn chỉnh sơ đồ luận lý.

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 5: TÁC VỤ THIẾT KẾ GIAO DIỆN

Học xong bài này người học sẽ nắm được các nội dung sau:

- *Hiểu được các khái niệm cơ bản về thiết kế giao diện liên quan các dạng màn hình (tra cứu, nhập liệu ...)*
- *Vận dụng được các phương pháp thiết kế để phác thảo giao diện người dùng hoàn chỉnh có các tính năng phù hợp với hệ thống phần mềm.*

5.1 TỔNG QUAN

Bài học này giới thiệu phương pháp thiết kế giao diện, đây là công đoạn quan trọng trong quá trình làm phần mềm và còn là công đoạn phác thảo bản mẫu cho phần mềm, từ đó nhận được phản hồi yêu cầu của khách hàng đối với chương trình và để người thiết kế có thể điều chỉnh theo yêu cầu đề ra.

Tùy theo mục đích, yêu cầu và độ phức tạp của chương trình, người thiết kế giao diện nên chú ý:

- Biết thông tin người dùng (họ là ai? mục đích của người dùng là gì? kỹ năng và kinh nghiệm của người dùng, nhu cầu của họ?),
- Quan sát, ghi nhận hành vi của người dùng trên hệ thống quen thuộc,
- Thể hiện:
 - Tính sẵn sàng của các chức năng trên giao diện của phần mềm,
 - Sự tương phản với giao diện của chương trình khi người dùng thay đổi trong ứng xử,
 - Tính tập trung (để các giao diện được tập trung chú ý nhiều hơn)
 - Quy luật của phần mềm, thông qua luật thao tác trên giao diện,

- Sử dụng biểu tượng sử dụng tắt (shortcut) để cung cấp cách thức cụ thể và tóm tắt tác vụ được làm trong phần mềm
- Cung cấp trợ giúp và độ an toàn cho phần mềm, giao diện đẹp, hạn chế phụ thuộc vào ngữ cảnh của người dùng ...

5.1.1 Kết quả thiết kế

❖ Màn hình giao diện

Màn hình giao diện là một trong các hình thức giao tiếp giữa người sử dụng và phần mềm khi họ thực hiện công việc của mình trên máy tính. Mục tiêu chính của thiết kế giao diện là mô tả hệ thống màn hình giao diện này.

❖ Kết quả thiết kế giao diện

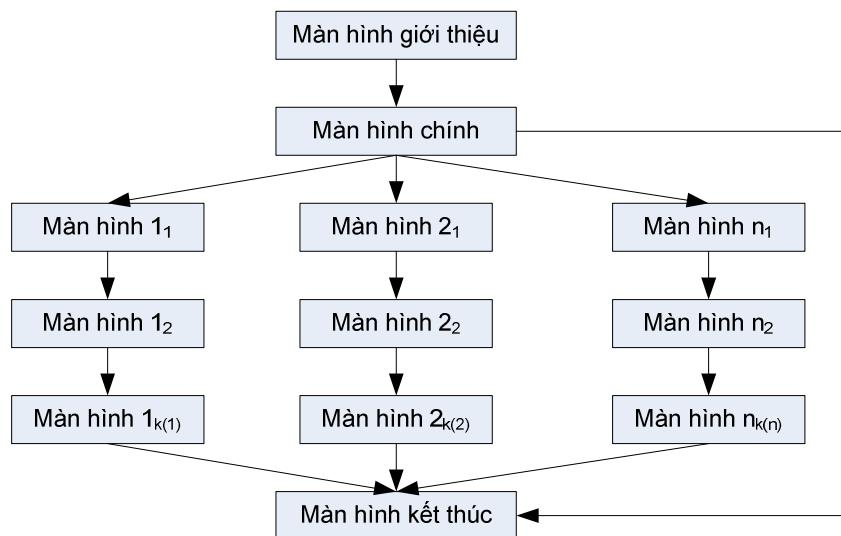
Kết quả gồm 2 phần:

- *Sơ đồ màn hình*: nhằm mô tả thông tin tổng quát về hệ thống những màn hình cùng với quan hệ về việc chuyển đổi điều khiển giữa chúng.
- Mô tả chi tiết từng màn hình: nhằm mô tả chi tiết nội dung, hình thức trình bày và các thao tác thực hiện trên từng màn hình.

Ví dụ: Liệt kê các phần sau: màn hình, ý nghĩa sử dụng trong Danh sách các thao tác có thể thực hiện:

STT	Thao tác	Ý nghĩa	Xử lý liên quan	Ghi chú
1				
2				

Sơ đồ màn hình



Hình 5.1: Tổ chức các màn hình của phần mềm

Ký hiệu:

Tên màn hình Màn hình với tên tương ứng

→ Chuyển điều khiển đến màn hình khác

Mô tả màn hình giao diện

Các thông tin cần mô tả một màn hình giao diện bao gồm:

Tên màn hình

Đó là *tên của công việc* tương ứng muốn thực hiện trên máy tính.

Nội dung màn hình

Nội dung màn hình bao gồm cấu trúc các thành phần bên trong màn hình. Các thành phần này có thể chia làm 2 loại:

- Thành phần *dữ liệu*: chứa đựng các thông tin liên quan đến công việc đang xét, như:
 - *Thông tin nhập liệu*: loại thông tin do người dùng chịu trách nhiệm cung cấp giá trị (ví dụ ngày lập, hóa đơn, hàng bán ...), có thể là nhập trực tiếp hay dựa vào giá trị định sẵn (có thể sửa nếu muốn) hoặc chọn trong danh sách có trước.

- *Thông tin kết xuất*: loại thông tin này do phần mềm chịu trách nhiệm cung cấp giá trị (ví dụ tổng tiền trả ...).

- Thành phần xử lý: bao gồm các nút điều khiển cho phép người dùng yêu cầu phần mềm thực hiện một xử lý nào đó.

Ngoài ra, nội dung màn hình còn liên quan đến hình thức trình bày, gồm:

- Cách thức bố trí sắp xếp các thành phần trong màn hình (như vị trí, màu sắc, kích thước ...).
- Với màn hình có biểu mẫu liên quan, ta cần trình bày đúng với biểu mẫu tương ứng, hoặc trình bày đúng như yêu cầu của khách hàng.
 - Tuy nhiên ta cần lưu ý trong trường hợp biểu mẫu liên quan chỉ là kết quả cuối cùng cần ghi nhận (trước khi đạt đến kết quả đó), ta cần thực hiện một số công việc trung gian không có biểu mẫu rõ ràng. Khi đó, ta cần bổ sung, sáng tạo hình thức trình bày các màn hình trung gian thể hiện các công việc trung gian.

- Với màn hình không có biểu mẫu liên quan, hình thức trình bày màn hình hoàn toàn là sự sáng tạo khi thiết kế.

Các thao tác có thể thực hiện

Ta cần mô tả hệ thống các thao tác mà người dùng có thể thực hiện trên màn hình cùng với ý nghĩa của chúng. Có rất nhiều loại thao tác khác nhau để có thể cung cấp cho người dùng trên một màn hình giao diện, nhưng quan trọng nhất là việc mô tả thao tác khi người dùng nhấn vào nút điều khiển, nút lệnh hoặc kết thúc việc nhập liệu tại một thành phần nhập liệu nào đó.

5.1.2 Phân loại màn hình giao diện

Quá trình sử dụng phần mềm bao gồm các bước sau:

- Chọn công việc muốn thực hiện trên máy tính,
- Cung cấp các thông tin cần thiết tương ứng với công việc đã chọn,
- Yêu cầu phần mềm thực hiện,
- Xem xét kết quả thực hiện.

Dựa theo quá trình trên, các màn hình giao diện có thể được chia thành nhiều loại tùy theo ý nghĩa sử dụng.

- *Màn hình chính*: nhằm cho phép người dùng sử dụng chọn lựa công việc mong muốn thực hiện trên máy tính từ danh sách các công việc.
- *Màn hình nhập liệu lưu trữ*: nhằm cho phép người dùng thực hiện lưu trữ các thông tin được phát sinh trong thế giới thực.
- *Màn hình nhập liệu xử lý*: nhằm cho phép người sử dụng cung cấp các thông tin cần thiết cho việc thực hiện một công việc nào đó.
- *Màn hình kết quả*: nhằm trình bày cho người sử dụng các kết quả việc thực hiện của một công việc nào đó.
- *Màn hình thông báo*: nhằm thông báo, nhắc nhở người sử dụng trong quá trình thực hiện một công việc nào đó.
- *Màn hình tra cứu*: nhằm cho phép tìm kiếm thông tin đã được lưu trữ với các tiêu chuẩn tìm kiếm.

Một màn hình giao diện có thể thuộc một trong các loại trên, hay cũng có thể tích hợp từ nhiều màn hình cơ sở thuộc vào các loại trên tùy theo bản chất công việc liên quan.

Trong thực tế, hiện nay còn có rất nhiều màn hình khác, nhưng ta quan tâm 3 loại màn hình quan trọng và thông dụng nhất: *màn hình chính*, *màn hình tra cứu*, *màn hình nhập liệu lưu trữ*.

5.1.3 Quá trình thiết kế

Quy trình chung:

Dựa trên yêu cầu chức năng, đầu tiên người thiết kế sẽ xem xét thiết kế các giao diện từ tính đúng đắn, đến tính tiện dụng thỏa yêu cầu về tiện dụng, và xét đến tính hiệu quả nếu yêu cầu về hiệu quả được đưa ra và một số thiết kế theo yêu cầu khác

...

5.1.3.1 Thiết kế giao diện với tính đúng đắn

❖ Sơ đồ màn hình

Giả sử cần thực hiện n công việc trên máy tính, sơ đồ màn hình trong trường hợp này chỉ bao gồm n+1 màn hình sau:

- Một màn hình chính cho phép chọn công việc
- n màn hình liên quan trực tiếp đến n công việc muốn thực hiện

❖ Mô tả chi tiết từng màn hình

Màn hình Chính

Mục tiêu của màn hình này là xác định chính xác nội dung dựa trên danh sách các công việc được yêu cầu và chọn hình thức trình bày đơn giản nhất.

Ví dụ: Phần mềm thư viện

Màn hình chính

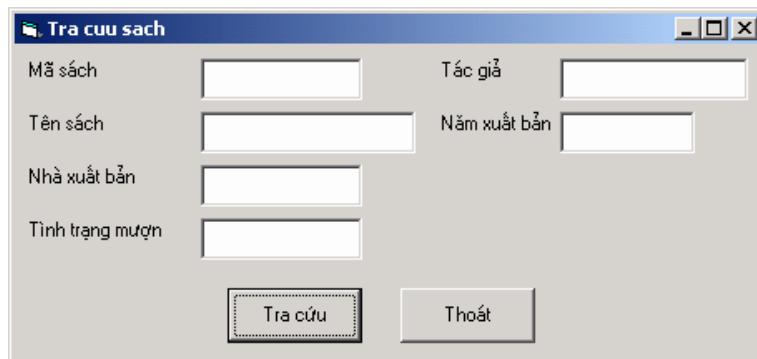
- | | |
|-------------------------|--------------------------------|
| 1. Cho mượn sách | 8. Lập báo cáo về độc giả |
| 2. Nhận trả sách | 9. Nhận sách mới |
| 3. Tìm sách | 10. Thanh lý sách |
| 4. Lập báo cáo mượn trả | 11. Lập báo cáo sách |
| 5. Lập thẻ độc giả | 12. Thay đổi qui định tổ chức |
| 6. Gian hạn thẻ độc giả | 13. Thay đổi quy định mượn trả |
| 7. Tìm độc giả | 14. Thoát |

Đây là thiết kế cho dạng phần mềm thực thi độc lập (desktop-application trên máy tính, hay native-application trên thiết bị di động) có thể hiển thị tất cả danh sách các màn hình, còn đối với dạng phần mềm web ta có thể tùy theo quyền hạn sử dụng để đưa ra màn hình chính có hạn chế bởi các màn hình tương tác cho người sử dụng đó.

Màn hình Tra cứu

Màn hình này giúp người dùng chọn tiêu chuẩn tra cứu đơn giản nhất (chỉ có mã số) và trả về kết quả tìm kiếm đơn giản (có mã số trên hay không).

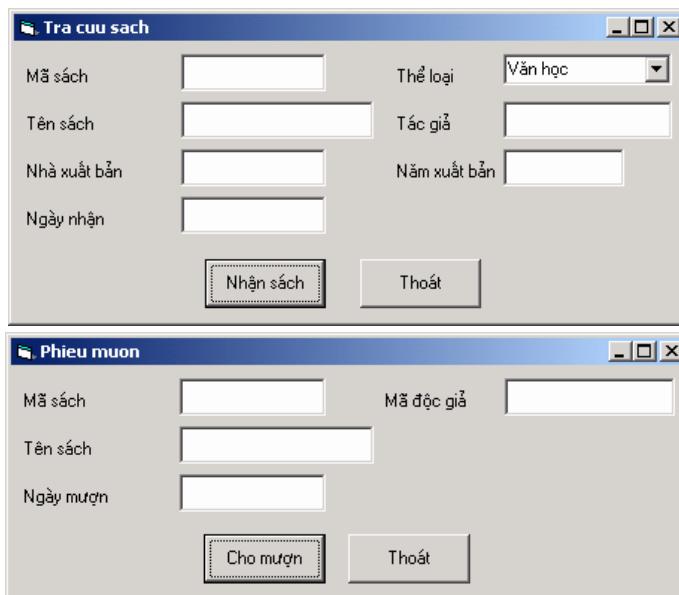
Ví dụ: Tra cứu sách với phần mềm quản lý thư viện



Màn hình Nhập liệu

Trong màn hình này, ta xác định được *chính xác nội dung* dựa trên biểu mẫu hoặc *thông tin liên quan* đến công việc tương ứng và chọn hình thức trình bày đơn giản nhất có thể có (liệt kê tuần tự các nội dung)

Ví dụ: Giao diện cho chức năng nhập sách, mượn sách (quản lý thư viện).



5.1.3.2 Thiết kế giao diện với tính tiện dụng

❖ Sơ đồ màn hình

Ta cần bổ sung vào sơ đồ các màn hình công việc trung gian giúp cho việc sử dụng những màn hình công việc chính dễ dàng hơn, tự nhiên hơn.

❖ Mô tả chi tiết từng màn hình

Màn hình Chính

Màn hình này phân chia các công việc theo từng nhóm tùy theo ý nghĩa và chọn hình thức trình bày tự nhiên nhất có thể có (menu, sơ đồ ...)

Màn hình Tra cứu

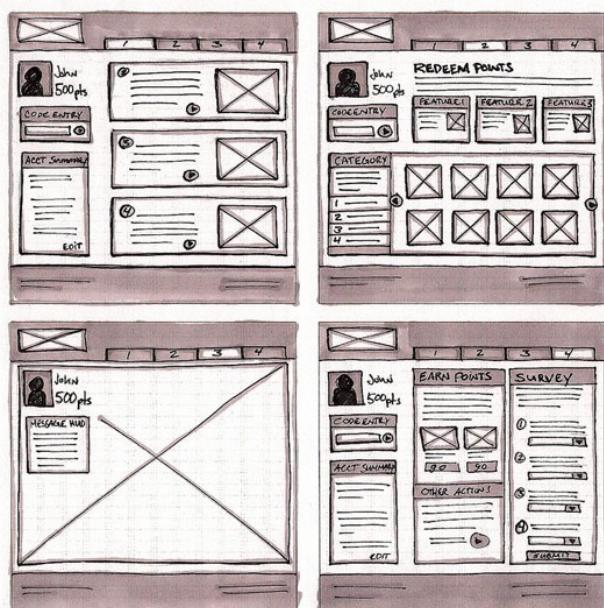
Trong màn hình này, ta mở rộng các tiêu chuẩn tra cứu (các thông tin khác về đối tượng cần tìm) và kết quả tìm kiếm (các thông tin liên quan đến đối tượng khi tìm thấy), sau đó cho phép người dùng xem các kết quả tra cứu dưới nhiều hình thức trình bày khác nhau (các thứ tự khác nhau với một danh sách, các dạng thể hiện biểu đồ, hình ảnh, ...)

Màn hình Nhập liệu

Màn hình này giúp ta chọn dạng trình bày là biểu mẫu liên quan (nếu có) và bổ sung vào đó các thông tin giúp việc sử dụng thuận tiện hơn. Ngược lại, ta cố gắng thiết kế hình thức trình bày tự nhiên nhất có thể có.

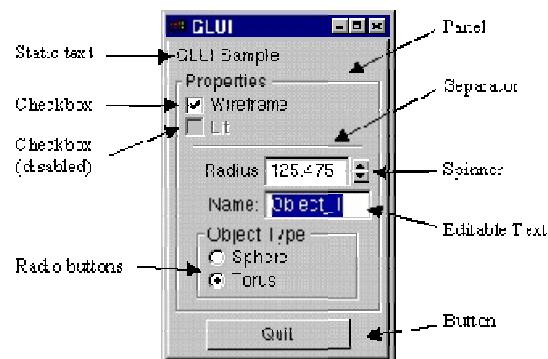
5.1.4 Các vấn đề khác

Đặc tả giao diện: ta thường thực hiện phác thảo đơn giản qua các sơ đồ (gồm cả dạng vẽ tay hay mô phỏng, được gọi là *mock-up*)

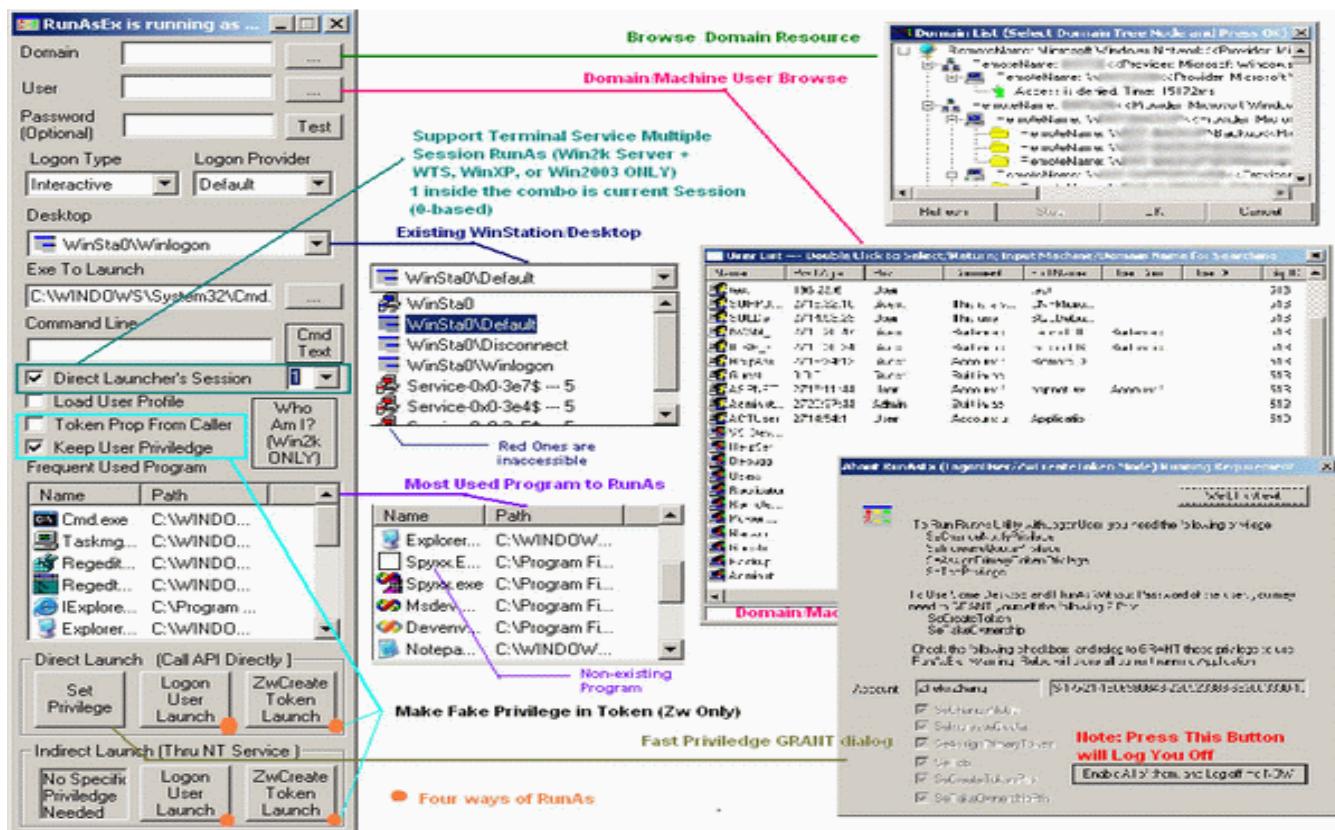


**Hình 5.2: Mô phỏng giao diện chức năng
(nguồn [5])**

Ngoài ra, việc đặc tả giao diện còn chú trọng đến các thành phần chi tiết theo loại thiết bị sử dụng (máy tính, điện thoại ...).



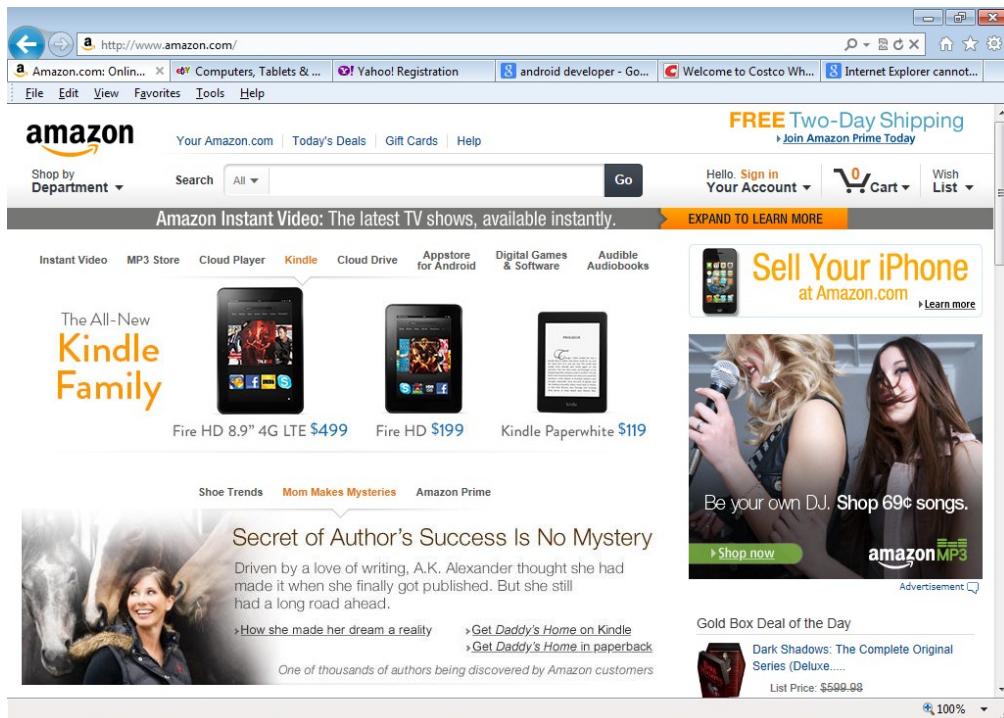
Hình 5.3: Minh họa các thành phần trong thiết kế giao diện (nguồn [5])



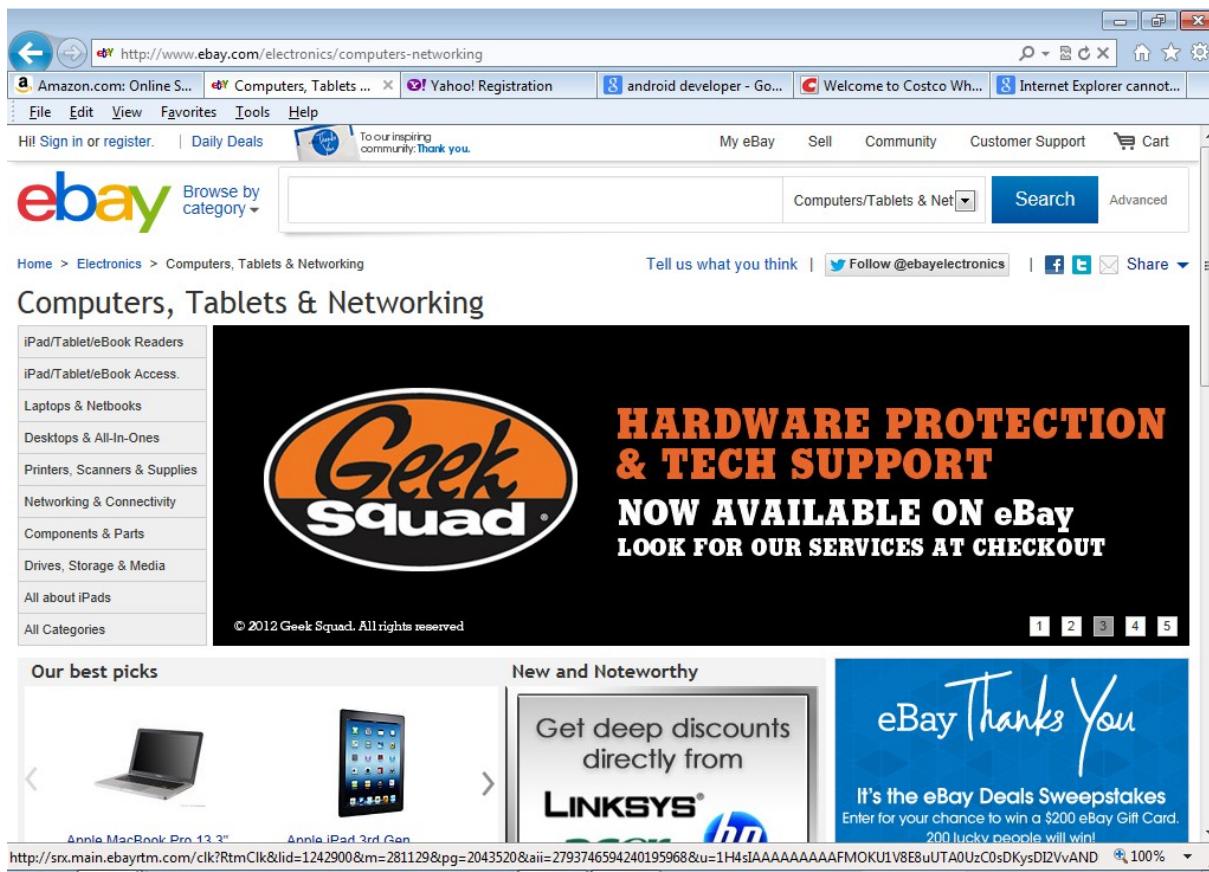
Hình 5.4: Minh họa giao diện chi tiết trên máy tính (nguồn [5])



Hình 5.5: Minh họa giao diện chi tiết trên điện thoại (nguồn [5])



Hình 5.6: Minh họa giao diện dạng web của website Amazon



Hình 5.7: Minh họa giao diện dạng web của website eBay

5.1.5 Một số chú ý chung

❖ Biểu diễn thông tin

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

- Thông tin tĩnh: được khởi tạo ở đầu của mỗi phiên. Nó không thay đổi trong suốt phiên đó và có thể là ở dạng số hoặc dạng văn bản.
- Thông tin động: thay đổi trong cả phiên sử dụng và sự thay đổi này phải được người sử dụng quan sát.

Nếu chúng ta cần hiển thị số lượng lớn thông tin thì nên trực quan hóa dữ liệu. Trực quan hóa có thể phát hiện ra mối quan hệ giữa các thực thể và các xu hướng trong dữ liệu.

Ví dụ: thông tin về thời tiết được hiển thị dưới dạng biểu đồ, trạng thái của mạng điện thoại nên được hiển thị bởi các nút có liên kết với nhau.

❖ Sử dụng màu

Ta thường sử dụng màu trong khi thiết kế giao diện. Màu bổ sung thêm một chiều nữa cho giao diện và giúp cho người sử dụng hiểu được những cấu trúc thông tin phức tạp. Màu có thể được sử dụng để đánh dấu những sự kiện ngoại lệ. Tuy nhiên, khi sử dụng màu để thiết kế giao diện có thể gây phản tác dụng. Do đó, chúng ta nên quan tâm tới một số hướng dẫn sau:

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

Khi người sử dụng tương tác với hệ thống, rất có thể xảy ra lỗi và hệ thống phải thông báo cho người sử dụng biết lỗi gì đã xảy ra hoặc đã có chuyện gì xảy ra với hệ thống. Do đó, thiết kế thông báo lỗi vô cùng quan trọng. Nếu thông báo lỗi nghèo nàn có thể làm cho người sử dụng từ chối hơn là chấp nhận hệ thống.

Vì vậy, thông báo lỗi nên ngắn gọn, súc tích, thống nhất, có cấu trúc. Việc thiết kế thông báo lỗi dựa vào kỹ năng và kinh nghiệm của người sử dụng.

5.2 THIẾT KẾ MÀN HÌNH

5.2.1 Mô tả màn hình chính

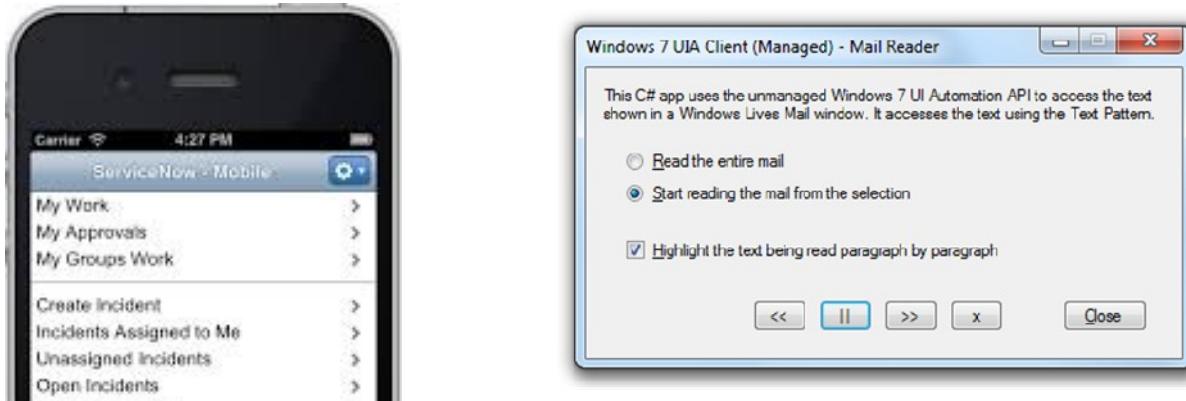
- Ý nghĩa sử dụng: đây là màn hình cho phép người dùng chọn được công việc mà họ muốn thực hiện với phần mềm, thông thường mỗi phần mềm chỉ có một màn hình chính duy nhất.
- Nội dung: gồm danh mục công việc có thể thực hiện với phần mềm
- Hình thức trình bày: bao gồm các thành phần như:
 - *Phím nóng*: nhằm cho phép chọn nhanh một công việc cần thực hiện đối với người sử dụng chuyên nghiệp, thông thường không được sử dụng riêng rẽ mà phải kết hợp với các hình thức khác.
 - *Thực đơn*: nhóm từng công việc theo chức năng (ví dụ lưu trữ, kết xuất), đây là dạng sử dụng thông dụng nhất.
 - *Biểu tượng*: thể hiện công việc trực quan qua biểu tượng (ký hiệu hay hình ảnh tượng trưng cho công việc), tương tự như phím nóng nhưng thông dụng hơn và thường kết hợp với các hình thức khác.
 - *Sơ đồ*: nhằm hiển thị trực quan các đối tượng chính, được thể hiện qua các thao tác trực tiếp trên sơ đồ.
 - *Tích hợp*: nhằm sử dụng đồng thời nhiều hình thức, thường hình thức thực đơn sẽ được ưu tiên trước và kết hợp với nhiều hình thức khác.
- Thao tác người dùng: giúp người dùng chọn công việc trong danh sách công việc được đưa ra bởi phần mềm.

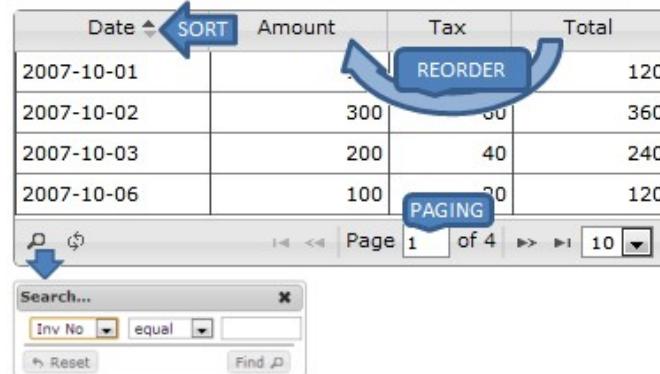
5.2.2 Thiết kế màn hình chính dùng thực đơn (menu)

- Tổ chức của thực đơn: Thực đơn bao gồm nhiều nhóm chức năng (tương ứng nhóm các công việc) mỗi nhóm chức năng bao gồm nhiều chức năng, mỗi chức năng tương ứng với một công việc.
- Phân loại thực đơn: có 3 loại:
 - *Thực đơn hướng chức năng:* là các nhóm chức năng tương ứng với các loại yêu cầu. Ví dụ: (i) Tổ chức: các công việc liên quan đến tổ chức, (ii) Lưu trữ: Các công việc liên quan đến lưu trữ, (iii) Tra cứu: Các công việc liên quan đến tra cứu tìm kiếm
 - *Thực đơn hướng đối tượng:* là các nhóm chức năng tương ứng với các lớp đối tượng. Với sơ đồ lớp gồm n lớp đối tượng, thực đơn sẽ bao gồm (n+1) nhóm chức năng. Trong đó:
 - Một nhóm chức năng tương ứng với đối tượng thế giới thực.
 - n nhóm chức năng tương ứng n lớp đối tượng.
 - *Thực đơn hướng quy trình:* là các nhóm chức năng tương ứng với các giai đoạn trong hoạt động của thế giới thực. Thông thường thế giới thực bao gồm các giai đoạn sau như Tổ chức, Kế hoạch, Tiếp nhận, Hoạt động, Tổng kết.

5.2.3 Ví dụ

Một số dạng màn hình giao diện như sau :





The screenshot shows a user interface for a database or application. At the top, there is a toolbar with a 'SORT' button, a 'REORDER' button, and a 'PAGING' button. Below the toolbar is a grid table with four columns: Date, Amount, Tax, and Total. The data in the grid is as follows:

Date	Amount	Tax	Total
2007-10-01			120
2007-10-02	300	60	360
2007-10-03	200	40	240
2007-10-06	100	20	120

Below the grid is a search bar labeled 'Search...' with dropdown menus for 'Inv No' and 'equal', and a text input field. There are also 'Reset' and 'Find' buttons. At the bottom of the interface is a navigation bar with icons for search, refresh, and help, followed by page navigation buttons ('Page 1 of 4') and a page number input field ('10').

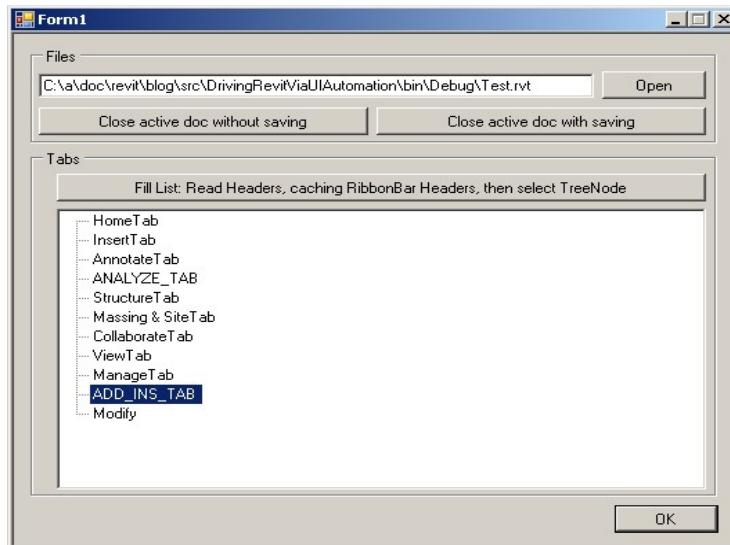
Hình 5.8: Giao diện thực đơn hay tra cứu (nguồn [5])

5.3 THIẾT KẾ MÀN HÌNH TRA CỨU

5.3.1 Mô tả màn hình tra cứu

- Ý nghĩa sử dụng: đây là màn hình cho phép người dùng tìm kiếm và xem các thông tin về các đối tượng.
- Nội dung: gồm có
 - *Tiêu chuẩn tra cứu*: gồm các thông tin được sử dụng cho việc tìm kiếm (thông thường là các thuộc tính).
 - *Kết quả tra cứu*: để biết có tìm thấy hay không, với các thông tin cơ bản về đối tượng tìm kiếm (các thuộc tính) và những thông tin về quá trình hoạt động của đối tượng (quan hệ với các đối tượng khác).
- Hình thức trình bày: gồm có
 - *Tiêu chuẩn tra cứu*: gồm biểu thức luận lý, cây, tích hợp
 - *Kết quả tra cứu*: gồm thông báo, danh sách đơn, xâu các danh sách, cây danh sách.
- Thao tác người dùng: giúp người dùng nhập giá trị cho các tiêu chuẩn tra cứu, yêu cầu bắt đầu tra cứu, xem chi tiết các kết quả tra cứu.

Ví dụ minh họa về màn hình tra cứu:



Hình 5.9: Minh họa màn hình tra cứu (nguồn [5])

5.3.2 Thể hiện tiêu chuẩn tra cứu

- Tra cứu với *biểu thức luận lý*: Tiêu chuẩn này được thể hiện dưới dạng một biểu thức luận lý có dạng như sau:

<Biểu thức luận lý> = <biểu thức cơ sở>

Phép toán luận lý ...

Phép toán AND, OR, NOT, phép so sánh

- Tra cứu với *hình thức cây*: Tiêu chuẩn này được thể hiện qua cây mà các nút chính là các bộ phận trong tổ chức của thế giới thực. Hình thức này rất thích hợp với các thế giới thực có cấu trúc tổ chức phân cấp.
- Tích hợp: Sử dụng đồng thời cả hai hình thức trên.

5.3.3 Thể hiện kết quả tra cứu

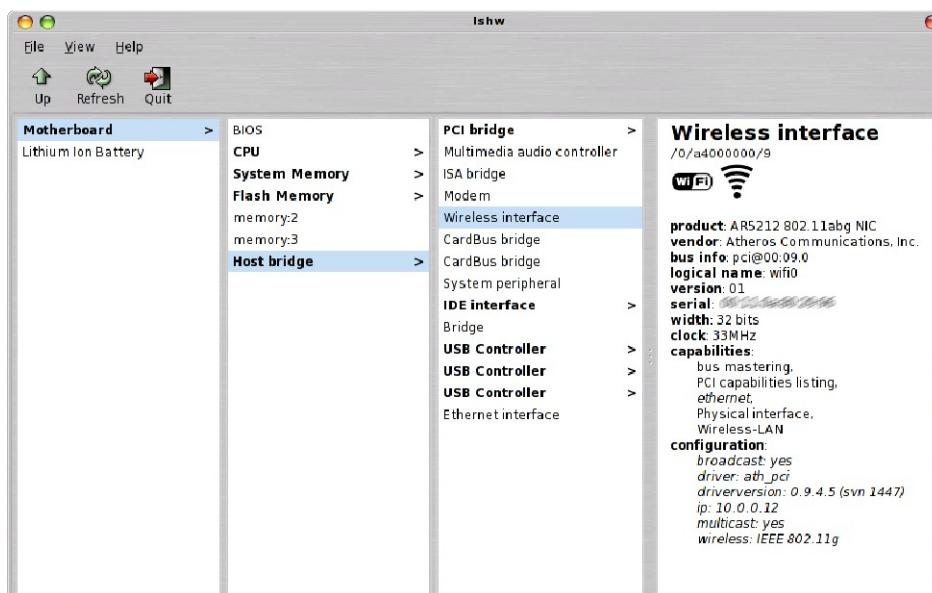
- Kết quả tra cứu dùng *Thông báo*: khi đó kết quả tra cứu chỉ là câu thông báo cho biết có hay không đối tượng cần tìm. Đây là hình thức đơn giản nhất, có tính tiện dụng thấp nhất. Với hình thức này người sử dụng không biết thêm thông tin gì của đối tượng tìm thấy. Ví dụ việc tìm kiếm sách cho thông báo "Hệ thống không có sách bạn cần tìm."
- Kết quả tra cứu dùng *danh sách đơn*: khi đó kết quả tra cứu là danh sách các đối tượng tìm thấy cùng với một số thông tin cơ bản về đối tượng. Hình thức này cho

phép người dùng biết thêm thông tin cơ bản về đối tượng tìm thấy nhưng không biết chi tiết về các hoạt động của đối tượng qua các quan hệ với đối tượng khác.

Item	Category	Name	Status	Price	Last Price	Description	Quantity
EST-1	Fish	Angelfish	Pet	16.50	16.72	Salt Water fish from Australia	9,930
EST-10	Dogs	Dalmatian	Pet	18.50	18.21	Great dog for a Fire Station	9,993
EST-11	Reptiles	Rattlesnake	Pet	18.50		Doubles as a watch dog	10,000
EST-12	Reptiles	Rattlesnake	Pet	18.50	18.16	Doubles as a watch dog	9,979
CGT-13	Reptiles	Iguana	Pet	10.50	10.97	Friendly green friend	9,971
EST-14	Cats	Manx	Pet	58.50		Great for reducing mouse populations	10,000
EST-15	Cats	Manx	Pet	23.50		Great for reducing mouse populations	10,000
EST-16	Cats	Persian	Pet	93.50	90.71	Friendly house cat; doubles as a prin...	9,997
EST-17	Cats	Persian	Pet	93.50	91.63	Friendly house cat; doubles as a prin...	9,947
EST-18	Birds	Amazon Parrot	Pet	193.50		Great companion for up to 75 years	10,000
EST-19	Birds	Finch	Pet	15.50	16.11	Great stress reliever	9,997
EST-2	Fish	Angelfish	Pet	16.50		Salt Water fish from Australia	10,000
EST-20	Fish	Goldfish	Pet	5.50		Fresh water fish from China	10,000
CGT-21	Fish	Goldfish	Pet	5.29		Fresh Water fish from China	10,000
EST-22	Dogs	Labrador Retriever	Pet	135.50		Great hunting dog	10,000
EST-23	Dogs	Labrador Retriever	Pet	145.49		Great hunting dog	10,000
EST-24	Dogs	Labrador Retriever	Pet	255.50		Great hunting dog	10,000
EST-25	Dogs	Labrador Retriever	Pet	325.29		Great hunting dog	10,000
EST-26	Dogs	Chihuahua	Pet	125.50		Great companion dog	10,000
EST-27	Dogs	Chihuahua	Pet	155.29		Great companion dog	10,000
EST-28	Dogs	Golden Retriever	Pet	155.29		Great family dog	10,000

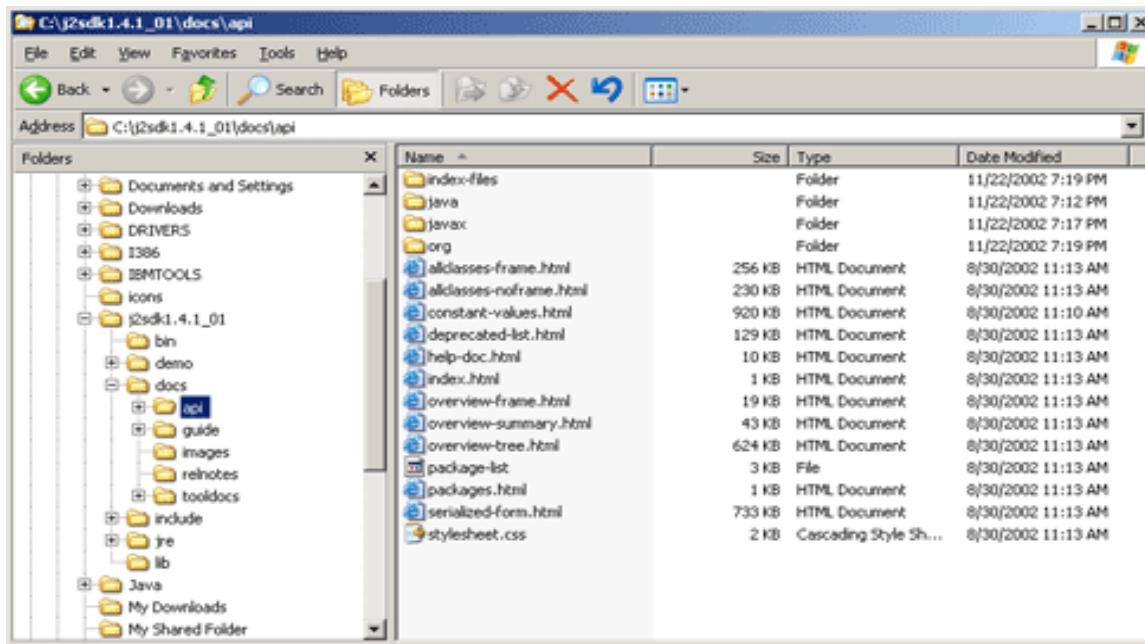
Hình 5.10: Minh họa giao diện danh sách đơn (nguồn [5])

- Kết quả tra cứu dùng *xâu các danh sách*: khi đó kết quả tra cứu gồm nhiều danh sách trong đó danh sách $ds_{(k)}$ chứa các mô tả cho một phần tử trong danh sách $ds_{(k-1)}$ trước đó. Danh sách đầu tiên chính là danh sách đơn trong hình thức trên. Hình thức này cho phép xem các thông tin cơ bản về đối tượng tìm thấy và chi tiết về hoạt động của đối tượng qua các quan hệ với các đối tượng khác.



Hình 5.11: Minh họa giao diện xâu danh sách (nhiều cấp) (nguồn [5])

- **Cây các danh sách:** khi đó kết quả tra cứu là cây mà các nút chính là những danh sách (tương tự giao diện cây thư mục trong cửa sổ phần mềm Windows Explorer), với danh sách tương ứng trong một nút con là các thông tin mô tả chi tiết về một phần tử được chọn trong danh sách của nút cha và danh sách đầu tiên chính là danh sách đơn trong hình thức phía trên. Hình thức này cho phép xem được quá trình hoạt động của đối tượng với nhiều quan hệ, nhiều loại hoạt động khác.



Hình 5.12: Minh họa giao diện cây danh sách

5.3.4 Thao tác người dùng và xử lý của phần mềm

- **Nhập giá trị** cho các tiêu chuẩn tra cứu:
 - Có thể nhập một số hoặc tất cả tiêu chuẩn tra cứu,
 - Với các tiêu chuẩn thường dùng có thể dùng giá trị định sẵn (loại sách thường tìm ...) để tiện dụng hơn cho người dùng.
 - Trong quá trình nhập liệu thông thường, phần mềm không xử lý tính toán, chỉ chờ nhập giá trị cho các tiêu chuẩn tra cứu.
- **Yêu cầu bắt đầu tra cứu:**
 - Nhấn vào nút tra cứu.

- Dựa vào giá trị các tiêu chuẩn tra cứu phần mềm sẽ tiến hành đọc và xuất các kết quả tra cứu tương ứng (xử lý tra cứu).
- Xem xét chi tiết các kết quả tra cứu:
 - Chọn đối tượng cần xem chi tiết trong danh sách kết quả tra cứu.
 - Nhập phạm vi thời gian cần quan sát (thường là thời gian từ ngày ... đến này... hoặc đơn vị thời gian cụ thể tháng ... năm ...).
 - Dựa vào đối tượng được chọn và phạm vi thời gian, phần mềm sẽ đọc và xuất kết quả tra cứu chi tiết hơn theo từng loại hoạt động.
- *Yêu cầu kết xuất:* Có thể bổ sung các nút điều khiển tương ứng với việc in ấn hoặc ghi lên tập tin các kết quả tra cứu. Thông thường mỗi kết quả tra cứu sẽ có một nút riêng, nhưng cũng có thể dùng chung một nút cho mọi kết quả tra cứu (dựa vào kết quả hiện hành). Việc kết xuất thông thường là qua máy in, tuy nhiên cũng có thể cho phép người dùng xác định lại đích của kết xuất (tập tin Excel, trang web,...) tùy theo mục đích sử dụng.

5.4 THIẾT KẾ MÀN HÌNH NHẬP LIỆU

5.4.1 Mô tả màn hình nhập liệu

- Ý nghĩa sử dụng: Đây là màn hình cho phép người dùng thực hiện các công việc có liên quan đến ghi chép trong thế giới thực.
- Nội dung:
 - Các thông tin nhập liệu: Với dạng thông tin này, người dùng chịu trách nhiệm nhập trực tiếp các giá trị, phần mềm sẽ tiến hành kiểm tra tính hợp lệ các giá trị nhập dựa vào quy định liên quan.
 - Các thông tin tính toán: Với dạng thông tin này, phần mềm chịu trách nhiệm tính toán và xuất trên màn hình. Thông thường loại thông tin này giúp việc nhập liệu thuận tiện hơn (nhập số lượng hàng bán khi biết số lượng đang tồn tương ứng, nhập sách mượn khi biết số sách độc giả đang mượn ...).

- Hình thức trình bày: một số hình thức thông dụng gồm:
 - *Danh sách*: là màn hình nhập liệu có dạng danh sách trong thế giới thực (danh sách các thể loại sách, danh sách các lớp học).
 - *Hồ sơ*: là màn hình nhập liệu có dạng hồ sơ với nhiều thông tin chi tiết (hồ sơ học sinh, hồ sơ cầu thủ).
 - *Phiếu*: là màn hình nhập liệu có dạng phiếu với nhiều dòng chi tiết (hóa đơn bán hàng, phiếu nhập hàng ...).
 - *Tích hợp*: màn hình được sử dụng đồng thời các hình thức trên.
- Thao tác người dùng:
 - Có 3 thao tác cơ bản trên màn hình nhập liệu.
 - Nút *Ghi*: để lưu trữ thông tin.
 - Nút *Xóa*: để loại bỏ các thông tin đã lưu trữ.
 - Nút *Tìm*: để tìm và cập nhật lại thông tin đã lưu trữ.
 - Các thao tác khác nhằm tăng tính tiện dụng như:
 - Tạo *phím nóng*: để xác định các phím nóng tương ứng giá trị nhập liệu thường dùng, cho phép tăng tốc độ nhập liệu.
 - Tạo các *nút chuyển điều khiển*: để giúp chuyển điều khiển hiện hành đến màn hình liên quan đến việc nhập liệu hiện hành (bổ sung thể loại sách mới, nhà xuất bản mới ...).

Ví dụ minh họa về màn hình tra cứu:

Trade	Ref	Description	Unit	Quantity	Rate (\$)	Value/Formula	Extension
PR	a	Preliminaries	m2	4	\$45,00		\$120,00
GW	b	Ground Works	m3	264	\$6,50		
GW	c	Ground Works	m2	15	\$8,00		
GW	d	Ground Works	m2	1177	\$8,50		
CN-C	e	Concrete - Formwork	m3	279	\$2,30		
CN-C	f	Concrete - Formwork	m3	10	\$4,50		
CN-F	g	Concrete - Formwork	m	550	\$1,20		
CN-C	h	Concrete - Formwork	m	4	\$3,00		
PR	j	Preliminaries	m2	0	\$0,00		
PR	k	Preliminaries	m2	4	\$45,00		
GW	l	Ground Works	m3	264	\$6,50		
GW	m	Ground Works	m2	15	\$8,00		

Hình 5.13: Minh họa màn hình nhập liệu phức hợp (nguồn [5])

5.4.2 Thiết kế màn hình nhập liệu dạng danh sách

- Sử dụng: Dạng danh sách thích hợp khi cần nhập liệu các bảng danh sách với kích thước nhỏ (danh sách các thể loại sách, các môn học,...).
- Thành phần nhập liệu:
 - Thông tin *nhập liệu*: các thuộc tính của những bảng liên quan
 - Thông tin *tính toán*: mã số thường được tự động phát sinh.
- Thành phần xử lý:
 - *Ghi*: ghi nhận các thay đổi trên danh sách (thêm mới, sửa đổi).
 - *Xóa*: loại bỏ 1 dòng trong danh sách.
 - *Thoát*: quy về màn hình trước đó.
- Các thao tác:
 - Người dùng có thể sửa đổi các thông tin trên các dòng hoặc thêm dòng mới (vào cuối danh sách), chọn và xóa dòng, lưu vào kho.
 - Trong trường hợp đặc biệt, một số thao tác có thể bị cấm (không cho xóa hay đổi thuộc tính ...) tùy vào ý nghĩa cụ thể của danh sách.

5.4.3 Thiết kế màn hình nhập liệu dạng hồ sơ

- Sử dụng: Dạng hồ sơ thích hợp khi cần nhập liệu các hồ sơ các đối tượng trong thế giới thực (hồ sơ học sinh, đội bóng).
- Thành phần dữ liệu:
 - Thông tin nhập liệu: các thuộc tính các bảng liên quan
 - Thông tin tính toán: các mã số thường được tự động phát sinh.
- Thành phần xử lý: Thêm, Ghi, Xóa, Tìm, Thoát
- Các thao tác: Người dùng có thể thêm hồ sơ mới, tìm lại hồ sơ đã lưu trữ, xoá hay sửa đổi các thông tin trên hồ sơ tìm thấy và cuối cùng yêu cầu lưu trữ hồ sơ. Tuy nhiên để tăng tính tiện dụng, một số thao tác chuyển điều khiển có thể được bổ sung cho phép di chuyển nhanh đến các màn hình nhập liệu liên quan khi cần thiết.

5.4.4 Thiết kế màn hình nhập liệu dạng phiếu

- Sử dụng: Dạng phiếu thích hợp khi cần nhập liệu các phiếu ghi nhận thông tin về hoạt động các đối tượng trong thế giới thực.
- Thành phần dữ liệu:
 - Thông tin nhập liệu: các thông tin liên quan đến bảng.
 - Thông tin tính toán: các mã số thường được tự động phát sinh.
- Thành phần xử lý: Thêm, Thêm chi tiết, Ghi, Xóa, Xóa chi tiết, Tìm, Sửa chi tiết, Thoát.

TÓM LƯỢC

Bài học này nhằm cung cấp các kiến thức về:

- *Phân loại màn hình giao diện*
- *Quá trình thiết kế giao diện với tính đúng đắn và tiện dụng*
- *Thiết kế màn hình chính thông thường và dùng thực đơn (menu)*
- *Thiết kế màn hình tra cứu
 - *Mô tả màn hình tra cứu*
 - *Thể hiện tiêu chuẩn tra cứu*
 - *Thể hiện kết quả tra cứu*
 - *Thao tác người dùng và xử lý của phần mềm**
- *Thiết kế màn hình nhập liệu
 - *Mô tả màn hình nhập liệu*
 - *Thiết kế màn hình nhập liệu dạng danh sách*
 - *Thiết kế màn hình nhập liệu dạng hồ sơ*
 - *Thiết kế màn hình nhập liệu dạng phiếu**

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 6: TÁC VỤ XÂY DỰNG CHƯƠNG TRÌNH

Học xong bài này người học sẽ:

- Hiểu được các khái niệm cơ bản về tác vụ hiện thực chương trình, môi trường lập trình, phong cách lập trình, đánh giá chất lượng công việc.
- Áp dụng được các tiêu chí (chất lượng, mô-đun hóa ...) vào phong cách lập trình nhằm tối ưu mã nguồn tạo ra cho hệ thống phần mềm và đảm bảo được chất lượng của mã nguồn nhận được.

6.1 TỔNG QUAN

Trong quá trình thiết kế, ta đã chuẩn bị đầy đủ kiến trúc hệ thống cùng những thành phần thiết kế để từ đó có thể hiện thực được hệ thống dưới dạng các thành phần như mã nguồn, kịch bản, tập tin nhị phân, tập tin thực thi, thư viện, bảng, dữ liệu ... Hơn nữa, việc xây dựng một chương trình có chất lượng tốt sẽ phản ánh những quyết định của thiết kế. (Xem thêm Phụ lục C – Phần D)

Mục tiêu của bước xây dựng chương trình này là bổ sung thêm các thành phần giúp kiến trúc và hệ thống trở thành một khối hoàn chỉnh, cụ thể là:

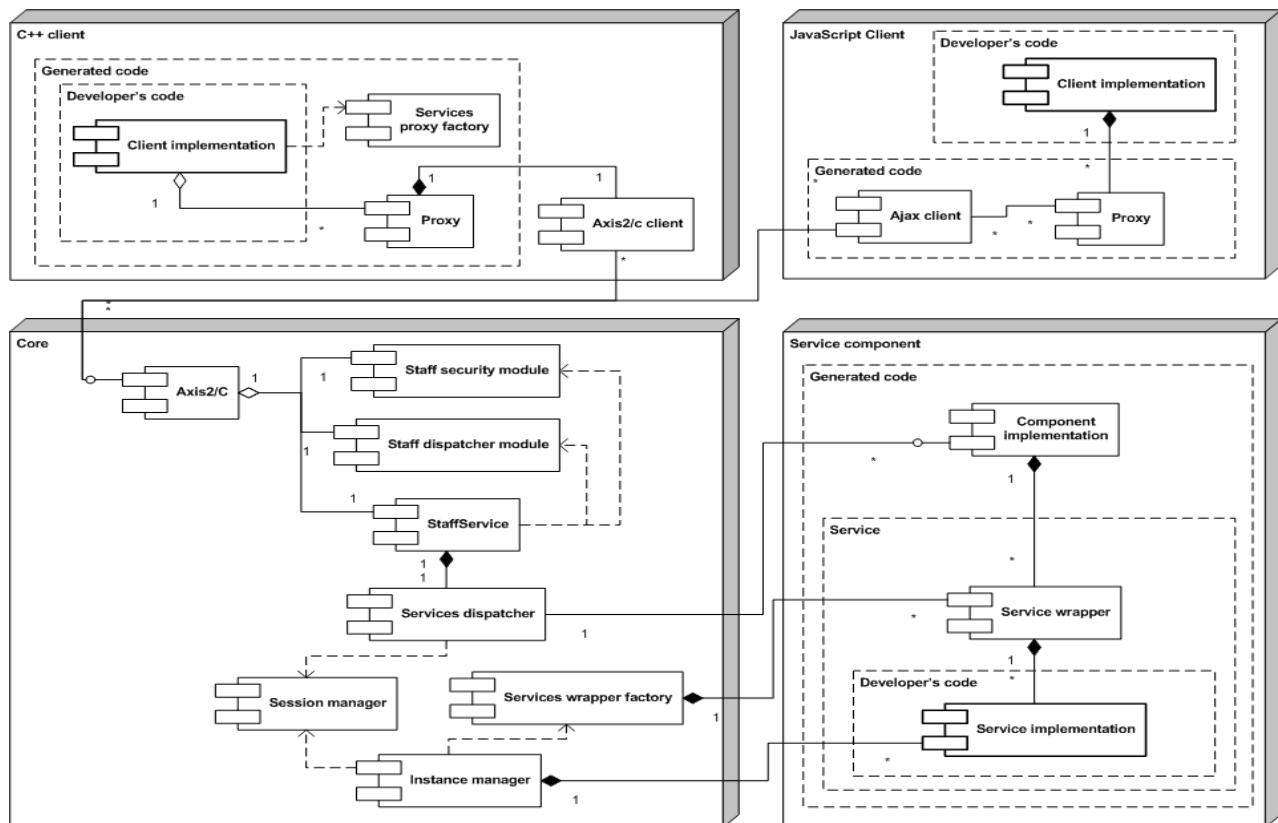
- *Lên kế hoạch tăng cường tích hợp hệ thống (system integration) trong mỗi bước phát triển lặp lại.* Điều này có nghĩa là một hệ thống được cài đặt bởi một dãy các bước nhỏ liên tiếp và có thể quản lý được.
- Triển khai hệ thống bằng cách ánh xạ các thành phần thực thi được vào các nút trong mô hình triển khai. Công việc này chủ yếu dựa vào các lớp động được tìm thấy trong quá trình thiết kế.

- *Xây dựng các lớp thiết kế và những hệ thống con đã tìm được trong quá trình thiết kế.* Đặc biệt, các lớp thiết kế được xây dựng thành những thành phần dạng tập tin mã nguồn.
- Kiểm thử đơn vị các thành phần, rồi tích hợp chúng bằng các biên dịch và liên kết chúng lại với nhau thành một hoặc nhiều thành phần thực thi được, sau đó kiểm thử tích hợp và kiểm thử hệ thống.

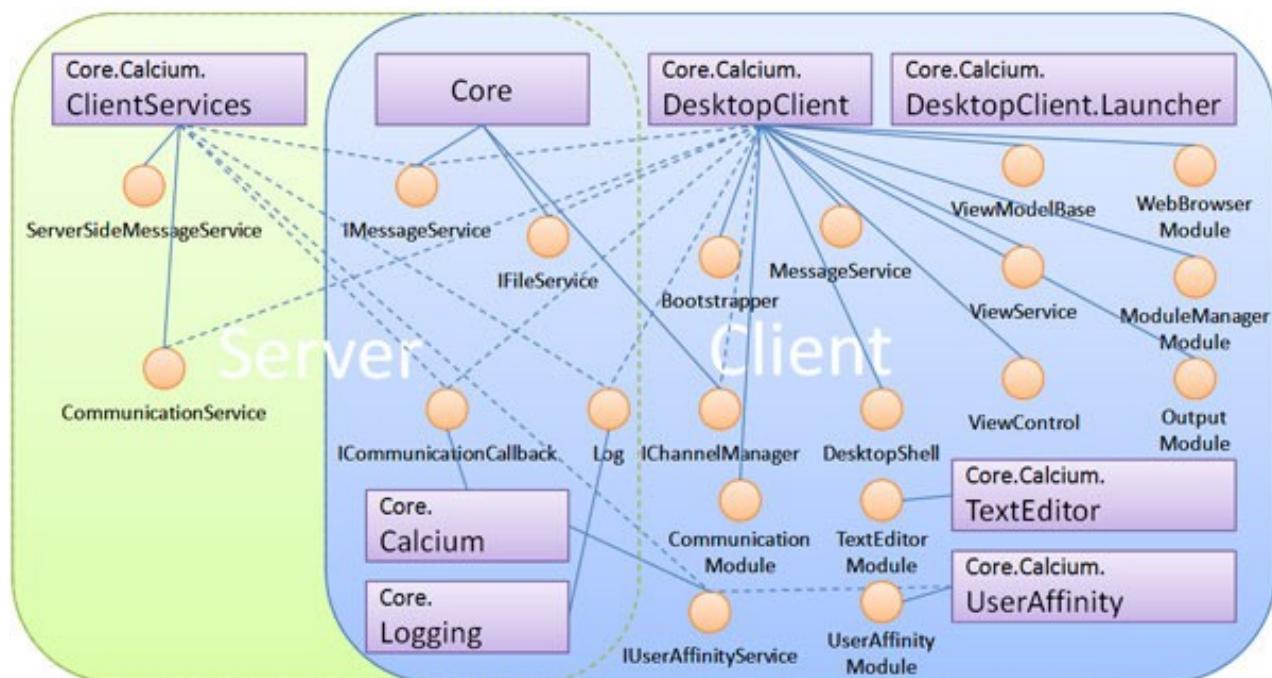
Việc xây dựng chương trình cần dựa theo các tiêu chí sau:

- Cấu trúc chương trình, cấu trúc dữ liệu cùng những định nghĩa được chọn lựa và thiết lập trong suốt quá trình thiết kế thủ tục cần được tổ chức tốt để dễ dàng nhận biết trong quá trình xây dựng chương trình,
- Mức trừu tượng của thiết kế về lớp đối tượng, mô-đun, thuật toán, cấu trúc dữ liệu, kiểu dữ liệu cũng phải linh động trong quá trình thực hiện,
- Giao diện giữa các thành phần của hệ thống phần mềm được mô tả rõ ràng trong thực hiện,
- Quá trình thực hiện cũng được kiểm tra độ tin cậy của đối tượng và thao tác với trình biên dịch (trước khi kiểm tra chương trình thực sự).
- Đảm bảo những đặc trưng ở trên phụ thuộc vào việc chọn lựa ngôn ngữ thực hiện và kiểu lập trình.

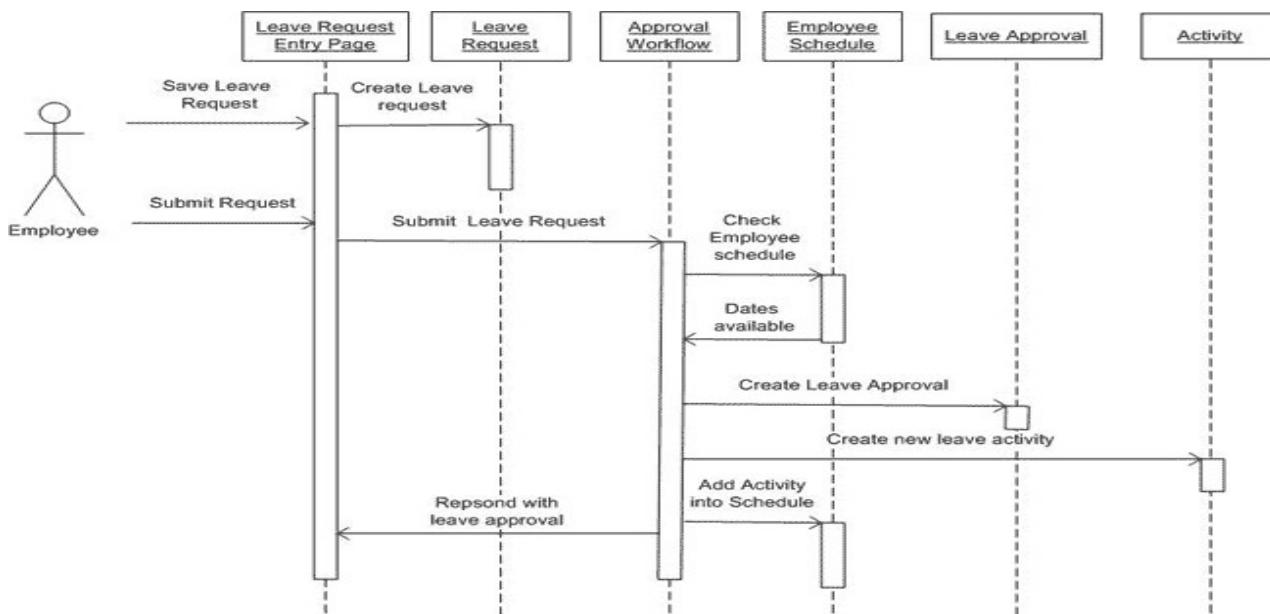
Ví dụ minh họa về cấu trúc thành phần của phần mềm cần hiện thực.



Hình 6.1: Minh họa sơ đồ thành phần của phần mềm (nguồn [5])



Hình 6.2: Minh họa tác các thành phần của phần mềm (nguồn [5])



Hình 6.3: Minh họa sơ đồ tuần tự của tác vụ thực hiện (nguồn [5])

6.2 MÔI TRƯỜNG LẬP TRÌNH

Việc chọn lựa “đúng” ngôn ngữ lập trình cho quá trình hiện thực hóa là vấn đề quan trọng trong quy trình lập trình. Mức lý tưởng, việc thiết kế nên được thực hiện độc lập và tránh bất kỳ liên quan nào đến dạng ngôn ngữ sẽ được dùng khi hiện thực sau đó, nhằm tạo ra bản thiết kế có thể thực hiện được trên ngôn ngữ lập trình bất kỳ. Việc chọn được ngôn ngữ lập trình phù hợp cho quá trình xây dựng phần mềm cần được căn cứ trên các tiêu chí sau:

6.2.1 Tiêu chí về Chất lượng của ngôn ngữ lập trình

Tiêu chí chất lượng này được thể hiện qua những yếu tố sau:

Tính mô-đun hóa	Luồng điều khiển	Khả năng tích hợp
Giá trị của tài liệu	Tính hiệu quả Hỗ trợ hộp thoại	Tính khả chuyển
Cấu trúc dữ liệu	Yếu tố ngôn ngữ chuyên biệt	

6.2.2 Tiêu chí về Khả năng mô-đun hóa của ngôn ngữ lập trình

Tiêu chí này nhấn mạnh về khả năng mô-đun hóa, tức là mức độ hỗ trợ việc phân chia chương trình thành những thành phần độc lập nhỏ hơn. Việc phác thảo một chương trình lớn thành nhiều mô-đun là điều kiện tiên quyết để thực thi trong dự án phần mềm.

Không có khả năng mô-đun hóa thì việc phân chia công việc trong giai đoạn thực hiện trở nên không khả thi. Những chương trình đơn nhất trở nên không thể quản lý vì chúng khó có thể bảo trì và tài liệu kỹ thuật và chúng thực hiện với thời gian biên dịch dài.

Nếu một ngôn ngữ lập trình hỗ trợ phát triển một chương trình thành những phần nhỏ, chúng phải đảm bảo những thành phần phải hoạt động với nhau. Nếu một thủ tục được thực thi ở mô-đun khác, cũng được kiểm tra để biết thủ tục đó có thực sự tồn tại và nó có được sử dụng chính xác hay không (nghĩa là số tham số và kiểu dữ liệu là chính xác).

6.2.3 Tiêu chí về Giá trị tài liệu kỹ thuật của ngôn ngữ lập trình

Tiêu chí này nhấn mạnh về khả năng có thể đọc và bảo trì của chương trình thông qua tài liệu kỹ thuật. Điều này càng quan trọng đối với những chương trình lớn hay phần mềm mà khách hàng vẫn tiếp tục phát triển.

Tài liệu kỹ thuật của một chương trình mang lại giá trị cao hơn kết quả đạt được hiện hành của chương trình đó, vì tài liệu kỹ thuật sẽ được tham khảo nhiều lần và được khai thác triệt để nhằm phục vụ quá trình bảo trì và nâng cấp cho chương trình đó. Hơn nữa, do nhiều ngôn ngữ mở rộng với quá nhiều chức năng chuyên biệt, nếu thiếu tài liệu kỹ thuật, ta sẽ khó hiểu được tất cả chi tiết bên trong của chương trình nên có thể dẫn đến việc hiểu hay giải thích sai ý tưởng của chúng.

6.2.4 Tiêu chí về Cấu trúc dữ liệu trong ngôn ngữ lập trình

Khi hiện thực hóa chương trình, các dữ liệu phức tạp cần được xử lý hoàn chỉnh. Khả năng (của ngôn ngữ lập trình) sẵn sàng hỗ trợ hiện thực cấu trúc dữ liệu sẽ đóng vai trò quan trọng trong quá trình hiện thực chương trình.

Ví dụ: họ ngôn ngữ C cho phép khai báo con trỏ đối với cấu trúc dữ liệu, điều này cho phép cấu trúc dữ liệu phức tạp, phạm vi và cấu trúc của chúng có thể thay đổi ở thời điểm thực thi, tuy nhiên không nghiêm ngặt trong truy xuất (khi so sánh với Java).

Trong dự án lớn với nhiều nhóm dự án, dữ liệu trừu tượng là một vấn đề trọng tâm và mang ý nghĩa cụ thể. Ngôn ngữ lập trình hướng đối tượng có những đặc trưng mở rộng về loại kiểu dữ liệu trừu tượng, nhằm cho phép hiện thực hóa những hệ thống

phần mềm phức tạp. Đối với những giải pháp mở rộng và uyển chuyển, ngôn ngữ lập trình hướng đối tượng cung cấp tùy chọn đặc biệt tốt hơn ngôn ngữ lập trình thông thường khác.

6.2.5 Ví dụ

Ví dụ: Ta xét giai đoạn thực hiện phần mềm quản lý thư viện (các giai đoạn trước đã được minh họa ở các chương trước), cụ thể là:

- Ngôn ngữ lập trình lựa chọn: Visual Basic / C++ / C# hay Java ...
- Phần mềm cơ sở dữ liệu: MS Access / SQL Server hay Oracle ...
- Hệ thống lớp đối tượng: tạo lập lớp đối tượng (THU_VIEN, DOC_GIA, SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó,
- Hệ thống giao diện: tạo các giao diện (màn hình chính, màn hình lập thẻ, màn hình cho mượn sách, màn hình nhận sách, màn hình trả sách) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó,
- Hệ thống lưu trữ: tạo lập cấu trúc cơ sở dữ liệu (các bảng THU_VIEN, DOC_GIA, SACH, MUON_SACH) theo mô tả của phần thiết kế trong môi trường cụ thể nào đó.

6.3 PHONG CÁCH LẬP TRÌNH

Sau khi hiện thực và kiểm tra chương trình, hệ thống phần mềm hiếm khi được sử dụng trong một thời gian dài mà không có sửa đổi hay điều chỉnh. Điều này thực sự đúng vì khi một yêu cầu của phần mềm được cập nhật hoặc mở rộng (sau khi hoàn chỉnh sản phẩm hay trong suốt quá trình thực hiện thao tác), ta thường khó phát hiện ra lỗi hay những thiếu sót phát sinh. Vì vậy, giai đoạn hiện thực phần mềm chắc chắn phải được sửa đổi và mở rộng, đòi hỏi lặp lại việc đọc hiểu chương trình nguồn nhiều lần. Trong trường hợp lý tưởng, chức năng của một thành phần chương trình được tìm hiểu chỉ dựa trên chương trình nguồn mà không được cung cấp thông tin nào từ tài liệu thiết kế. Tuy nhiên, chương trình nguồn chỉ là một dạng tài liệu phản ánh hiện trạng của thực thi.

Khả năng “đọc được” của một chương trình (tức là ta có thể đọc hiểu được mã nguồn của chương trình đó) phụ thuộc vào ngôn ngữ lập trình được dùng và phong

cách lập trình của người thực hiện. Việc viết một chương trình để có thể đọc được là một tiến trình sáng tạo. Phong cách lập trình của người thực hiện sẽ ảnh hưởng đến khả năng đọc được của chương trình hơn là ngôn ngữ lập trình được sử dụng.

Các yếu tố quan trọng nhất của phong cách lập trình tốt là:

6.3.1 Tính cấu trúc

Tính chất này thể hiện bằng việc:

- Phân rã một hệ thống phần mềm dựa trên độ phức tạp của nó thông qua mức trừu tượng của thành phần, tạo cấu trúc chương trình lớn,
- Hoặc chọn lựa những thành phần chương trình phù hợp trong việc định ra những thuật toán của thủ tục con, tạo cấu trúc chương trình nhỏ.

6.3.2 Ưu điểm của diễn đạt

Quy trình hiện thực một hệ thống phần mềm bao gồm việc đặt tên đối tượng và mô tả các công việc thực thi của đối tượng này. Việc chọn lựa tên sẽ đặc biệt quan trọng khi viết thuật toán.

❖ Một số đề nghị

- Với một hệ thống, ta gán tên chỉ dựa trên một ngôn ngữ (ví dụ dùng dùng lẵn lộn tiếng Anh và tiếng Việt).
- Nếu dùng chữ viết tắt, ta nên sử dụng tên đặt này để giúp người đọc chương trình có thể hiểu mà không cần bất cứ sự giải thích nào. Việc sử dụng những từ viết tắt chỉ bao gồm ngữ cảnh.
- Dùng chữ hoa và chữ thường để phân biệt các loại định nghĩa khác nhau (như chữ hoa đầu tiên cho kiểu dữ liệu, lớp, mô-đun, chữ thường đầu tiên cho biến), đặt tên dài hơn để dễ hiểu (như CheckInputValue).
- Dùng danh từ cho giá trị, động từ cho hoạt động, và thuộc tính cho điều kiện để làm rõ ý nghĩa nhận diện (như width, ReadKey, valid).
- Thiết lập những qui luật để sử dụng một cách thích hợp.

- Ghi chú rõ ràng và đầy đủ các diễn giải sử dụng, nhằm đóng góp cho khả năng đọc được của chương trình. Hiệu chỉnh việc ghi chú chương trình không dễ dàng và đòi hỏi kinh nghiệm, sáng tạo và khả năng diễn đạt thông điệp gọn gàng và chính xác.

❖ Một số luật cho cách tạo ghi chú

- Mỗi thành phần hệ thống (mô-đun hay lớp) nên bắt đầu với ghi chú chi tiết để cung cấp cho người đọc những thông tin liên quan đến thành phần của hệ thống như:
 - Thành phần này làm gì?
 - Thành phần này được sử dụng ra sao trong những ngữ cảnh gì?
 - Những phương thức đặc biệt được sử dụng.
 - Ai là tác giả của thành phần này?
 - Thành phần này được viết khi nào?
 - Những sửa đổi cập nhật nó được thực hiện.
- Mỗi thủ tục và phương thức nên được cung cấp các ghi chú mô tả công việc. Điều này cần đặc biệt phần mềm cho phần đặc tả giao diện.
- Ghi chú để giải thích ý nghĩa của biến và hằng.
- Gán nhãn và ghi chú cho các thành phần có những tác nhiệm riêng.
- Những khối lệnh khó hiểu (hoặc thành phần hay thủ tục rắc rối) nên được mô tả ghi chú sao cho người đọc dễ dàng hiểu chúng.
- Phản ánh trong ghi chú những cập nhật về các thay đổi của chương trình liên quan cả phần khai báo và những khối lệnh thành phần.

Việc tuân thủ những luật trên cần được cân nhắc vì không có luật áp dụng đồng nhất cho tất cả các hệ thống phần mềm và mỗi phạm vi phần mềm. Việc tạo ghi chú cho hệ thống phần mềm là một nghệ thuật cũng giống như phần thiết kế cài đặt hệ thống phần mềm.

6.3.3 Cách thức trình bày bên ngoài

Ngoài việc chọn tên và tạo ghi chú, khả năng đọc được của hệ thống phần mềm cũng phụ thuộc vào cách thức trình bày bên ngoài.

Một số luật đề nghị cho hình thức trình bày chương trình gồm:

- Mỗi thành phần của chương trình, những khai báo (của kiểu dữ liệu, hằng biến ...) nên được tách biệt mỗi phần của khối lệnh.
- Phần khai báo nên có một cấu trúc đồng nhất khi có thể như thứ tự sau: hằng, kiểu dữ liệu, lớp, mô-đun, phương thức và thủ tục.
- Mô tả giao diện (danh sách tham số cho phương thức và thủ tục) nên tách tham số nhập liệu, kết xuất và nhập/xuất.
- Phần ghi chú và chương trình nguồn nên tách biệt.
- Cấu trúc của chương trình nên được nhấn mạnh ở phần canh chỉnh lề (sử dụng phím tab cho từ mỗi đầu khối lệnh đến khối lệnh theo sau).

6.4 ĐÁNH GIÁ CHẤT LƯỢNG CÔNG VIỆC

6.4.1 Hiện thực tăng cường

Ý tưởng cơ bản của việc hiện thực tăng cường gần với việc kết hợp giai đoạn thiết kế và hiện thực hơn là tách biệt hai giai đoạn này, như mô hình quy trình phát triển tuần tự cổ điển đề ra. Phương pháp này cho thấy những quyết định trong thiết kế và cài đặt có tác động lẫn nhau, nếu ta tách biệt thiết kế rời rạc thì sẽ không đạt được mục tiêu tăng cao chất lượng công việc.

Việc hiện thực tăng cường nghĩa là sau mỗi bước thiết kế có liên quan đến kiến trúc, thì kiến trúc phần mềm hiện hành được thẩm định dựa trên những trường hợp thực tế. Theo đó, sự tác động qua lại giữa các thành phần trong hệ thống cụ thể – trong thiết kế và trong hình thức đặc tả giao diện – cần được thẩm định. Để có thể làm được điều này, những thành phần hệ thống (với hành vi xuất/nhập) được mô phỏng hay thực tế hóa như khuôn mẫu. Nếu có những nghi ngờ liên quan đến tính khả thi của thành phần thì tiến trình thiết kế được ngắt và những thành phần được thực hiện. Chỉ khi hiện thực và nhúng chúng vào trong kiến trúc hệ thống (đã được

kiểm tra trước đó) thì tiến trình thiết kế mới được tiếp tục, hoặc kiến trúc được chấp nhận tương ứng với kiến thức thu được trong khi hiện thực thành phần.

Hiệu quả của phương pháp này phụ thuộc vào việc mở rộng khả năng tích hợp những thành phần hệ thống (được hoàn chỉnh theo chuẩn khác nhau ở cấp độ khác nhau) đối với toàn bộ hệ thống để hiện thực gần với thực tế. Vài thành phần hệ thống (như giao diện người dùng và mô hình dữ liệu) được thể hiện dưới dạng khuôn mẫu, còn các thành phần khác (thư viện có sẵn, hay tồn tại như các hiện thực hoàn chỉnh) được thể hiện dưới dạng mã nguồn thực thi, còn các thành phần hệ thống khác có sẵn được thể hiện như đặc tả giao diện. Đối với sự hợp lệ của thiết kế hệ thống hiện hành, khi giao diện người dùng được triển khai thì tương ứng khuôn mẫu cũng cần được kích hoạt.

6.4.2 Đánh giá lại thiết kế và chương trình (Design and Code Review)

Việc xem lại (*review*) thiết kế và chương trình sẽ giúp ta hoàn chỉnh chất lượng hiệu quả của công việc hơn là chỉ điều chỉnh những thay đổi đơn lẻ trong quá trình phát triển phần mềm. Trong các phần mềm lớn, vấn đề quan trọng là nhu cầu xem xét lại những yêu cầu, đặc tả, thiết kế, và cả chương trình của ta, để từ đó giúp ta điều chỉnh thiếu sót, luận lý, cấu trúc, tính sáng tạo. Khi chương trình không rõ hay mơ hồ xáo trộn, việc thêm ghi chú hay viết lại nó một cách đơn giản hơn sẽ làm cho chương trình dễ đọc và dễ hiểu.

Mục đích của việc xem lại là để đảm bảo chương trình tạo ra đạt chất lượng cao nhất. Một số dạng xem lại là thao tác kiểm duyệt, duyệt qua, xem xét mục riêng từ thiết kế đến từng dòng lệnh. Việc xem lại có thể được dùng trên yêu cầu, thiết kế, hồ sơ tài liệu hay bất kỳ yếu tố của sản phẩm.

Trong thực tế, nhiều dự án phần mềm đã được triển khai đến tận giai đoạn kiểm thử nhưng thiếu công tác xem lại, điều này không thực sự hiệu quả. Xem lại thiết kế và chương trình là các cách thức hiệu quả để tìm và sửa chữa thiếu sót. Bằng việc xem lại, ta có thể tìm ra trong chương trình những thiếu sót trực tiếp, các lỗi mà ta cần cập nhật hoàn chỉnh và chính xác hơn.

Công việc xem lại cho phép ta quay trở lại bất kỳ việc gì đã làm trước đó trong chương trình. Nhóm phát triển nên cùng nhau đọc lại mọi thiết kế và chương trình, rồi nghiên cứu để hiểu chúng tường tận nhằm sửa những sai sót về luận lý, cấu trúc hay tính rõ ràng, sau đó có thể viết lại chương trình hay thêm ghi chú và viết lại hoàn chỉnh cho những phần nội dung không rõ ràng để làm cho chúng dễ đọc và dễ hiểu hơn.

6.5 VÍ DỤ MINH HỌA

Ví dụ: Xét phần mềm hỗ trợ giải bài tập phương trình đại số với yêu cầu là Soạn đề bài, Soạn đáp án, Giải bài tập, Chấm điểm. Các giai đoạn thực hiện trong quy trình bao gồm:

❖ **Giai đoạn 1: Xác định yêu cầu**

- Yêu cầu 1: Soạn đề bài với mô tả và quy tắc về Soạn đề bài
- Yêu cầu 2: Soạn đáp án với mô tả, quy tắc, biểu mẫu về Soạn đáp án.
- Yêu cầu 3: Giải bài tập với mô tả, quy tắc và biểu mẫu Giải bài tập.
- Yêu cầu 4: Chấm điểm với mô tả, quy tắc về Chấm điểm.

❖ **Giai đoạn 2: Sơ đồ luồng công việc cho 4 chức năng**

❖ **Giai đoạn 3: Phân tích yêu cầu chức năng**

❖ **Giai đoạn 4: Thiết kế phần mềm**

❖ **Giai đoạn 5: Thực hiện phần mềm**

- Hệ thống Lớp đối tượng: tạo các lớp đối tượng SACH_BAI_TAP, BAI_TAP theo mô tả thiết kế trong môi trường cụ thể (VB, C#, Java),
- Hệ thống giao diện: tạo (vẽ) giao diện (màn hình chính, màn hình soạn đề bài, màn hình soạn đáp án, màn hình giải bài tập, màn hình chấm điểm) theo mô tả của thiết kế trong môi trường cụ thể (VB, C#, Java),
- Hệ thống lưu trữ: tạo cấu trúc cơ sở dữ liệu (các bảng SACH_BAI_TAP, BAI_TAP, BAI_GIAI, BUOC_GIAI) theo mô tả của phần thiết kế trong một môi trường cụ thể nào đó (MS Access / SQL Server, Oracle ...)

❖ **Giai đoạn 6: Kiểm chứng phần mềm xem ở bài sau**

TÓM TẮT

Bài học này cung cấp các kiến thức về:

- *Môi trường lập trình và các tiêu chí của ngôn ngữ lập trình (Chất lượng, Khả năng mô-đun hóa, Giá trị tài liệu kỹ thuật, Cấu trúc dữ liệu)*
- *Phong cách lập trình và các vấn đề liên quan (Tính cấu trúc, Ưu điểm của diễn đạt, Cách thức trình bày bên ngoài)*
- *Đánh giá chất lượng công việc thông qua việc Hiện thực tăng cường và Đánh giá lại thiết kế và chương trình.*

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 7: TÁC VỤ KIỂM THỬ PHẦN MỀM

Học xong bài này người học sẽ nắm được các nội dung sau:

- Hiểu được các khái niệm cơ bản về kiểm thử, yêu cầu đối với kiểm thử, các kỹ thuật kiểm thử, chiến lược kiểm thử & các giai đoạn.
- Vận dụng được các phương pháp kiểm thử (hộp đen, hộp trắng ...) và triển khai được các chiến lược kiểm thử (đơn vị, chức năng, tích hợp ...) để giúp đảm bảo chất lượng của sản phẩm phần mềm.

7.1 TỔNG QUAN

7.1.1 Lịch sử

Sự tách biệt giữa việc gỡ lỗi (sửa lỗi, debugging) với kiểm thử (testing) lần đầu tiên được Glenford J. Myers đưa ra vào năm 1979. Mặc dù sự quan tâm của ông là kiểm thử sự gián đoạn ("một kiểm thử thành công là tìm ra được một lỗi"), nhưng điều đó minh họa mong muốn của cộng đồng công nghệ phần mềm để tách biệt các hoạt động phát triển cơ bản, giống như việc tách phần gỡ lỗi ra riêng khỏi quá trình kiểm thử. Vào năm 1988, Dave Gelperin và William C. Hetzel đã phân loại các giai đoạn và mục tiêu trong kiểm thử phần mềm theo trình tự sau:

- Trước 1956: Hướng về việc kiểm soát lỗi.
- 1957-1978: Hướng về chứng minh lỗi.
- 1979-1982: Hướng về tính phá hủy của lỗi.
- 1983-1987: Hướng về đánh giá lỗi.
- 1988-2000: Hướng về việc phòng ngừa lỗi.

Một nghiên cứu được tiến hành bởi NIST trong năm 2002 cho biết rằng các lỗi phần mềm gây tổn thất cho nền kinh tế Mỹ 59,5 tỷ đô mỗi năm, hơn một phần ba chi

phí này có thể tránh được nếu việc kiểm thử phần mềm được thực hiện tốt hơn. Theo đó, một kiểm khuyết nếu được tìm ra sớm hơn thì chi phí để sửa chữa nó sẽ rẻ hơn. Bảng dưới đây cho thấy chi phí sửa chữa các khuyết khuyết tùy thuộc vào giai đoạn nó được tìm ra. Ví dụ, một vấn đề được tìm thấy sau khi đã ra bản phần mềm chính thức rồi sẽ có chi phí gấp 10-100 lần khi giải quyết vấn đề từ lúc tiếp nhận yêu cầu. Với sự ra đời của cách thức triển khai thực tiễn liên tục và các dịch vụ dựa trên đám mây, chi phí tái triển khai và bảo trì có thể làm giảm bớt theo thời gian.

Chi phí sửa chữa một khuyết khuyết		Thời gian phát hiện				
		Yêu cầu phần mềm	Kiến trúc phần mềm	Xây dựng phần mềm	Kiểm thử hệ thống	Sau khi phát hành
Thời gian sử dụng	Yêu cầu phần mềm	1x	3x	5-10x	10x	10-100x
	Kiến trúc phần mềm	-	1x	10x	15x	25-100x
	Xây dựng phần mềm	-	-	1x	10x	10-25x

7.1.2 Giới thiệu

Kiểm thử phần mềm là việc tiến hành thí nghiệm để so sánh kết quả thực tế với lý thuyết nhằm mục đích phát hiện lỗi. (Xem thêm Phụ lục C – Phần E)

Bộ thử nghiệm (*test cases*) là dữ liệu dùng để kiểm tra hoạt động của chương trình. Bộ kiểm thử tốt là bộ có khả năng phát hiện ra lỗi của chương trình. Khi tiến hành kiểm thử, ta chỉ có *thể chứng minh được sự tồn tại của lỗi* nhưng **không chứng minh được** rằng trong chương trình không có lỗi.

Nội dung của bộ thử nghiệm gồm:

- Tên mô-đun/chức năng muốn kiểm thử
- Dữ liệu vào:
 - Dữ liệu của chương trình: số, xâu ký tự, tập tin,
 - Môi trường thử nghiệm: phần cứng, hệ điều hành,
 - Thứ tự thao tác (kiểm thử giao diện)

- Kết quả mong muốn
 - Thông thường: số, xâu ký tự, tập tin ...
 - Màn hình, thời gian phản hồi
- Kết quả thực tế

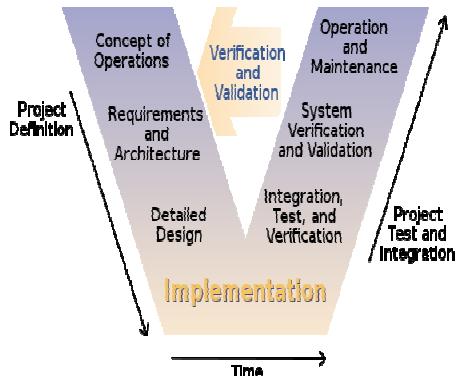
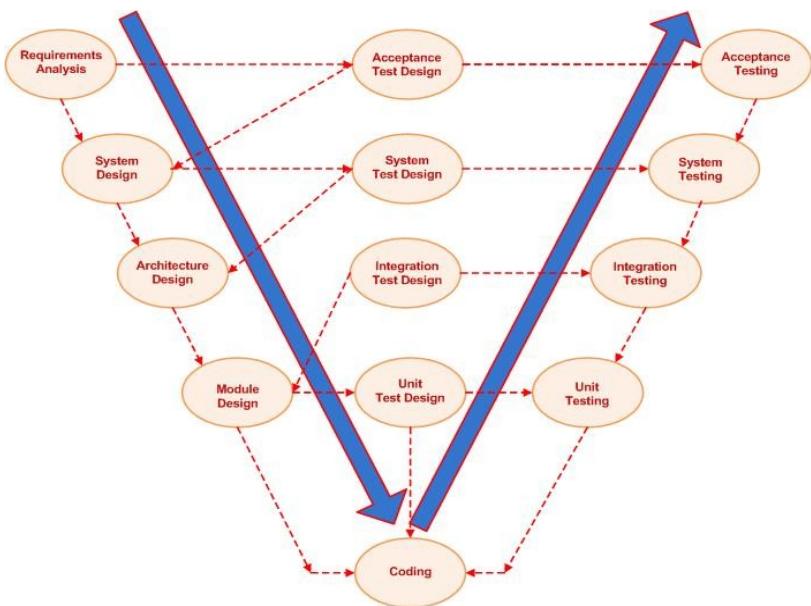
Không gian thử nghiệm là tập các bộ số thử nghiệm. Không gian này nói chung là rất lớn. Nếu ta có thể vét cạn được không gian thử nghiệm thì chắc chắn qua việc xử lý của phép kiểm tra đơn vị thì sẽ chương trình sẽ không còn lỗi. Tuy nhiên điều này không khả thi trong thực tế. Do đó khi đề cập đến tính đúng đắn của phần mềm ta dùng khái niệm độ tin cậy.

Phương pháp kiểm thử là cách chọn bộ số thử nghiệm để tăng cường độ tin cậy của đơn vị cần kiểm tra. Nói cách khác, phương pháp kiểm thử là cách phân hoạch không gian thử nghiệm thành nhiều miền rồi chọn bộ số liệu thử nghiệm đại diện cho miền đó. Như vậy ta cần tránh trường hợp mọi bộ thử nghiệm đều rơi vào một miền kiểm tra.

7.1.3 Mô hình chữ V

Mô hình này (h7.1 và h7.2) biểu diễn sự liên hệ giữa kiểm thử và các tác vụ khác, đồng thời đại diện cho quá trình phát triển phần mềm (hay phần cứng) và có thể được xem là một mở rộng của mô hình thác nước⁴.

⁴ Xem thêm tại [http://en.wikipedia.org/wiki/V-Model_\(software_development\)](http://en.wikipedia.org/wiki/V-Model_(software_development))

**Hình 7.1: Mô hình chữ V****Hình 7.2: Mô hình chữ V chi tiết**

Mô hình chữ V thể hiện mối quan hệ giữa mỗi giai đoạn của vòng đời phát triển và giai đoạn liên quan của thử nghiệm. Các trục ngang và dọc đại diện cho thời gian hay hoàn thiện dự án (từ trái qua phải) và mức độ trừu tượng (thô ở tầng trừu tượng cao nhất), tương ứng. Mô hình chữ V hướng đến:

- Thực hiện *Xác nhận* (Verification) với tác vụ phân tích yêu cầu, thiết kế hệ thống (kiến trúc, mô-đun ...) ở mức cao và mức chi tiết
- Kiểm tra *Hợp lệ* (Validation) với các tác vụ kiểm thử (đơn vị, tương tác, hệ thống, chấp nhận, triển khai)

7.2 YÊU CẦU ĐỐI VỚI KIỂM THỬ

Ta cần chú ý các yêu cầu sau đây:

- Tính lặp lại:
 - Kiểm thử phải lặp lại được (kiểm tra lỗi đã được sửa hay chưa)
 - Dữ liệu/trạng thái phải mô tả được
- Tính hệ thống: phải đảm bảo đã kiểm tra hết các trường hợp.
- Được lập tài liệu: phải kiểm soát được tiến trình/kết quả.

7.3 CÁC KỸ THUẬT KIỂM THỬ

7.3.1 Phương pháp Hộp đen (Black box)

Phương pháp kiểm thử này là dạng kiểm thử chức năng (*functional test*) chỉ dựa trên bản đặc tả các chức năng. Do đó, ta chỉ chú tâm đến *phát hiện các sai sót về chức năng mà không quan tâm đến cách hiện thực cụ thể*. Với phương pháp này ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của mô-đun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

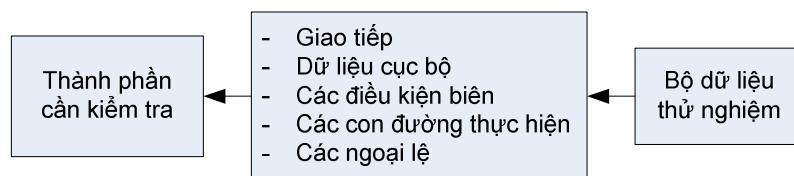
Do không thể kiểm thử mọi trường hợp trên thực tế, ta sẽ chia nhỏ không gian thử nghiệm dựa vào giá trị nhập xuất của đơn vị cần kiểm tra. Ứng với mỗi vùng dữ liệu, ta sẽ thiết kế những bộ thử nghiệm tương ứng và đặc biệt là các bộ thử nghiệm tại các giá trị biên của vùng dữ liệu.

Để kiểm chứng chương trình giải phương trình bậc 2 theo phương pháp hộp đen, ta sẽ phân chia không gian thử nghiệm thành 3 vùng là "Vô nghiệm", "Có nghiệm kép" hay "Có 2 nghiệm riêng biệt".

Sau khi đã kiểm tra thử với các bộ thử nghiệm đã thiết kế, ta cần mở rộng bộ thử nghiệm cho các trường hợp đặc biệt như: biên của số nguyên trong máy tính (32767, -32768), số không, số âm, số thập phân, dữ liệu sai kiểu, dữ liệu ngẫu nhiên.

7.3.2 Phương pháp Hộp trắng (White box)

Phương pháp này hướng đến việc kiểm thử cấu trúc trong đó ta sẽ chia không gian thử nghiệm dựa vào cấu trúc của đơn vị cần kiểm tra.



Hình 7.3: Cấu trúc hộp trắng

Trong đó:

- *Kiểm tra giao tiếp của đơn vị* là để đảm bảo dòng thông tin vào ra đơn vị luôn đúng (đúng giá trị, khớp kiểu ...)

- *Kiểm tra dữ liệu cục bộ* để đảm bảo dữ liệu được lưu trữ trong đơn vị toàn vẹn trong suốt quá trình thuật giải được thực hiện. Ví dụ: nhập dữ liệu sai, tên biến sai, kiểu dữ liệu không nhất quán, ràng buộc hay ngoại lệ.
- *Kiểm tra các điều kiện biên* của các câu lệnh điều kiện và vòng lặp để đảm bảo đơn vị luôn chạy đúng tại các biên này.
- Kiểm tra để đảm bảo mọi *con đường thực hiện* phải được đi qua ít nhất một lần. Con đường thực hiện của một đơn vị chương trình là một dãy có thứ tự các câu lệnh bên trong đơn vị đó sẽ được thực hiện khi kích hoạt đơn vị. Ví dụ: Con đường thực hiện của p1 và p2 như sau:

Thủ tục 1	Thủ tục 2	Khi đó, ta có các đường thực hiện sau:
Lệnh 1	Lệnh 1	Thủ tục 1: Lệnh 1 → Lệnh 2 → Lệnh 3 → Lệnh 4
Lệnh 2	Nếu <i>đk</i> thì Lệnh 2	Thủ tục 2: Lệnh 1 → Lệnh 2 → Lệnh 3 và Lệnh 1 →
Lệnh 3	Ngược lại thì Lệnh 3	Lệnh 2 → Lệnh 4
Lệnh 4	Lệnh 4	

7.3.3 Phương pháp Hộp xám (Grey box)

Phương pháp này liên quan đến hiểu biết về cấu trúc dữ liệu bên trong và các thuật toán cho mục đích của các bài kiểm thử thiết kế. Khi thực hiện những bài kiểm thử với người dùng hoặc mức độ hộp đen, nhóm kiểm tra không nhất thiết phải truy cập vào mã nguồn của phần mềm. Ta có thể thao tác với dữ liệu đầu vào và định dạng đầu ra không xác định (như hộp xám), bởi vì đầu vào và đầu ra rõ ràng ở bên ngoài của hộp đen mà chúng được hệ thống gọi ra trong quá trình kiểm thử. Sự phân biệt này đặc biệt quan trọng khi tiến hành kiểm thử tích hợp giữa hai mô-đun được viết mã bởi hai nhóm phát triển khác nhau, chỉ có các giao diện được bộc lộ ra để kiểm thử.

Tuy nhiên, với các kiểm thử có yêu cầu thay thế một kho lưu trữ dữ liệu bên dưới hệ thống (*back-end*) – như cơ sở dữ liệu hay tập tin đăng nhập – không xác định như hộp xám, người dùng sẽ không thể thay đổi các kho lưu trữ dữ liệu trong khi sản phẩm vẫn đang hoạt động bình thường.

Kiểm thử hộp xám cũng có thể bao gồm kỹ thuật đảo ngược để xác định đối tượng, giá trị biên hoặc các thông báo lỗi.

Khi biết được những khái niệm cơ bản về cách thức các phần mềm hoạt động như thế nào, nhóm kiểm tra thực hiện kiểm thử phần mềm từ bên trong tốt hơn so với bên ngoài. Thông thường, nhóm kiểm thử theo phương pháp Hộp Xám sẽ được phép thiết lập một môi trường kiểm thử đã bị cô lập với các hoạt động liên quan cơ sở dữ liệu. Các kiểm thử có thể quan sát trạng thái của sản phẩm được kiểm thử (sau khi thực hiện hành động nhất định giống như việc thực hiện các câu lệnh SQL đối với cơ sở dữ liệu) và thực hiện truy vấn để đảm bảo những thay đổi dự kiến đã được phản ánh. Kiểm thử hộp xám thực hiện kịch bản kiểm thử thông minh, dựa trên thông tin hạn chế. Điều này sẽ đặc biệt áp dụng cho các kiểu xử lý dữ liệu, kể cả xử lý ngoại lệ.

7.4 CHIẾN LƯỢC & CÁC GIAI ĐOẠN

Với những dự án phần mềm lớn, những người tham gia được chia thành:

- *Nhóm thứ nhất:* gồm những người tham gia trong dự án phát triển phần mềm. Nhóm này chịu trách nhiệm kiểm tra các đơn vị của chương trình để chắc chắn chúng thực hiện đúng theo thiết kế.
- *Nhóm thứ hai:* gồm các chuyên gia tin học nhưng độc lập và không thuộc nhóm thứ nhất. Nhóm này có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chủ quan còn để lại.

Ví dụ minh họa về kiểm thử phần mềm:

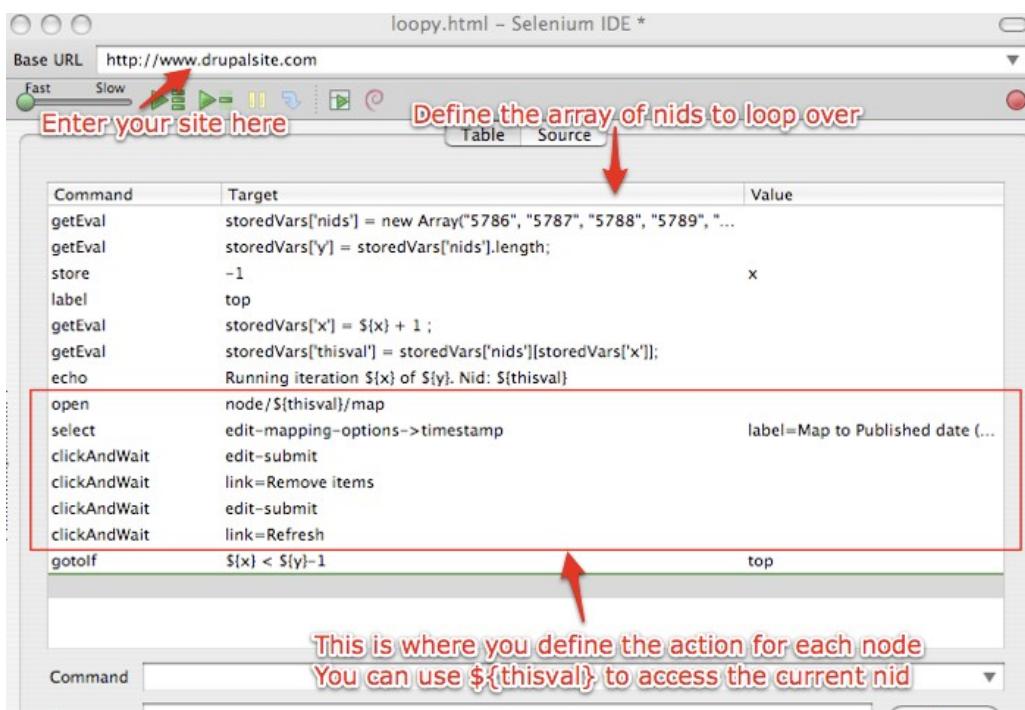
Goal	Fully automated test of an entity bean	Manual test of an entity bean	Manual test of an session bean with real database	Manual test of an session bean with mock objects
Test	BaseEntityFixture	PoJoFixture	BaseSessionBeanFixture	MockedSessionBeanFixture
Ejb3Unit Features	Fields and relations are filled by data generators Tests provided - write of all fields - getter / setter - nullable fields - equals() - hashCode()	Access to entity manager or datasource Service methods - persist() of complex object graph - findAll() - deleteAll()	Access to entity manager or datasource Dependency Injection (DataSource, EntityManager, Session Context, TimerService) Calling of Life-Cycle methods for Creation	Dependency Injection of mocked objects - EJB - Resource, - PersistenceContext - DataSource - EntityManager - SessionContext - TimerService
User action	- name entity to test - configure the generators	- write test cases - configure „Initial Data Sets“ - cleanup data	- write test cases - configure „Initial Data Sets“ - cleanup data	- write test cases - write expectations

Hình 7.4: Phân tích hướng tiếp cận kiểm thử phần mềm (nguồn [5])

```

▼ ✓ Run test suite All tests 11 out of 11 tests passed, 1.039 seconds
  ▼ ✓ Run test suite /Users/tonyarnold/Library/Developer/Xcode/DerivedData/Ki...
    ▼ ✓ Run test suite DWKBookListViewControllerSpec 9 out of 9 tests passed, :
      ✓ Run test case example example
      ✓ Run test suite KWSSpec 0 out of 0 tests passed, 0.000 seconds
      ✓ Run test suite KWTestCase 0 out of 0 tests passed, 0.000 seconds
    ▼ ✓ Run test suite NSStringFileUtilsSpec 2 out of 2 tests passed, 0.007 secor...
      ✓ Run test case example example
      ✓ Run test case example example

```

Hình 7.5: Phân tích kết quả kiểm thử phần mềm (nguồn [5])**Hình 7.6: Minh họa kịch bản của ca kiểm thử tự động (nguồn [5])**

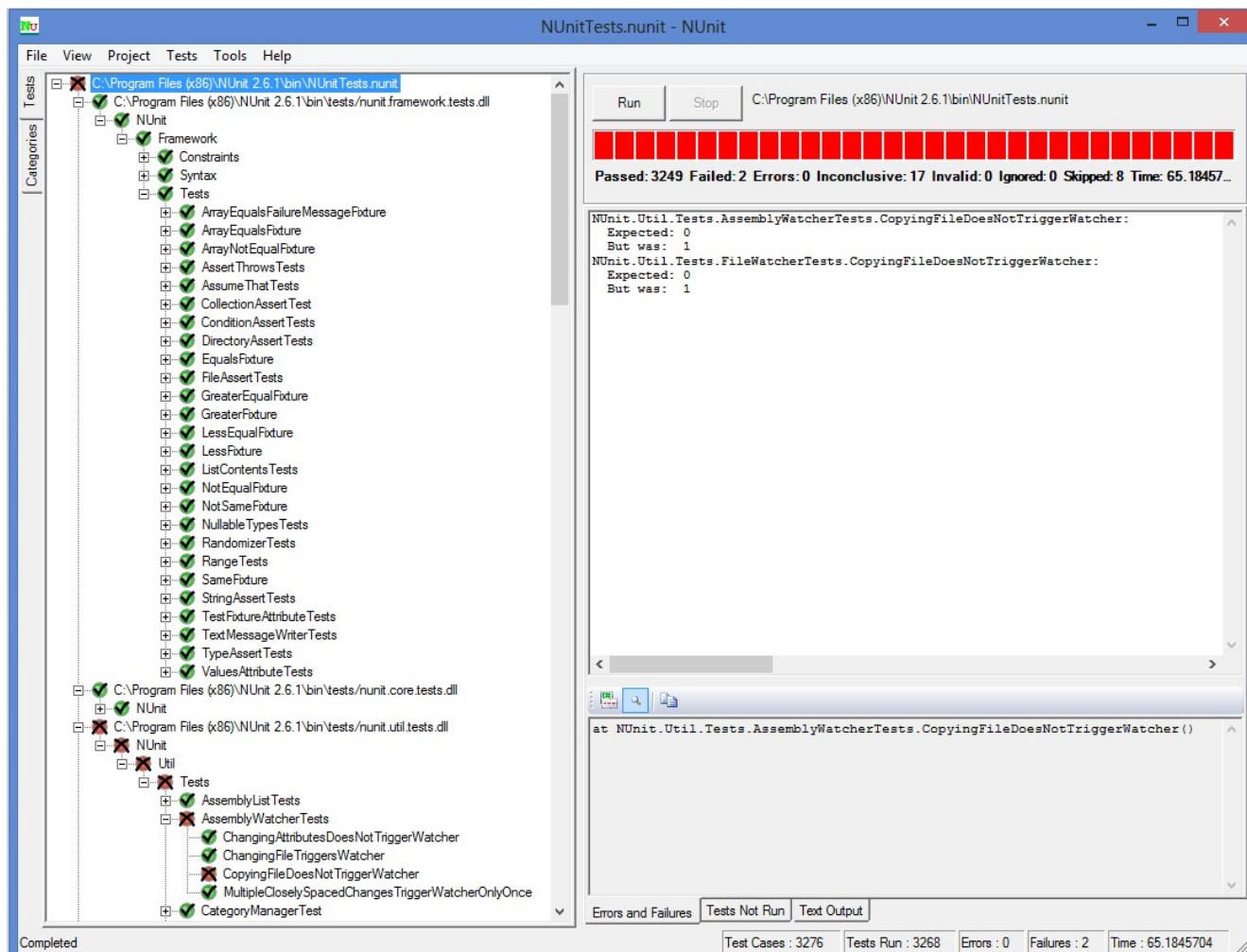
Việc kiểm thử chương trình được tiến hành qua các giai đoạn sau:

7.4.1 Kiểm thử Đơn vị (Unit Test)

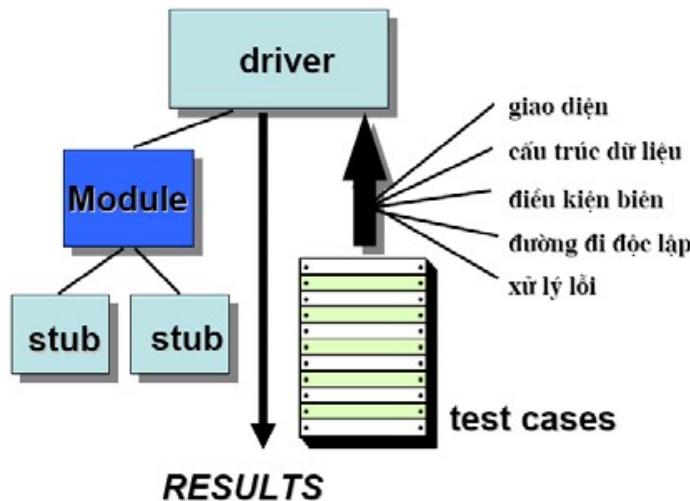
Giai đoạn này *sử dụng kỹ thuật hộp trắng* và dựa vào *hồ sơ thiết kế* để xây dựng các bộ thử nghiệm *sao cho khả năng phát hiện lỗi là lớn nhất*.

Vì đơn vị (*unit*) được kiểm tra thường là một đoạn chương trình hay một thủ tục nhưng không phải là 1 chương trình đầy đủ, hơn nữa đơn vị đó có thể được gọi thực thi bởi những đơn vị khác (hoặc gọi đến những đơn vị khác để thực thi), nên dù chương trình đã được hoàn tất đầy đủ các đơn vị, ta cũng *không nên giả thuyết sự tồn tại* hoặc *tính đúng đắn* của các đơn vị khác mà phải xây dựng các mô-đun giả lập đơn vị gọi (đặt tên là *driver*) và đơn vị bị gọi (đặt tên là *stub*):

- *Driver* đóng vai trò như một chương trình chính nhập các bộ số thử nghiệm và gởi chúng đến đơn vị cần kiểm tra đồng thời nhận kết quả trả về của đơn vị cần kiểm tra.
- *Stub* là chương trình giả lập thay thế các đơn vị được gọi bởi đơn vị cần kiểm tra. Stub thực hiện các thao tác xử lý dữ liệu đơn giản như in ấn, kiểm tra dữ liệu nhập và trả kết quả ra.



Hình 7.7: Minh họa quá trình kiểm thử đơn vị (nguồn [5])



Hình 7.8: Mô phỏng chương trình giả lập

7.4.2 Kiểm thử Chức năng (Functional Test)

Như trình bày trong phần “Phương pháp Hộp đen”, giai đoạn này thực hiện việc kiểm tra khả năng thực thi của các chức năng có đáp ứng được đầy đủ những yêu cầu (đã nêu trong bản đặc tả) của các chức năng đó hay không. Vì thế, ta tập trung phát hiện các sai sót về chức năng mà không quan tâm đến cách hiện thực cụ thể. Với phương pháp này ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của mô-đun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

7.4.3 Kiểm thử Tích hợp (Integration Test)

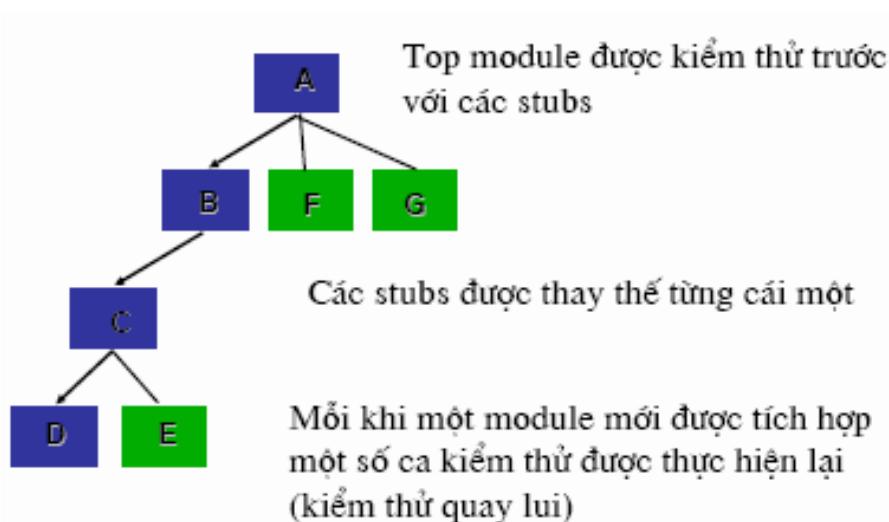
Giai đoạn này được tiến hành sau khi đã hoàn tất công việc kiểm thử chức năng cho từng mô-đun riêng lẻ bằng cách tích hợp các mô-đun này lại với nhau. Mục đích của giai đoạn này là kiểm tra giao diện của các đơn vị và sự kết hợp hoạt động giữa chúng, kiểm tra tính đúng đắn so với đặc tả, kiểm tra tính hiệu quả, với phương pháp thực hiện chủ yếu sử dụng kiểm tra chức năng dựa trên chiến lược từ trên xuống, hoặc từ dưới lên.

❖ Hướng xử lý Trên Xuống (Top-Down)

Thuật giải của hướng tiếp cận này gồm những bước sau:

- Sử dụng mô-đun chính như 1 driver và các stub được thay cho tất cả các mô-đun là con trực tiếp của mô-đun chính,

- Lần lượt thay thế các stub bởi các mô-đun thực sự,
- Tiến hành kiểm tra tính đúng đắn,
- Một tập hợp bộ thử nghiệm được hoàn tất khi hết stub,
- Kiểm tra lùi có thể được tiến hành để đảm bảo không phát sinh lỗi mới,



Hình 7.9: Phương pháp kiểm thử từ trên xuống

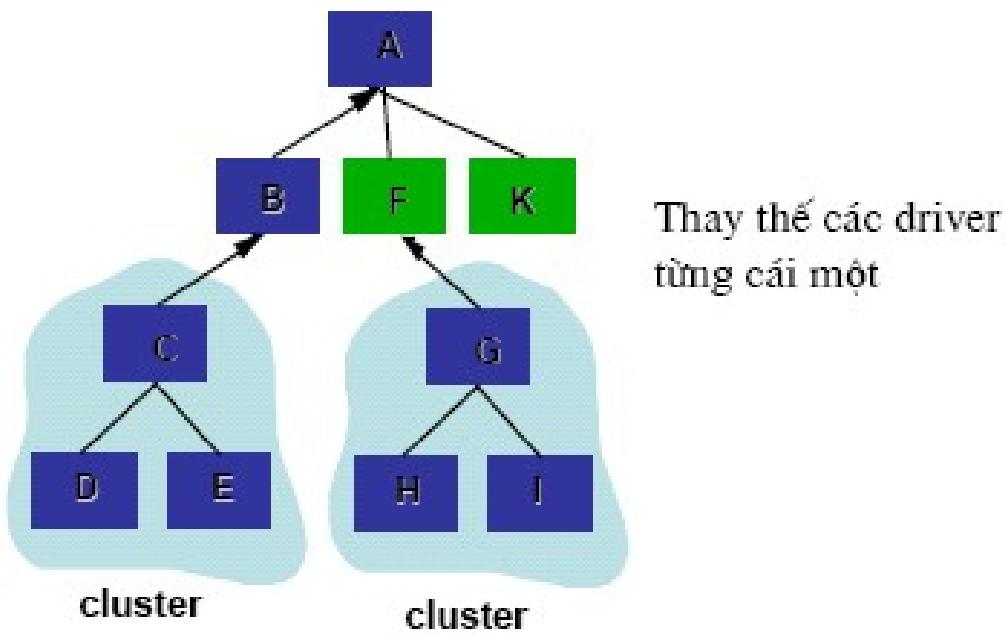
- **Ưu điểm:**
 - Kiểm thử trên xuống kết hợp với phát triển trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và làm giảm giá thành sửa đổi.
 - Nhanh chóng có phiên bản thực hiện với các chức năng chính. Có thể thẩm định tính dùng được của sản phẩm sớm.
- **Nhược điểm:** nhiều mô-đun cấp thấp rất khó mô phỏng: thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp...

❖ Hướng xử lý Dưới Lên (Bottom-Up)

Ta kiểm tra mô-đun lá (cấp thấp nhất) trước, do đó ta không cần phải viết stub. Thuật giải của hướng này là:

- Các mô-đun cấp thấp được nhóm thành từng nhóm (thực hiện cùng chức năng),
- Viết driver điều khiển tham số nhập xuất,
- Bỏ driver và gắn chùm vào mô-đun cao hơn.

Kiểm thử dưới lên



Hình 7.10: Phương pháp kiểm thử từ dưới lên

- **Ưu điểm:** Tránh xây dựng các mô-đun tạm thời phức tạp, tránh sinh các kết quả nhân tạo, thuận tiện cho phát triển các mô-đun để dùng lại.
- **Nhược điểm:** Chậm phát hiện lỗi kiến trúc, chậm có phiên bản thực hiện

7.4.4 Kiểm thử Hệ thống (System Test)

Đến giai đoạn này, công việc kiểm thử được tiến hành với nhìn nhận phần mềm như là một yếu tố trong một hệ thống thông tin phức tạp hoàn chỉnh.

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

7.4.5 Kiểm thử Chấp nhận (Acceptance Test)

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

7.4.6 Kiểm thử beta

Đây là giai đoạn mở rộng của alpha testing. Khi đó, việc kiểm thử được thực hiện bởi *một số lượng lớn* người sử dụng. Công việc kiểm thử được tiến hành một cách ngẫu nhiên mà không có sự hướng dẫn của các nhà phát triển. Các lỗi nếu được phát hiện sẽ được thông báo lại cho nhà phát triển.

7.4.7 Một số công việc khác

Ngoài những giai đoạn kiểm thử chủ chốt nêu trên, một số công việc khác trong quá trình kiểm thử bao gồm:

- *Báo cáo kết quả kiểm thử*: Sau khi hoàn tất kiểm thử, nhóm kiểm tra tạo ra các số liệu và báo cáo cuối cùng về nỗ lực kiểm thử của họ và cho biết phần mềm có sẵn sàng để phát hành hay không.
- *Phân tích kết quả kiểm thử hoặc phân tích thiếu sót*: được thực hiện bởi nhóm phát triển kết hợp với khách hàng để đưa ra quyết định xem những thiếu sót gì cần phải được chuyển giao, cố định và từ bỏ (tức là tìm ra được phần mềm hoạt động chính xác) hoặc giải quyết sau.
- *Kiểm tra lại khiếm khuyết*: Khi một khiếm khuyết đã được xử lý bởi đội ngũ phát triển, nó phải được kiểm tra lại bởi nhóm kiểm thử.
- *Kiểm thử hồi quy*: ta thường xây dựng chương trình kiểm thử nhỏ là tập hợp các bài kiểm tra cho mỗi tích hợp mới, sửa chữa hoặc cố định phần mềm, để đảm bảo rằng những cung cấp mới nhất đã không phá hủy bất cứ điều gì và toàn bộ phần mềm vẫn còn hoạt động một cách chính xác.
- *Kiểm thử đóng gói*: mỗi phép thử thỏa mãn các chỉ tiêu truy xuất và thu được những kết quả quan trọng như: bài học kinh nghiệm, kết quả, các bản ghi, tài liệu liên quan được lưu trữ và sử dụng như một tài liệu tham khảo cho các dự án trong tương lai.
- *Kiểm thử tự động hóa*: Nhiều nhóm lập trình càng ngày càng dựa vào phương pháp kiểm thử tự động, đặc biệt là các nhóm sử dụng mô hình TDD (*Test Drive Development*). Có rất nhiều khung phần mềm được viết bên trong và mã trong mỗi phiên bản được chạy kiểm thử tự động mọi lúc khi tích hợp liên tục phần

mềm. Tuy kiểm thử tự động không thể sao chép tất cả mọi thứ như con người có thể làm nhưng nó có thể rất hữu ích cho việc kiểm thử hồi quy. Tuy nhiên, nó cũng đòi hỏi phải có những kịch bản phát triển tốt để tiến hành kiểm thử.

- Kiểm thử chức năng và phi chức năng:

- *Kiểm thử chức năng* nhằm trả lời câu hỏi “người dùng có hay không làm được với tính năng cụ thể này” (xem Phương pháp Hộp đen)
- *Kiểm thử phi chức năng* đề cập đến các khía cạnh của phần mềm có thể không liên quan đến một chức năng cụ thể hoặc hành động người dùng, chẳng hạn như khả năng mở rộng và hiệu suất khác, hành vi dưới những hạn chế hoặc bảo mật nhất định. Việc kiểm thử sẽ xác định điểm cuộn mà tại đó khả năng mở rộng và thực hiện của các điểm cực trị hoạt động không ổn định. Những yêu cầu phi chức năng thường là những phản ánh về chất lượng của sản phẩm, đặc biệt là trong bối cảnh các quan điểm phù hợp của người sử dụng nó.
- *Kiểm thử sự phá hủy*: cố gắng làm hỏng phần mềm hoặc một hệ thống con. Nó xác minh rằng các phần mềm có chức năng đúng ngay cả khi nó nhận được đầu vào không hợp lệ hoặc không mong muốn, do đó tạo ra sự vững mạnh của xác nhận đầu vào và thói quen quản lý các lỗi. Chèn lỗi phần mềm ở dạng mờ nhạt là một ví dụ về kiểm thử thất bại. Các công cụ kiểm thử phi chức năng thường mại được liên kết từ các trang chèn lỗi phần mềm mà ở đó có sẵn vô số các mã nguồn mở và các công cụ miễn phí để thực hiện kiểm thử sự phá hủy phần mềm.
- *Kiểm thử hiệu suất phần mềm*: thường được chạy để xác định một hệ thống hay hệ thống con thực hiện như thế nào về độ nhạy và tính ổn định theo một khối lượng công việc cụ thể. Nó cũng có thể dùng để điều tra, đánh giá, xác nhận hoặc xác minh các thuộc tính chất lượng khác của hệ thống, chẳng hạn như khả năng mở rộng, độ tin cậy và sử dụng tài nguyên. Dạng kiểm thử này bao gồm một số dạng con như:
 - *Kiểm thử lượng tải* chủ yếu liên quan đến việc kiểm thử hệ thống có thể tiếp tục hoạt động dưới một lượng tải cụ thể, cho dù đó là một lượng lớn dữ liệu hoặc một số lượng lớn người sử dụng. Điều này thường được gọi là khả năng mở

rộng phần mềm. Các hoạt động kiểm thử lượng tải có liên quan khi thực hiện như một hoạt động phi chức năng thường được gọi là kiểm thử sức chịu đựng.

- *Kiểm tra khối lượng* là một cách để kiểm tra các chức năng của phần mềm ngay cả khi một số thành phần (ví dụ như một tập tin hoặc cơ sở dữ liệu) tăng triệt để kích thước. Kiểm thử độ căng là một cách để kiểm tra độ bền đột xuất hoặc ít gấp theo khối lượng công việc.
- *Kiểm thử tính ổn định* (thường được tham chiến lượng tải và kiểm thử độ bền) giúp kiểm tra xem phần mềm có thể hoạt động tốt liên tục trong hoặc trên một chu kỳ chấp nhận được. Có rất ít quy ước về các mục tiêu cụ thể của kiểm thử hiệu suất như là: Các thuật ngữ lượng tải, kiểm thử hiệu suất, kiểm thử tính mở rộng và kiểm thử khối lượng, thường được sử dụng thay thế cho nhau.
- Hệ thống phần mềm thời gian thực có những ràng buộc chính xác thời gian. Để kiểm thử những ràng buộc thời gian được đáp ứng thì người ta dùng phương pháp *kiểm thử thời gian thực*.
- *Kiểm thử tính khả dụng*: là rất cần thiết để kiểm tra xem giao diện có tiện dụng và dễ hiểu với người dùng không, nó liên quan trực tiếp đến năng lực sử dụng của phần mềm.
- *Kiểm thử khả năng tiếp cận*: chú ý việc tuân thủ các tiêu chuẩn sau:
 - Đạo luật bất khả thi của Mỹ năm 1990
 - Mục 508 sửa đổi của đạo luật Phục hồi năm 1973.
 - Sáng kiến tiếp cận Web (WAI) của W3C.
- *Kiểm thử bảo mật*: rất cần thiết trong việc xử lý dữ liệu mật và ngăn chặn các hacker xâm nhập hệ thống.

7.5 VÍ DỤ MINH HOẠ

Ví dụ: Xét phần mềm quản lý thư viện trong giai đoạn kiểm thử, trong đó các giai đoạn thực hiện trước đó đã được trình bày ở những bài trước.

❖ Giai đoạn 6: Kiểm chứng phần mềm hướng đối tượng

- Kiểm tra tính đúng đắn của các lớp đối tượng

- Chuẩn bị dữ liệu thử nghiệm: Nhập dữ liệu thử nghiệm cho các bảng THU_VIEN, SACH, DOC_GIA, MUON_SACH
- Kiểm tra:
 - Kiểm tra từng lớp đối tượng:
 - Kiểm tra lớp THU_VIEN (Tra cứu độc giả, Tra cứu sách)
 - Kiểm tra lớp DOC_GIA (Lập thẻ, cho mượn sách)
 - Kiểm tra lớp SACH (Nhận sách, Trả sách)
 - Kiểm tra phối hợp các lớp đối tượng
 - Kiểm tra phối hợp giữa lớp THU_VIEN và lớp DOC_GIA (Lập thẻ và sau đó tra cứu độc giả)
 - Kiểm tra phối hợp giữa lớp THU_VIEN và lớp SACH (Nhận sách và sau đó tra cứu sách)
 - Kiểm tra phối hợp giữa lớp DOC_GIA và lớp SACH (Lập thẻ, Nhận sách, Cho mượn sách và Tra sách)
 - Kiểm tra phối hợp giữa 3 lớp THU_VIEN, DOC_GIA và lớp SACH
- Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực

TÓM TẮT

Trong bài học này, chúng ta đã được giới thiệu các kiến thức về:

- *Mô hình chữ V*
- *Các yêu cầu đối với kiểm thử*
- *Các kỹ thuật kiểm thử*
 - *Phương pháp Hộp đen (Kiểm thử chức năng)*
 - *Phương pháp Hộp trắng (Kiểm thử cấu trúc)*
- *Chiến lược kiểm thử & các giai đoạn*
 - *Kiểm thử Đơn vị (Unit Test)*
 - *Kiểm thử Chức năng (Functional Test)*
 - *Kiểm thử Tích hợp (Integration Test)*
 - *Kiểm thử Hệ thống (System Test)*
 - *Kiểm thử Chấp nhận (Acceptance Test)*
 - *Kiểm thử beta*

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 8: TÁC VỤ BẢO TRÌ PHẦN MỀM

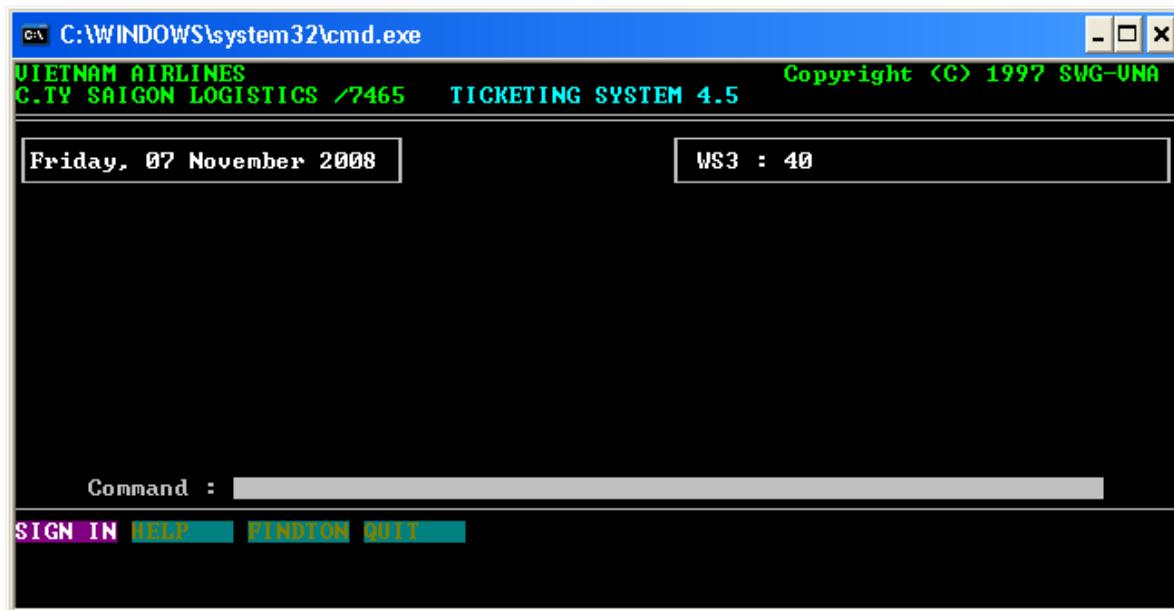
Học xong bài này người học sẽ:

- *Hiểu được tầm quan trọng của bảo trì phần mềm, kế hoạch bảo trì phần mềm, quy trình bảo trì phần mềm.*
- *Biết các xây dựng kế hoạch và quy trình bảo trì hệ thống phần mềm.*

8.1 GIỚI THIỆU

Bảo trì phần mềm⁵ là tác vụ sửa đổi một sản phẩm phần mềm sau khi xây dựng, nhằm sửa chữa lỗi hay thiếu sót để cải thiện hiệu suất hoặc các thuộc tính khác của phần mềm. Như hình minh họa sau đây, hệ thống bán vé máy bay này có thể hoạt động tốt trong giai đoạn 1990-2000 trên công nghệ cũ nhưng không thể vận hành được trên nền tảng công nghệ mới, dẫn đến vấn đề cấp thiết là phải được bảo trì hay nâng cấp.

⁵ http://en.wikipedia.org/wiki/Software_maintenance



Hình 8.1: Hệ thống bán vé máy bay hoạt động trên công nghệ cũ

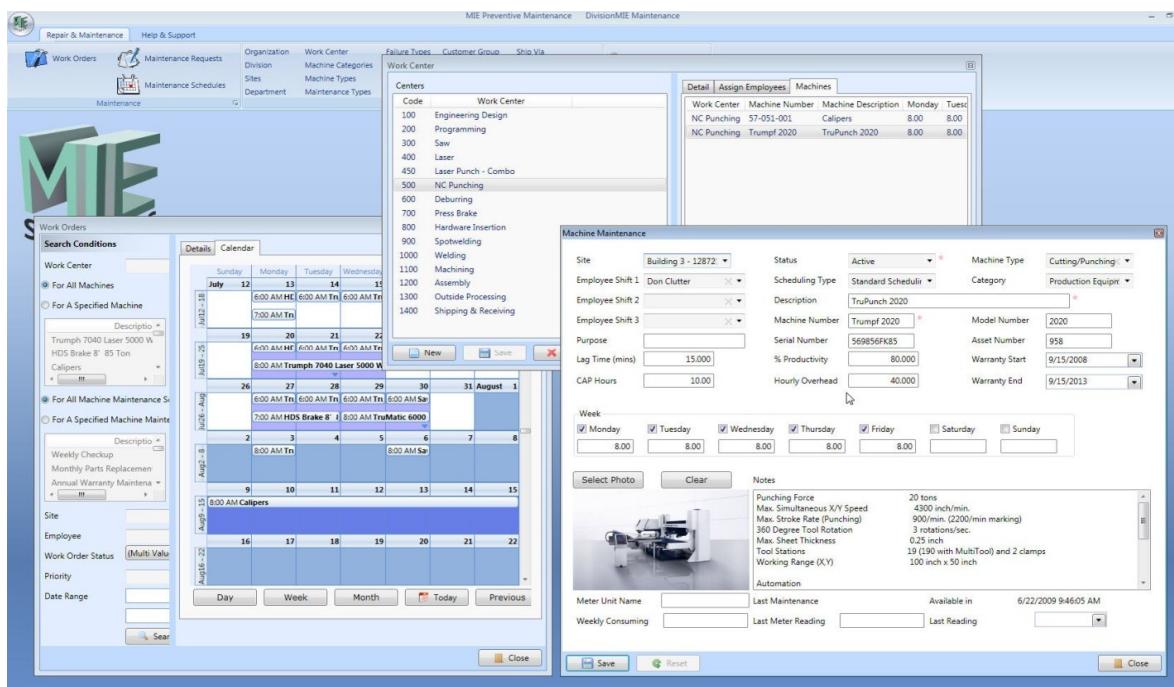
Trong thực tế, bảo trì phần mềm được nhận định là chỉ liên quan đến việc sửa chữa các khuyết điểm của phần mềm. Một nghiên cứu cho biết trên 80% công sức trong tác vụ này được dùng cho các hoạt động khác ngoài việc khắc phục lỗi, do người dùng thường gửi báo cáo về các vấn đề mà trong thực tế là cải tiến chức năng cho hệ thống mà không phải là sửa lỗi. Nhiều nghiên cứu gần đây đưa tỷ lệ lỗi cố định gần gũi hơn với 21%.

Việc bảo trì phần mềm và cải tiến hệ thống được nhóm nghiên cứu của Meir M. Lehman giải quyết lần đầu tiên năm 1969 đã dẫn đến việc xây dựng Luật Lehman vào năm 1997. Trong nghiên cứu đó, những phát hiện chính là (i) tác vụ bảo trì phần mềm thực sự là việc phát triển, (ii) các quyết định bảo trì được hỗ trợ bởi sự hiểu biết những gì sẽ xảy ra với hệ thống (và phần mềm) theo thời gian, đồng thời cho thấy hệ thống tiếp tục phát triển theo thời gian ngày càng phức tạp hơn (trừ khi một số hành động như mã sắp xếp được thực hiện để giảm sự phức tạp).

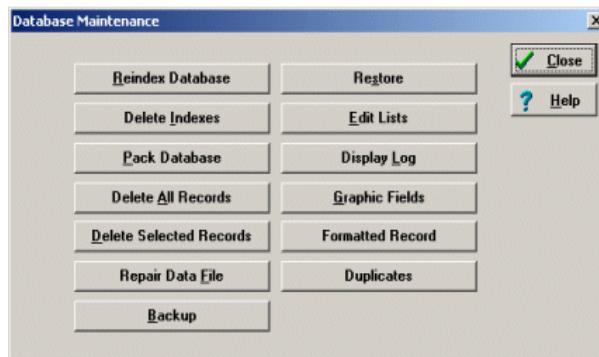
Trong bảo trì phần mềm, hai vấn đề quan trọng nhất là:

- **Vấn đề quản lý:** chính là liên kết với các ưu tiên của khách hàng, nhân viên, trong đó tổ chức thực hiện bảo trì, ước tính chi phí.
- **Vấn đề kỹ thuật:** chính là sự hiểu biết hạn chế, phân tích tác động, kiểm tra, đo lường bảo trì.

Bảo trì phần mềm là hoạt động rất rộng, từ việc sửa lỗi, cải tiến chức năng đến loại bỏ chức năng lỗi thời và tối ưu hóa. Thực tế cho thấy, sự thay đổi về chức năng hay yêu cầu của phần mềm là không thể tránh khỏi, nên cơ chế xử lý cần được phát triển để có thể đánh giá, kiểm soát việc sửa đổi. Vì thế, bất kỳ công việc nào thực hiện để thay đổi phần mềm sau đi vào hoạt động đều được xem là công việc bảo trì. Mục đích là để bảo tồn giá trị của phần mềm trên thời gian. Giá trị có thể được nâng cao bằng cách mở rộng nền tảng phần mềm, đáp ứng yêu cầu bổ sung, để việc sử dụng được dễ dàng và hiệu quả hơn. Bảo trì phần mềm có thể kéo dài trong 20 năm, trong khi phát triển phần mềm chỉ diễn ra trong 1-2 năm.



Hình 8.2: Minh họa hệ thống kiểm soát bảo trì (nguồn [5])



Hình 8.3: Minh họa ứng dụng cung cấp tác vụ bảo trì (nguồn [5])

8.2 TẦM QUAN TRỌNG CỦA BẢO TRÌ

Cuối những năm 1970, một nghiên cứu khảo sát nổi tiếng đã nhận định là phần lớn chi phí trong chu kỳ phát triển phần mềm đó đã được tiêu hao cho công tác bảo trì. Hoạt động bảo trì được chia thành bốn dạng:

- *Thích ứng (adaptive)*: nhằm sửa đổi hệ thống để đối phó với những thay đổi trong môi trường phần mềm (như dữ liệu, hệ điều hành).
- *Hoàn bị (perfective)*: nhằm thực hiện yêu cầu người dùng mới hoặc thay đổi này liên quan đến cải tiến chức năng phần mềm.
- *Khắc phục (corrective)*: nhằm tìm & sửa lỗi, tìm thấy bởi người dùng.
- *Phòng ngừa (preventive)*: nhằm tăng khả năng bảo trì phần mềm hoặc độ tin cậy để ngăn ngừa các vấn đề xấu trong tương lai.

Bảng sau đây mô tả tác động của các yếu tố điều chỉnh quan trọng về bảo trì phần mềm (được xếp theo thứ tự tác động tích cực tối đa):

Yếu tố Bảo trì	Mức ↑	Yếu tố Bảo trì	Mức ↑
1. Các chuyên gia bảo trì	35%	15. Đo lường chất lượng	16%
2. Kinh nghiệm nhân viên cao	34%	16. Kiểm tra mã cơ sở chính thức	15%
3. Biển bảng điều khiển và dữ liệu	33%	17. Thư viện Kiểm tra hồi quy	15%
4. Độ phức tạp thấp của mã cơ sở	32%	18. Thời gian đáp ứng tuyệt vời	12%
5. Y2K và công cụ tìm kiếm đặc biệt	30%	19. Đào tạo hàng năm của người dùng nhiều hơn 10 ngày	12%
6. Công cụ đang cơ cấu lại	29%	20. Kinh nghiệm quản lý cao	12%
7. Các công cụ tái kỹ thuật	27%	21. Hỗ trợ tự động (help-desk)	12%
8. Ngôn ngữ lập trình mức cao	25%	22. Không có mô-đun dễ bị lỗi	10%
9. Công cụ tái thiết kế	23%	23. Báo cáo lỗi trực tuyến	10%
10. Công cụ phân tích phức tạp	20%	24. Đo lường năng suất	8%
11. Các công cụ theo dõi lỗi	20%	25. Tuyệt vời dễ sử dụng	7%
12. Chuyên gia "cập nhật hàng loạt" Y2K	20%	26. Đo lường sự hài lòng của người sử dụng	5%
13. Công cụ kiểm soát sự thay đổi tự động	18%	27. Tinh thần đồng đội cao	5%
14. Làm thêm giờ chưa thanh toán	18%	Tổng hợp	503%

Các khảo sát cho thấy ~75% công sức trong bảo trì liên quan đến 2 loại đầu tiên (21% công sức để sửa lỗi). Ngoài ra, người dùng còn có đóng góp quan trọng trong quá trình thu thập, phân tích yêu cầu mới. Đó là nguyên nhân chính của vấn đề bất kỳ trong quá trình bảo trì cải tiến phần mềm.

Theo bảng trên, ta thấy các sự cố không chỉ do yếu tố #22 mà còn nhiều yếu tố khác, đều có thể làm giảm hiệu suất của phần mềm. Tình huống làm giảm hiệu suất rất phổ biến là thiếu các công cụ bảo trì phù hợp, như thiếu phần mềm theo dõi lỗi hay phần mềm quản lý thay đổi hoặc và phần mềm thư viện kiểm tra. Bảng sau đây mô tả một số yếu tố và phạm vi tác động bảo trì phần mềm (xếp theo thứ tự tác động tiêu cực tối đa)

Các yếu tố bảo trì	Mức ↓	Các yếu tố bảo trì	Mức ↓
1. mô-đun dễ bị lỗi	-50%	15. Không có thư viện kiểm tra hồi quy	-15%
2. Biến và dữ liệu nhúng	-45%	16. Không hỗ trợ tự động	-15%
3. Nhân viên thiếu kinh nghiệm	-40%	17. Không có báo cáo lỗi trực tuyến	-12%
4. Độ phức tạp cao của mã nguồn	-30%	18. Thiếu kinh nghiệm quản lý	-15%
5. Công cụ tìm kiếm đặc biệt không có Y2K	-28%	19. Không có công cụ tái cấu trúc mã nguồn	-10%
6. Phương pháp kiểm soát thay đổi thủ công	-27%	20. Không đào tạo hàng năm	10%
7. Ngôn ngữ lập trình cấp thấp	-25%	21. Không có công cụ tái thiết kế	-10%
8. Không có công cụ theo dõi lỗi	-24%	22. Không có công cụ phân tích độ phức tạp	-10%
9. Không có chuyên gia "cập nhật hàng loạt" Y2K	-22%	23. Không đo lường năng suất	-7%
10. Việc sử dụng nghèo nàn	-18%	24. tinh thần đồng đội nghèo nàn	-6%
11. Không đo lường chất lượng	-18%	25. Không đo lường sự hài lòng của người dùng	-4%
12. Không có các chuyên gia bảo trì	-18%	26. Không thanh toán tiền vượt giờ	0%
13. Thời gian đáp ứng kém	-16%	Tổng hợp	
14. Thiếu kiểm tra mã nguồn	-15%		

8.3 KẾ HOẠCH BẢO TRÌ PHẦN MỀM

Trong chu trình phát triển phần mềm, bảo trì là một tác vụ quan trọng cần được kế hoạch chính xác trong thời gian phần mềm được xây dựng. Kế hoạch đó cần xác định

được cách người dùng sẽ yêu cầu về sửa đổi hoặc báo cáo về các vấn đề trực tiếp, đồng thời tính toán được ngân sách dự trù cho việc bảo trì phải liên quan đến nguồn tài nguyên và chi phí có thể cần dùng. Việc bảo trì phần mềm (có thể kéo dài 5-6 năm hoặc lâu hơn, sau quá trình phát triển) đòi hỏi cần có một kế hoạch hiệu quả để có thể xác định phạm vi cần bảo trì phần mềm, điều chỉnh quá trình triển khai duy tu, hay chỉ định dịch vụ bảo trì, và ước tính chi phí cho cả quá trình.

8.4 QUY TRÌNH BẢO TRÌ PHẦN MỀM

Việc bảo trì phần mềm được bao gồm trong 6 quy trình như sau:

- *Quy trình hiện thực* cần có các tác vụ chuẩn bị phần mềm và chuyển giao, chuẩn bị để kiểm soát các sự cố được xác định trong giai đoạn phát triển phần mềm và theo dõi về quản lý cấu hình sản phẩm.
- *Quy trình phân tích* các sự cố hay thay đổi, là trách nhiệm của nhóm bảo trì, trong đó nhóm bảo trì cần phân tích từng yêu cầu, xác nhận nó (bằng cách tái tạo tình hình) & kiểm tra tính hợp lệ, điều tra và đề xuất một giải pháp, sau đó tài liệu hóa yêu cầu và giải pháp để xuất đó, cuối cùng cần giấy phép để triển khai giải pháp cho yêu cầu thay đổi.
- Quy trình xem xét việc thực hiện các sửa đổi nội tại.
- Quy trình chấp thuận các sửa đổi bởi việc xác nhận chúng với người đã yêu cầu để đảm bảo việc sửa đổi sẽ được thực hiện.
- Quy trình chuyển đổi (như nền tảng phần mềm) là trường hợp đặc biệt và không phải là công việc bảo trì hàng ngày. Nếu phần mềm cần được chuyển đổi sang nền tảng khác mà không thay đổi gì trong hệ thống chức năng, quy trình này sẽ được dùng và được nhóm bảo trì thực hiện.
- Cuối cùng là quy trình bảo dưỡng chỉ thực hiện trên một số nhóm phần mềm chứ không được thực hiện hàng ngày.

TÓM TẮT & ÔN TẬP

Bài học này cung cấp những thông tin tổng quát về:

- *Tầm quan trọng của bảo trì phần mềm*
- *Lập kế hoạch bảo trì phần mềm*
- *Quy trình bảo trì phần mềm*

Phần bài tập liên quan được trình bày trong Phụ lục A và B.

BÀI 9: QUẢN TRỊ DỰ ÁN PHẦN MỀM

Học xong bài này người học sẽ nắm được các nội dung sau:

- *Hiểu được các khái niệm cơ bản về quản trị dự án, hoạt động của quản trị dự án, độ đo phần mềm, các tác vụ cần thiết.*
- *Biết vận dụng các tác vụ trong quy trình quản lý dự án phần mềm.*

9.1 GIỚI THIỆU

9.1.1 Khái niệm dự án

Dự án là tập hợp các công việc được thực hiện bởi một tập thể (có thể có chuyên môn khác nhau, thực hiện công việc khác nhau, thời gian tham gia dự án khác nhau), nhằm đạt được một kết quả như dự kiến, trong thời gian dự kiến, với một kinh phí dự kiến.

Trong lĩnh vực Công nghệ phần mềm, công tác quản trị dự án phần mềm bao gồm các hoạt động (trong suốt quy trình thực hiện dự án) liên quan đến việc lập kế hoạch, giám sát và điều phối tài nguyên dự án (ví dụ như kinh phí, con người, thời gian thực hiện) hay xử lý các rủi ro, nhằm đảm bảo thành công cho dự án. Quản trị dự án phần mềm cần đảm bảo cân bằng giữa ba yếu tố: thời gian, tài nguyên và chất lượng. Ba yếu tố này được gọi là tam giác dự án

9.1.2 Các vấn đề thường xảy ra đối với một dự án phần mềm

- Thời gian thực hiện dự án vượt mức dự kiến
- Chi phí thực hiện dự án vượt mức dự kiến
- Kết quả của dự án không như dự kiến

9.2 TÓM LƯỢC VỀ QUẢN TRỊ DỰ ÁN

Quản trị dự án là tầng đầu tiên trong quy trình phát triển phần mềm vì nó là bước kỹ thuật cơ sở kéo dài suốt chu trình phát triển phần mềm.

Trách nhiệm của người quản trị dự án bao gồm:

- *Quản lý thời gian*: lập lịch, kiểm tra đối chiếu quá trình thực hiện dự án với lịch trình, điều chỉnh lịch trình khi cần thiết,
- *Quản lý tài nguyên*: xác định, phân bổ và điều phối tài nguyên
- *Quản lý sản phẩm*: thêm, bớt các chức năng phù hợp với yêu cầu của khách hàng
- *Quản lý rủi ro*: xác định, phân tích rủi ro, đề xuất cách khắc phục
- Tổ chức cách làm việc

Mục tiêu của việc quản trị dự án là đảm bảo cho dự án đáp ứng:

- Đúng thời hạn
- Không vượt dự toán
- Đầy đủ các chức năng đã định
- Đáp ứng yêu cầu của khách hàng (tạo ra sản phẩm tốt)

Quản lý dự án bao gồm các pha công việc sau:

- Thiết lập: viết đề án
- Ước lượng chi phí
- Phân tích rủi ro
- Lập kế hoạch
- Chọn người
- Theo dõi và kiểm soát dự án
- Viết báo cáo và trình diễn sản phẩm

Khi tiến hành quản lý dự án, người quản trị dự án có các nhiệm vụ và quyền hạn như sau:

- Về mặt thời gian:
 - Tạo lập kế hoạch, điều chỉnh kế hoạch
 - Kiểm tra/đổi chiếu các tiến trình con với kế hoạch
 - Giữ mức độ mềm dẻo nhất định trong kế hoạch
 - Phối thuộc các tiến trình con
- Về mặt tài nguyên: có thể thêm kinh phí, thiết bị, nhân lực ...
- Về mặt sản phẩm: thêm bớt chức năng của sản phẩm ...
- Về mặt rủi ro: phân tích/tìm cách xử lý, chấp nhận một số rủi ro

Ngoài ra, người quản trị dự án còn cần phải quan tâm đến sự phối thuộc với các dự án khác và thông tin cho người quản lý cấp trên ... Phương pháp tiếp cận của người quản trị dự án thường là:

- Hiểu rõ mục tiêu (tìm cách định lượng các mục tiêu ngay khi có thể),
- Hiểu rõ các ràng buộc (chi phí, lịch biểu, tính năng ...),
- Lập kế hoạch để đạt được mục tiêu trong các ràng buộc,
- Giám sát và điều chỉnh kế hoạch,
- Tạo môi trường làm việc ổn định, năng động cho nhóm.

Việc quản lý tồi sẽ dẫn đến sự chậm trễ của dự án, tính năng yếu kém và tăng chi phí phát triển. Một ví dụ kinh điển về quản lý tồi là dự án hệ điều hành OS360 của IBM bị chậm 2 năm so với kế hoạch.

9.3 HOẠT ĐỘNG CỦA QUẢN TRỊ DỰ ÁN

Các hoạt động chính trong quản trị dự án phần mềm gồm:

❖ Xác định dự án phần mềm cần thực hiện

- Xác định yêu cầu chung:
 - Trước tiên, ta cần xác định các yêu cầu chức năng (công cần thực hiện) cũng như phi chức năng (công nghệ dùng để phát triển, sử dụng trong hệ điều hành nào ...) của phần mềm.

- Sau đó ta xác định rõ các tài nguyên cần thiết để xây dựng phần mềm, liên quan đến nhân lực, các thành phần, phần mềm có thể sử dụng lại, các phần cứng hoặc công cụ có sẵn cần dùng đến; trong đó nhân tố con người là quan trọng nhất.
 - Điều cuối cùng là xác định thời gian cần thiết để thực hiện dự án. Trong quá trình này ta cần nắm bắt được bài toán thực tế cần giải quyết, cũng như các hoạt động mang tính nghiệp vụ của khách hàng, để có thể xác định rõ yêu cầu chung của đề án, xem xét dự án có khả thi hay không.
- Viết đề án:
- Đây là quá trình xây dựng tài liệu mô tả đề án để xác định phạm vi của dự án, trách nhiệm của những người tham gia dự án; là cam kết giữa người quản trị dự án, người tài trợ dự án và khách hàng. Nội dung của tài liệu mô tả đề án thường có những nội dung sau:
 - *Bối cảnh thực hiện dự án*: căn cứ pháp lý để thực hiện dự án, hiện trạng công nghệ thông tin của khách hàng trước khi có dự án, nhu cầu phần mềm phần mềm của khách hàng, đặc điểm và phạm vi của phần mềm sẽ xây dựng.
 - *Mục đích và mục tiêu của dự án*: xác định mục đích tổng thể như: Tin học hóa hoạt động nào trong quy trình nghiệp vụ của khách hàng? Xác định mục tiêu của phần mềm: lượng dữ liệu xử lý, lợi ích phần mềm đem lại.
 - *Phạm vi dự án*: những người liên quan tới dự án, các hoạt động nghiệp vụ cần tin học hóa.
 - *Nguồn nhân lực tham gia dự án*: chuyên viên nghiệp vụ, người phân tích, người thiết kế, người lập trình, người kiểm thử, người cài đặt triển khai dự án cho khách hàng, người hướng dẫn khách hàng sử dụng phần mềm, người bảo trì dự án phần mềm.
 - *Ràng buộc thời gian thực hiện dự án*: Ngày nghiệm thu dự án, ngày bàn giao dự án.
 - *Ràng buộc kinh phí*: kinh phí trong từng giai đoạn thực hiện dự án.

- *Ràng buộc công nghệ phát triển:* công nghệ nào được phép sử dụng để thực hiện dự án.
- Chữ kí các bên liên quan tới dự án.

❖ Lập kế hoạch thực hiện dự án

Lập kế hoạch thực hiện dự án là hoạt động diễn ra trong suốt quá trình từ khi bắt đầu thực hiện dự án đến khi bàn giao sản phẩm với nhiều loại kế hoạch khác nhau nhằm hỗ trợ kế hoạch chính của dự án phần mềm về lịch trình và ngân sách.

Các loại kế hoạch thực hiện dự án gồm có:

- *Kế hoạch đảm bảo chất lượng:* mô tả chuẩn, quy trình được sử dụng.
- *Kế hoạch thẩm định:* mô tả phương pháp, nguồn lực, lịch trình thẩm định hệ thống.
- *Kế hoạch quản lý cấu hình:* mô tả thủ tục, cấu trúc quản lý cấu hình được sử dụng.
- *Kế hoạch bảo trì:* dự tính yêu cầu về hệ thống, chi phí, nỗ lực cần thiết cho bảo trì.
- *Kế hoạch phát triển đội ngũ:* mô tả kĩ năng và kinh nghiệm của các thành viên trong nhóm dự án sẽ phát triển như thế nào.

Quy trình lập kế hoạch thực hiện dự án bao gồm:

- *Thiết lập các ràng buộc của dự án:* thời gian, nhân lực, ngân sách,
- Đánh giá bước đầu về các tham số của dự án: quy mô, độ phức tạp, nguồn lực,
- Xác định các mốc thời gian trong thực hiện dự án và sản phẩm thu được ứng với mỗi mốc thời gian,
- Trong khi dự án chưa hoàn thành hoặc chưa bị hủy bỏ *thì thực hiện lặp đi lặp lại các công việc sau:*
 - Lập lịch thực hiện dự án
 - Thực hiện các hoạt động theo lịch trình
 - Theo dõi sự tiến triển của dự án, so sánh với lịch trình
 - Đánh giá lại các tham số của dự án
 - Lập lại lịch thực hiện dự án cho các tham số mới

- Thỏa thuận ràng buộc và sản phẩm bàn giao của mỗi mốc thời gian
- Nếu có vấn đề nảy sinh thì xem xét lại các kĩ thuật khởi đầu đưa ra các biện pháp cần thiết

Cấu trúc kế hoạch thực hiện dự án bao gồm:

- Tổ chức dự án
- Phân tích các rủi ro
- Yêu cầu về tài nguyên phần cứng, phần mềm
- Phân công công việc
- Lập lịch dự án
- Cơ chế kiểm soát và báo cáo
- ❖ **Tổ chức thực hiện dự án**
- ❖ **Quản lý quá trình thực hiện dự án**
- ❖ **Kết thúc dự án**

9.4 ĐỘ ĐO PHẦN MỀM

Để quản lý dự án, ta cần định lượng được đối tượng cần quản lý là phần mềm và quy trình phát triển. Ta cần đo kích cỡ phần mềm, chất lượng phần mềm, năng suất phần mềm ...

9.4.1 Đo lường kích cỡ phần mềm

Có hai phương pháp phổ biến để đo kích cỡ phần mềm là đo số dòng lệnh (LOC – Lines Of Code) và đo điểm chức năng (FP – Function Points).

Phương pháp Độ đo LOC tương đối trực quan, nhưng phụ thuộc rất nhiều vào ngôn ngữ lập trình cụ thể. Từ kích cỡ của phần mềm (tính theo đơn vị LOC), ta tính được một số giá trị như sau:

- Hiệu năng = KLOC/người/tháng
- Chất lượng = số lỗi/KLOC

- Chi phí = giá thành/KLOC

Các thông số của các dự án đã phát triển trong quá khứ sẽ được dùng để phục vụ cho ước lượng cho các phần mềm sẽ phát triển.

Phương pháp Điểm chức năng (FP) được tính dựa trên đặc tả yêu cầu và độc lập với ngôn ngữ phát triển. Tuy nhiên nó lại có sự phụ thuộc vào các tham số được thiết lập dựa trên kinh nghiệm. Mô hình cơ sở của tính điểm chức năng là: $FP = a_1I + a_2O + a_3E + a_4L + a_5F$

Trong đó: I: số Input, O: số Output, E: số yêu cầu, L: số tập tin truy cập, F: số giao diện ngoại lai (devices, systems)

9.4.2 Độ đo dựa trên thống kê

Một số độ đo phần mềm khác dựa trên thống kê như sau:

- *Độ tin cậy* MTBF (Mean Time Between Failure): thời gian chạy liên tục của hệ thống
- *Thời gian khôi phục* MTTR (Mean Time To Repair)
- *Tính sẵn có* được tính bằng $MTBF/(MTBF + MTTR)$

9.5 CÁC TÁC VỤ CÂN THIẾT

9.5.1 Ước lượng

Việc đầu tiên của người quản trị dự án là ước lượng về kích cỡ, chi phí, thời gian tiến hành dự án. Việc này thông thường được tiến hành bằng cách phân rã phần mềm cần phát triển thành các khối nhỏ và áp dụng các kinh nghiệm (các thông số như kích cỡ, chi phí, năng lực nhân viên ...) đối với các phần mềm đã phát triển để ước lượng, đánh giá công việc.

Một mô hình ước lượng hay được dùng là mô hình COCOMO (Constructive Cost Model) ước lượng chi phí từ số dòng lệnh. Dùng mô hình này ta sẽ có thể ước lượng các thông số sau:

- Nỗ lực phát triển $E = a * L^b$

- Thời gian phát triển $T = c * E^d$
- Số người tham gia $N = E/T$

Trong đó a, b, c, d là các tham số tùy thuộc vào từng loại dự án (xem bảng sau). Điểm đáng chú ý ở đây là từ nỗ lực phát triển ta suy ra thời gian và số người tham gia vào dự án.

Hình 9.1: COCOMO - Các tham số cơ sở

	a	b	c	d
<i>Organic (đơn giản)</i>	3.2	1.05	2.5	0.38
<i>Semi-detached (trung bình)</i>	3.0	1.12	2.5	0.35
<i>Embeded (phức tạp)</i>	2.8	1.2	2.5	0.32

Các bước tiến hành của COCOMO như sau:

- Thiết lập kiểu dự án (organic, semi-detached, embeded)
- Xác lập các mô-đun và ước lượng dòng lệnh
- Tính lại số dòng lệnh trên cơ sở tái sử dụng
- Tính nỗ lực phát triển E cho từng mô-đun
- Tính lại E dựa trên độ khó của dự án (mức độ tin cậy, kích cỡ cơ sở dữ liệu, yêu cầu về tốc độ, bộ nhớ ...)
- Tính thời gian và số người tham gia

Ví dụ: Phần mềm với 33.3 ngàn dòng lệnh, và tham số a, b, c, d lần lượt là 3.0, 1.12, 2.5, 0.35, ta tính được:

- $E = 3.0 * 33.3^{1.12} = 152$ người/tháng
- $T = 2.5 * E^{0.35} = 14.5$ tháng
- $N = E / D \sim 11$ người

Ta cần nhớ rằng đo phần mềm là công việc rất khó khăn do

- Hầu hết các thông số đều không đo được một cách trực quan
- Rất khó thẩm định được các thông số
- Không có mô hình tổng quát

- Các kỹ thuật đo còn đang thay đổi

Ta không thể kiểm soát được quá trình sản xuất phần mềm nếu không ước lượng (đo) nó. Một mô hình ước lượng nghèo nàn vẫn hơn là không có mô hình nào và phải liên tục ước lượng lại khi dự án triển triển.

9.5.2 Quản lý nhân sự

Chi phí (trả công) con người là phần chính của chi phí xây dựng phần mềm. Ngoài ra, năng lực của người phát triển phần mềm lại rất biến thiên, kéo theo sự phức tạp trong tính toán chi phí. Phát triển phần mềm được tiến hành theo nhóm. Kích thước tốt của nhóm là từ 3 đến 8 người. Phần mềm lớn thường được xây dựng bởi nhiều nhóm nhỏ. Một nhóm phát triển có thể gồm các loại thành viên sau: (i) người phát triển, (ii) chuyên gia về miền ứng dụng, (iii) người thiết kế giao diện, (iv) Thủ thư phần mềm (quản lý cấu hình phần mềm), (v) Người kiểm thử.

Một nhóm phát triển cần có người quản trị, và người có vai trò lãnh đạo về mặt kỹ thuật. Một đặc trưng của phương pháp làm việc theo nhóm là sự trao đổi thông tin (giao tiếp) giữa các thành viên trong nhóm. Thời gian dùng cho việc giao tiếp có thể chiếm đến nửa tổng thời gian dành cho pháp triển phần mềm.

Ngoài ra, thời gian không dùng cho phát triển sản phẩm cũng chiếm phần lớn thời gian còn lại của người lập trình. Một người có thể đồng thời làm việc cho nhiều nhóm (dự án) phần mềm khác nhau. Điều này làm cho việc tính toán giá thành phần mềm phức tạp. Ta cần ghi nhớ, trong sản xuất phần mềm thì:

- Năng lực của các thành viên là không đồng đều
- Người tốt (nhất) có thể sản xuất hơn 5 lần trung bình, người kém có thể không cho kết quả gì
- Một số công việc quá khó đối với mọi người

Ta không nên tăng số thành viên một cách tùy tiện, vì như thế ta chỉ làm tăng sự phức tạp giao tiếp giữa các thành viên, khiến công việc nhiều khi chậm lại. Ta cũng cần lưu ý một số việc phức tạp, đặc thù chỉ nên để một người làm.

9.5.3 Quản lý cấu hình

Quản lý cấu hình phần mềm là một công việc quan trọng trong sản xuất phần mềm. Mã nguồn (và dữ liệu) là sản phẩm chính của dự án phần mềm. Quản lý cấu hình được tự động hóa thông qua các công cụ. Nhiệm vụ chính của công cụ quản lý là:

- Lưu trữ mã nguồn
- Tạo ra một điểm truy cập duy nhất (phiên bản thống nhất) cho người lập trình sửa đổi, thêm bớt mã nguồn.

Do đó ta có thể dễ dàng:

- Kiểm soát được tính thống nhất của mã nguồn,
- Kiểm soát được sửa đổi, lý do của sự sửa đổi, lịch các lần sửa đổi,
- Dễ dàng lưu trữ, truy cập tới các bản khác nhau của phần mềm,
- Tối ưu hóa vùng đĩa cần thiết cho lưu trữ,

Phương thức hoạt động của các công cụ quản lý cấu hình là:

- Quản lý tập chung (mã nguồn, tư liệu, công cụ phát triển ...)
- Các tập tin được tạo một lần duy nhất, các phiên bản sửa đổi chỉ ghi lại sai phân đối với bản gốc
- Sử dụng cách kiểm xuất/nhập (check out/in) khi sửa đổi tập tin

Thông thường, người phát triển khi muốn sửa đổi mã nguồn sẽ thực hiện thao tác check out tập tin đó. Khi tập tin đã bị check out thì các người phát triển khác chỉ có thể mở tập tin dưới dạng chỉ đọc. Khi kết thúc sửa đổi và ghi tập tin vào cơ sở dữ liệu, người sửa đổi tiến hành check in để thông báo kết thúc công việc sửa đổi, đồng thời có thể ghi lại các thông tin liên quan (lý do sửa đổi ...) đến sự sửa đổi.

Dữ liệu được lưu trữ của dự án thông thường bao gồm:

- Mã nguồn, Dữ liệu, Tư liệu
- Công cụ phát triển (chương trình dịch ...), thường cần để đảm bảo tương thích với các phiên bản cũ, và để đảm bảo chương trình được tạo lại (khi sửa lỗi ...) đúng như cái đã phân phát cho khách hàng

- Các ca kiểm thử

Một số các công cụ quản lý cấu hình phổ biến là RCS, CVS trên HĐH Solaris và SourceSafe của Microsoft.

9.5.4 Quản lý rủi ro

Quản lý rủi ro là một công việc đặc biệt quan trọng và khó khăn trong phát triển phần mềm. Có các nguyên nhân (rủi ro) sau dẫn đến chấm dứt dự án:

- Chi phí phát triển quá cao,
- Quá chậm so với lịch biểu,
- Tính năng quá kém so với yêu cầu.

Quản lý rủi ro bao gồm các công việc chính sau:

- Dự đoán rủi ro,
- Đánh giá khả năng xảy ra và thiệt hại,
- Tìm giải pháp khắc phục.

Dưới đây là các rủi ro thường xảy ra khi phát triển phần mềm và các phương pháp khắc phục chúng:

- *Thiếu người phát triển*: sử dụng những người tốt nhất; xây dựng nhóm làm việc; đào tạo người mới,
- *Kế hoạch, dự toán không sát thực tế*: ước lượng bằng các phương pháp khác nhau; lọc, loại bỏ các yêu cầu không quan trọng,
- *Phát triển sai chức năng*: chọn phương pháp phân tích tốt hơn; phân tích tính tổ chức/mô hình nghiệp vụ của khách hàng,
- *Phát triển sai giao diện*: phân tích thao tác người dùng; tạo kịch bản cách dùng; tạo bản mẫu,
- *Yêu cầu quá cao*: lọc bớt yêu cầu; phân tích chi phí/lợi ích,
- *Thay đổi yêu cầu liên tục*: áp dụng thiết kế che dấu thông tin; phát triển theo mô hình tiến hóa.

TÓM TẮT

Bài học này trình bày tóm lược về công tác quản trị dự án phần mềm thông qua các vấn đề sau đây:

- Khái niệm dự án
- Các vấn đề thường xảy ra đối với một dự án phần mềm
- Hoạt động của quản trị dự án
- Độ đo phần mềm (đo lường kích cỡ phần mềm, độ đo dựa trên thống kê)
- Các tác vụ cần thiết
 - Ước lượng
 - Quản lý nhân sự
 - Quản lý cấu hình
 - Quản lý rủi ro

BÀI 10: QUY TRÌNH PHÁT TRIỂN PHẦN MỀM

Học xong bài này người học sẽ nắm được các nội dung sau:

- *Hiểu được các khái niệm cơ bản về quy trình, quy trình ISO, CMM/CMMI.*
- *Biết được ưu nhược điểm của từng dạng quy trình, để từ đó có thể áp dụng phù hợp hơn trong thực tế.*

10.1 GIỚI THIỆU

Cũng như mọi ngành sản xuất khác, quy trình⁶ là một trong những yếu tố cực kỳ quan trọng đem lại sự thành công cho đơn vị nhà sản xuất phần mềm, nó giúp cho mọi thành viên trong dự án từ người cũ đến người mới, trong hay ngoài công ty đều có thể xử lý đồng bộ công việc tương ứng vị trí của mình thông qua cách thức chung của công ty, ít nhất ở cấp độ dự án.

Quy trình phát triển / xây dựng phần mềm (*Software Development / Engineering Process - SEP*) có tính chất quyết định để tạo ra sản phẩm chất lượng có tốt với chi phí thấp và năng suất cao, có ý nghĩa quan trọng đối với các công ty sản xuất hay gia công phần mềm, từ đó giúp củng cố và phát triển nền công nghiệp phần mềm đầy cạnh tranh.

⁶ http://en.wikipedia.org/wiki/Software_maintenance

10.2 GIỚI THIỆU VỀ QUY TRÌNH

Quy trình có thể hiểu là phương pháp thực hiện hoặc sản xuất ra sản phẩm. Tương tự như vậy, SEP chính là phương pháp phát triển hay sản xuất ra sản phẩm phần mềm.

Thông thường một quy trình bao gồm những yếu tố cơ bản sau:

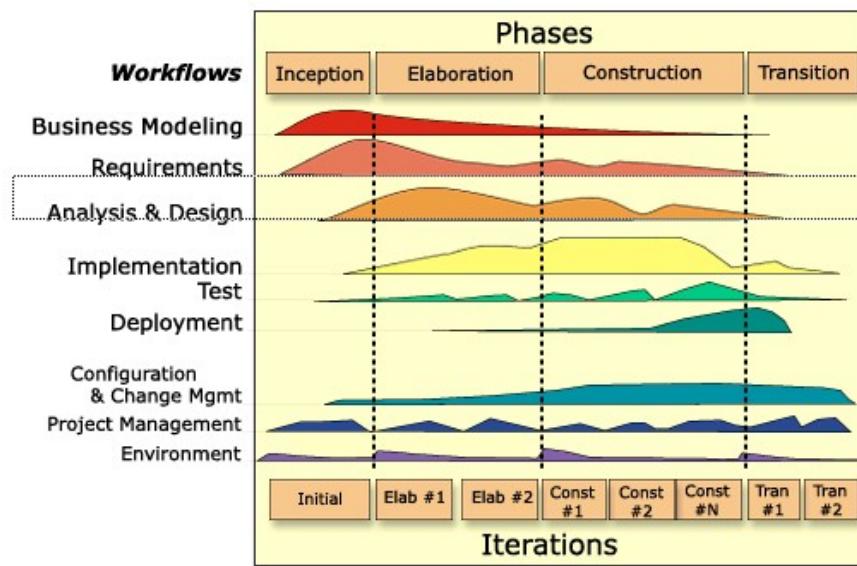
- Thủ tục (*Procedures*)
- Hướng dẫn công việc (*Activity Guidelines*)
- Biểu mẫu (*Forms/templates*)
- Danh sách kiểm định (*Checklists*)
- Công cụ hỗ trợ (*Tools*)

Quy trình gồm các nhóm công việc chính:

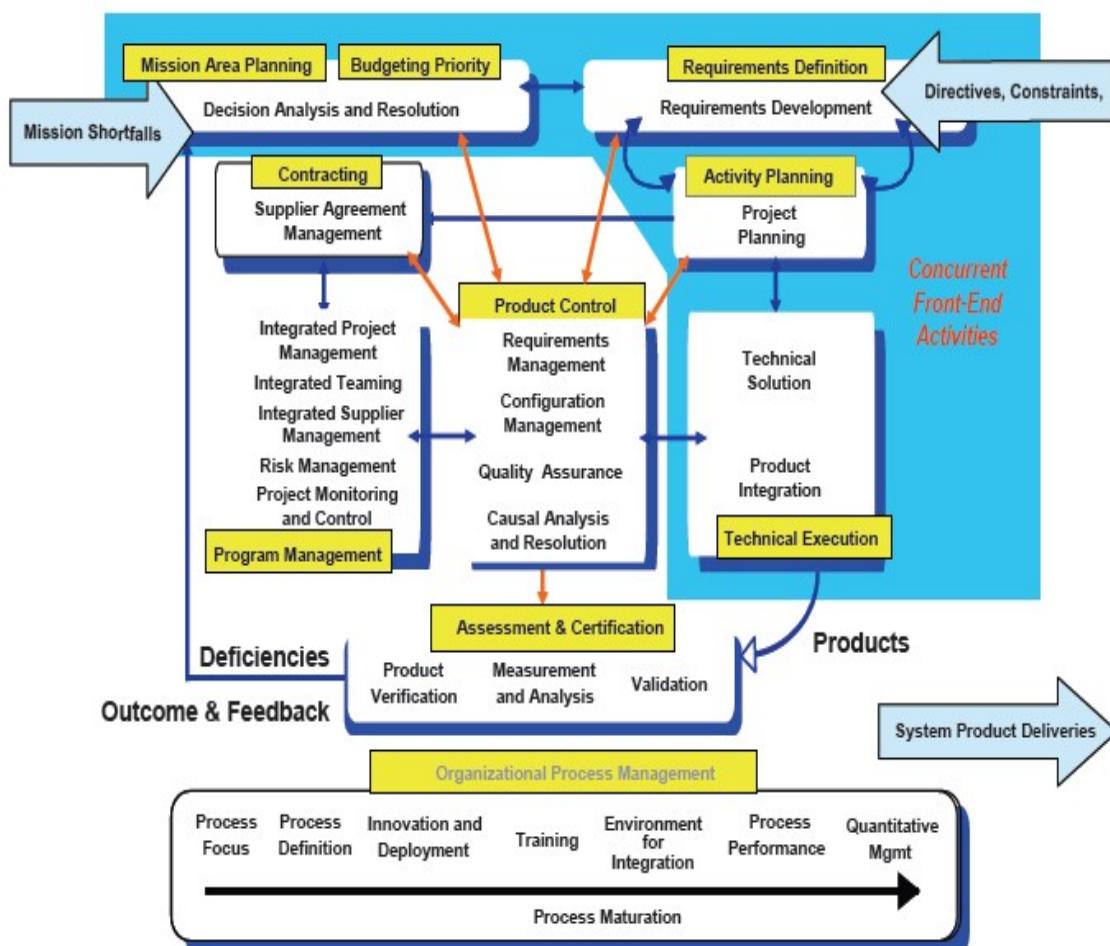
- *Đặc tả yêu cầu* (Requirements Specification): chỉ ra những “đòi hỏi” cho cả các yêu cầu chức năng và phi chức năng.
- *Phát triển phần mềm* (Development): tạo ra phần mềm thỏa mãn các yêu cầu được chỉ ra trong “Đặc tả yêu cầu”.
- *Kiểm thử phần mềm* (Validation/Testing): để bảo đảm phần mềm sản xuất đáp ứng những “đòi hỏi” trong “Đặc tả yêu cầu”.
- *Thay đổi phần mềm* (Evolution): đáp ứng các yêu cầu thay đổi của khách hàng.

Tùy theo mô hình phát triển phần mềm, các nhóm công việc được triển khai theo những cách khác nhau. Để sản xuất cùng một sản phẩm phần mềm người ta có thể dùng các mô hình khác nhau. Tuy nhiên không phải tất cả các mô hình đều thích hợp cho mọi phần mềm.

Một số hình ảnh minh họa về quy trình:



Hình 10.1: Quy trình Rational Unified Process



Hình 10.2: Các quy trình công nghệ liên quan

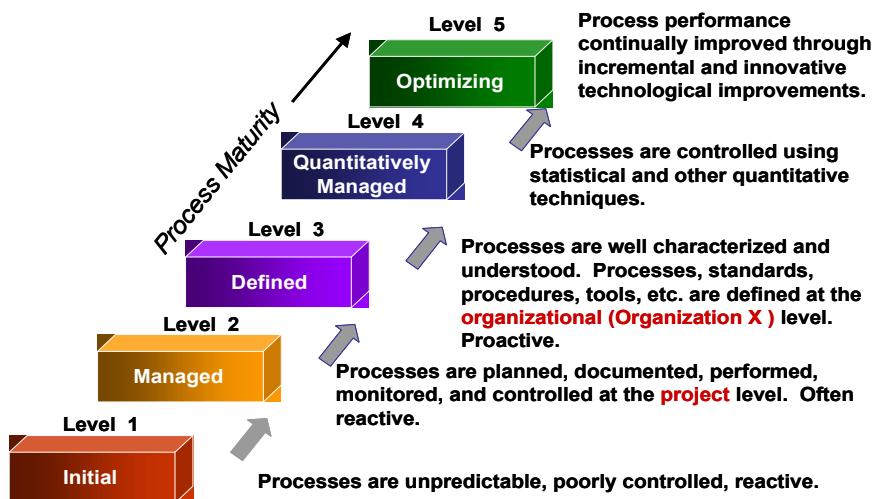
10.3 QUY TRÌNH ISO, CMM/CMMI

Phần này sẽ không đi sâu vào tìm hiểu các mô hình phát triển phần mềm mà chỉ cung cấp một cái nhìn tổng quát về chúng, cũng như mối quan hệ giữa SEP với ISO và CMM/CMMI.

Vấn đề được đặt ra là làm thế nào cải tiến quy trình để cải thiện chất lượng và năng suất? Câu trả lời chính là khung quy trình (*Process Framework - PF*). PF sẽ chỉ ra những yêu cầu mà một quy trình phải đáp ứng tùy theo mỗi mức độ. PF không chỉ ra bất kỳ một quy trình cụ thể nào mà chỉ đưa ra những yêu cầu ở mỗi mức độ trưởng thành khác nhau của quy trình phải đạt được. Đây chính là những hướng dẫn cho các hoạt động cải tiến để nâng mức độ trưởng thành từ thấp lên cao.

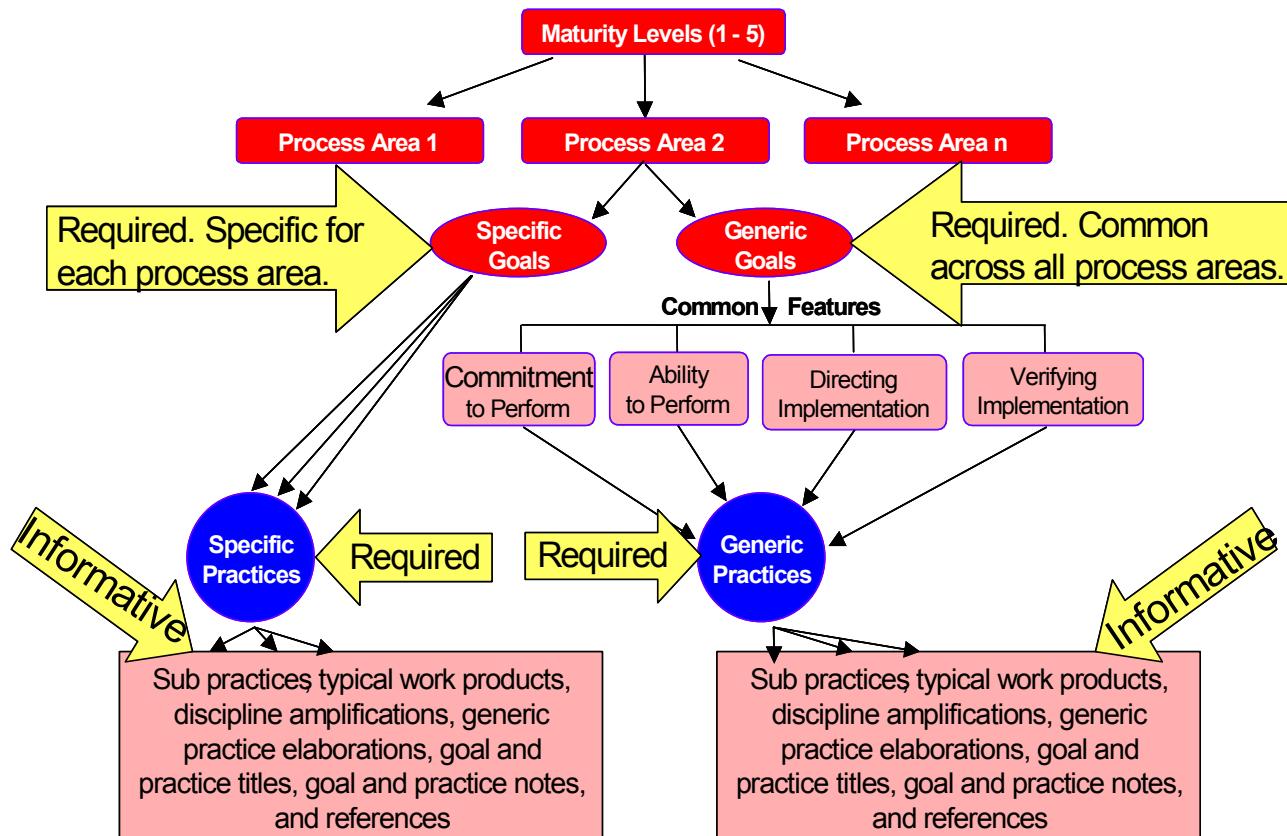
Trên thế giới hiện có nhiều PF, nhưng phổ biến nhất là ISO (International Organization for Standardization, <http://www.iso.org>) và CMM (Capability Maturity Model, <http://www.sei.cmu.edu/cmm>) được các tổ chức thế giới công nhận. ISO nhằm chung đến nhiều loại tổ chức cả sản xuất lẫn dịch vụ, trong khi CMM được dành riêng cho các tổ chức phát triển phần mềm.

Đối với phần mềm, ISO chỉ ra mức độ chất lượng yêu cầu tối thiểu mà một SEP phải đạt (ISO certified) và việc cải tiến quy trình được thực hiện thông qua quy trình kiểm định, trong khi CMM bao gồm những thực tiễn tốt nhất (*best practices*) được tập hợp từ rất nhiều tổ chức phát triển phần mềm khác nhau và chúng được tổ chức thành 5 mức độ như sau.



Hình 10.3: Các mức độ trưởng thành của CMM

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



Hình 10.4: Các thành phần trong hệ thống chuẩn CMM/CMMI

Mô hình SEP còn được gọi là chu trình hay vòng đời phần mềm (*SLC - Software Life Cycle*). SLC là tập hợp các công việc và quan hệ giữa chúng diễn ra trong quá trình phát triển phần mềm. Có khá nhiều mô hình SLC khác nhau, trong đó một số được ứng dụng khá phổ biến trên thế giới như:

Các mô hình một phiên bản (*Single-version models*)

- Mô hình Waterfall (Waterfall model)
- Mô hình chữ V (V-model)

Các mô hình nhiều phiên bản (*Multi-version models*)

- Mô hình mẫu (Prototype)
- Mô hình tiến hóa (Evolutionary)

- Mô hình lặp và tăng dần (Iterative and Incremental)
- Mô hình phát triển ứng dụng nhanh (RAD)
- Mô hình xoắn (Spiral)

TÓM TẮT

Bài học cuối cùng này giới thiệu tóm lược đến người học những vấn đề chuyên sâu trong lĩnh vực Công nghệ Phần mềm, đó là:

- *Giới thiệu về khung quy trình sản xuất phần mềm*
- *Quy trình ISO, CMM/CMMI.*

PHỤ LỤC A – BÀI TẬP

❖ CÂU HỎI LÝ THUYẾT

1. Nêu sự khác biệt của giai đoạn thiết kế trong các quy trình khác nhau
2. Nêu sự khác biệt của giai đoạn lập trình trong các quy trình khác nhau
3. Khi tiến hành thực hiện phần mềm qua các giai đoạn (trong quy trình 5 giai đoạn) có thể phát sinh lỗi trong một giai đoạn nào đó (kết quả chuyển giao không chính xác, thiếu sót, v.v...). Nêu lỗi (nếu phát sinh) của giai đoạn nào là nghiêm trọng nhất.
4. Trong các giai đoạn của quy trình công nghệ phần mềm, nêu giai đoạn: (i) quan trọng nhất, dễ thực hiện nhất, tốn nhiều thời gian và chi phí nhất, có thể bỏ qua (trong trường hợp nào). Giải thích tại sao.
5. Nêu sự khác biệt cơ bản giữa yêu cầu chức năng (yêu cầu nghiệp vụ, yêu cầu hệ thống) và phi chức năng (yêu cầu chất lượng). Loại yêu cầu nào là quan trọng hơn.
6. Xác định các yêu cầu chức năng hệ thống có thể có trong các phần mềm sau (chi tiết về quy định, biểu mẫu liên quan có trong mô tả của đề tài)
 - a. Phần mềm quản lý bán sách
 - b. Phần mềm quản lý học sinh trường cấp 3
 - c. Phần mềm đánh cờ gánh
 - d. Phần mềm hỗ trợ giải bài tập phương trình đại số
 - e. Phần mềm quản lý giải vô địch bóng đá quốc gia
7. Mỗi phát biểu sau là đúng hay sai? Nếu đúng, giải thích. Nếu sai, giải thích và ví dụ minh họa.
 - a. Mọi phần mềm đều có yêu cầu về tính tiện dụng
 - b. Mọi phần mềm đều có yêu cầu về tính hiệu quả
 - c. Mọi phần mềm đều có yêu cầu chức năng hệ thống

- d. Việc mô hình hóa yêu cầu không cung cấp thêm thông tin mới về yêu cầu của phần mềm mà chỉ giúp trình bày lại yêu cầu của phần mềm dưới dạng trực quan hơn
- e. Cho biết các kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước thiết kế giao diện của đối tượng hay không
- f. Kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định thuộc tính đối tượng (giai đoạn thiết kế)
- g. kết quả của việc mô hình hóa yêu cầu có được sử dụng trong bước xác định hàm xử lý của đối tượng (giai đoạn thiết kế)
8. Nếu không thực hiện qua bước mô hình hóa yêu cầu thì việc lập mô hình đối tượng sẽ có các khó khăn gì? Tại sao?

❖ **BÀI TẬP ĐÁNH GIÁ**

Hãy thực hiện đánh giá giao diện của:

- <http://www.google.com> cho màn hình kết quả sau khi tìm kiếm xong.
- <http://www.costco.com> cho màn hình giới thiệu và màn hình sản phẩm.
- Yahoo mail và Google mail cho màn hình chính sau khi đăng nhập
- Một số website của Việt nam (VN Express, ...)

❖ **BÀI TẬP PHÂN TÍCH**

Bài tập 1: Quản lý thuê bao điện thoại

Lưu trữ: Các thông tin về

- Hợp đồng thuê điện thoại (Khách hàng, loại thuê bao, máy điện thoại)
- Cuộc gọi (Máy điện thoại, Ngày, Giờ, Thời gian, Nơi gọi đến).

Tính toán:

- Số tiền phải trả của từng máy điện thoại trong từng tháng:
 - Tiền thuê bao hàng tháng (phụ thuộc vào từng loại thuê bao với các định mức riêng).
 - Tiền cước phí trả thêm (phụ thuộc vào thời gian gọi, số phút gọi, nơi gọi đến)

- Tính công nợ khách hàng đối với khách hàng chưa thanh toán tiền.

Kết xuất:

- Hóa đơn tính tiền điện thoại cho từng khách hàng trong từng tháng.
- Danh sách khách hàng chưa thanh toán tiền điện thoại.
- Thống kê về nơi gọi đến, thời điểm gọi theo khu vực trong từng tháng.

Bài tập 2: Quản lý học sinh trường phổ thông trung học

Lưu trữ: Các thông tin về

- Học sinh: Họ, tên, lớp, ngày sinh, giới tính, địa chỉ, thành phần, kết quả học tập, điểm danh.

Tra cứu: Thông tin về học sinh

Tính toán:

- Điểm trung bình từng môn học theo từng học kỳ: Tính theo điểm của từng hình thức kiểm tra (15 phút: HS1, 1 tiết: HS2, thi học kỳ: HS3)
- Điểm trung bình HK1, HK2, cả năm (học kỳ 1: HS1, học kỳ 2: HS2) .
- Xếp loại: Xuất sắc nếu điểm trung bình niên khóa ≥ 9.0 và không có môn nào có điểm trung bình dưới 7.5. Tiên tiến nếu điểm trung bình niên khóa ≥ 7.5 và không có môn nào có điểm trung bình dưới 6.0. Đạt yêu cầu nếu điểm trung bình niên khóa ≥ 5 và không có môn nào có điểm trung bình dưới 5. Không đạt yêu cầu nếu có ít nhất 1 môn dưới 5.

Ghi chú: Nếu tổng số ngày vắng vượt quá 20 \rightarrow loại không đạt yêu cầu. Nếu số ngày vắng vượt quá 10 hay số ngày vắng không phép vượt quá 5 thì sẽ bị hạ xuống một bậc (chỉ áp dụng với loại xuất sắc và tiên tiến).

Kết xuất:

- Danh sách học sinh theo từng lớp.
- Phiếu điểm cho mỗi học sinh.
- Bảng điểm các môn và bảng điểm tổng kết cho từng lớp.
- Thống kê về xếp loại học sinh của toàn trường trong 1 niên khóa.

Bài tập 3: Quản lý các tài khoản trong ngân hàng**Lưu trữ:**

- Tài khoản: Khách hàng, loại tài khoản, số tiền, loại tiền, ngày gởi, tình trạng
- Quá trình gửi và rút tài khoản: Khách hàng, ngày số tiền, hình thức.
- Các quy định về lãi suất và tỷ giá.

Tra cứu: Tài khoản theo các tiêu chuẩn

- Mã số, Khách hàng, Loại tài khoản, Ngày mở, ngày đóng.

Tính toán:

- Lãi suất cho từng tài khoản khi đến kỳ hạn hay khi khách hàng rút trước kỳ hạn (chỉ được không kỳ hạn).

Kết xuất:

- Danh sách các biến động trên 1 tài khoản
- Danh sách tài khoản cùng số dư hiện tại theo từng loại tài khoản.
- Tình hình gửi, rút tiền theo từng loại tài khoản.
- Số dư của ngân hàng theo từng ngày của tháng.

Bài tập 4: Theo dõi kế hoạch sản lượng cao su**Lưu trữ:** Các thông tin về

Nông trường: Tên, diện tích các lô cao su theo từng năm.

Sản lượng kế hoạch theo tháng, năm của từng loại mủ.

Sản lượng thực tế theo ngày của từng loại mủ.

Tính toán:

Tỷ lệ đạt của từng loại mủ theo từng nông trường theo kế hoạch.

Kế hoạch dự kiến cho năm tới.

Kết xuất:

Báo cáo nhanh hàng ngày.

Báo cáo tháng.

Kế hoạch năm cho từng nông trường cho từng loại mủ.

Bài tập 5: Quản lý giải vô địch bóng đá

Lưu trữ: Các thông tin về

- Các đội bóng tham gia giải: Tên đội bóng, tên huấn luyện viên, các cầu thủ, sân nhà.
- Lịch thi đấu: đội tham dự, sân, thời gian
- Kết quả các trận đấu: Trọng tài, tỷ số, khán giả, các cầu thủ ra sân của 2 đội cùng vị trí tương ứng, việc ghi bàn, phạt thẻ.

Tra cứu: Cầu thủ, đội bóng

Tính toán:

- Tính điểm cho từng đội: mỗi trận thắng được 3 điểm, mỗi trận hòa được 1 điểm, mỗi trận thua được 0 điểm.
- Xếp hạng cho từng đội: Dựa vào các tiêu chuẩn: tổng số điểm, tổng số bàn thắng, hiệu số, đối kháng trực tiếp, bốc thăm.

Kết xuất:

- Danh sách các cầu thủ theo từng đội, vị trí.
- Lịch thi đấu.
- Bảng xếp hạng các đội bóng.
- Tổng kết việc ghi bàn của giải.
- Tình hình phạt thẻ các đội bóng.

Bài tập 6: Thi trắc nghiệm trên máy tính

Lưu trữ: Các thông tin về

- Thí sinh dự thi: Họ và tên, môn thi, ngày thi, địa chỉ, đề thi, bài làm, phòng thi.
- Câu hỏi trắc nghiệm: Nội dung câu hỏi, các câu trả lời có thể có, đáp án, mức độ khó, thang điểm, môn tương ứng.

Tính toán:

- Phát sinh các đề thi tương đương cho một đề thi đã chọn cho một môn thi nào đó (đề thi tương đồng có cùng các câu hỏi trắc nghiệm nhưng có số thứ tự khác nhau và trật tự các câu trả lời cũng khác nhau).
- Tính điểm thi cho từng thí sinh: Tổng điểm các câu hỏi với thang điểm tương ứng.

Kết xuất:

- Danh sách các thí sinh theo từng phòng thi.
- Đề thi.
- Bài làm của từng thí sinh cùng với điểm số.
- Danh sách kết quả thi của mỗi môn thi.
- Thống kê kết quả thi theo từng mức theo từng môn thi.
- Thống kê kết quả thi theo từng câu hỏi.

Bài tập 7: Quản lý trung tâm giới thiệu việc làm sinh viên**Lưu trữ:** Các thông tin về

- Sinh viên đăng ký tìm việc: Họ và tên, ngày sinh, địa chỉ, tình hình sức khỏe, quá trình học tập và bằng cấp, các công việc có thể đảm nhận, các yêu cầu khi tìm việc.
- Đơn vị đăng ký tìm người: Tên, địa chỉ, người đại diện, các công việc cùng yêu cầu tuyển dụng.
- Giới thiệu việc làm: Sinh viên, đơn vị, công việc, tình trạng.

Tra cứu:**Sinh viên tra cứu công việc**

- Loại công việc, Mức lương, Hình thức làm việc, Nơi làm việc

Đơn vị tuyển dụng tra cứu các sinh viên

- Bằng cấp chuyên môn, Sức khỏe, Phương tiện làm việc.

Tính toán:

- Các công việc thích hợp cho sinh viên đăng ký làm việc.
- Các sinh viên thích hợp cho công việc cần tuyển dụng của 1 đơn vị.

Kết xuất:

- Danh sách sinh viên đăng ký theo từng công việc.
- Danh sách số lượng sinh viên đăng ký theo từng loại công việc.
- Danh sách các đơn vị tuyển dụng theo từng công việc.
- Danh sách số lượng đơn vị tuyển dụng theo từng công việc.
- Thống kê tình hình giới thiệu việc làm thực hiện trong năm.

Bài tập 8: Phần mềm quản lý bán sách

- Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

Bài tập 9: Phần mềm quản lý bán vé chuyến bay

- Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

Bài tập 10: Phần mềm quản lý phòng mạch

- Khảo sát thực tế và rút ra yêu cầu cần phải làm cho đề tài

❖ YÊU CẦU THỰC HIỆN ĐỒ ÁN MÔN HỌC**Yêu cầu chung**

Mỗi sinh viên đăng ký thực hiện phần mềm. Kết quả gồm báo cáo viết, đĩa/CD (chương trình nguồn, EXE, báo cáo viết)

Cấu trúc báo cáo viết**1. Hiện trạng và yêu cầu**

- Hiện trạng:
 - Giới thiệu về thế giới thực liên quan
 - Mô tả quy trình các công việc liên quan đến đề tài
 - Mô tả các mẫu biểu có liên quan

- Mô tả các quy định ràng buộc có liên quan
 - Mô tả các quy định công thức tính có liên quan
- Yêu cầu:
- Danh sách các công việc sẽ được hỗ trợ thực hiện trên máy tính (dựa theo tóm tắt yêu cầu đã cho)

2. Mô hình hóa yêu cầu

Mô hình luồng dữ liệu theo yêu cầu:

- Sơ đồ luồng dữ liệu cho từng yêu cầu
- Mô tả chi tiết cho từng sơ đồ

Mô hình luồng dữ liệu chung cho toàn bộ hệ thống

Sơ đồ luồng dữ liệu chung cho toàn bộ hệ thống

3. Thiết kế phần mềm

Thiết kế dữ liệu:

- Sơ đồ luận lý
- Danh sách các thành phần của sơ đồ

Stt	Tên	Loại	Ý nghĩa	Ghi chú

- Danh sách các thuộc tính của từng thành phần

Tên thành phần:

Stt	Tên	Loại	Kiểu	Miền giá trị	Ý nghĩa

Thiết kế giao diện:

Stt	Mã số	Loại	Ý nghĩa	Ghi chú

- Mô tả chi tiết từng màn hình

- Nội dung

- Danh sách biến cố và xử lý tương ứng trên màn hình

Số thứ tự	Biến cố	Ý nghĩa	Xử lý tương ứng	Mã số xử lý

Thiết kế xử lý

- Danh sách các xử lý (Các xử lý quan trọng)

Số thứ tự	Mã số	Loại	Ý nghĩa	Ghi chú

- Mô tả chi tiết từng xử lý
 - Sơ đồ luồng dữ liệu
 - Mô tả chi tiết sơ đồ

4. Cài đặt thử nghiệm

- Cài đặt
 - Danh sách tình trạng cài đặt các chức năng (mức độ hoàn thành)

Số thứ tự	Chức năng	Mức độ hoàn thành	Ý nghĩa

- Thử nghiệm
 - Nội dung các bảng dữ liệu
 - Một số test-case chạy thử nghiệm
 - Các báo biểu màn hình cùng các số liệu tương ứng

5. Tổng kết

- Các kết quả đã thực hiện
- Đánh giá ưu khuyết điểm
- Hướng mở rộng tương lai

PHỤ LỤC B – THAM KHẢO

PHẦN MỀM QUẢN LÝ THƯ VIỆN

❖ Mô tả chi tiết các thuộc tính

Độc giả:

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MDG	Chuỗi	Khóa chính		
2	MLDG	Chuỗi	Khóa ngoại		
3	HoTen	Chuỗi			
4	NgaySinh	Ngày			
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

Sách

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MSACH	Chuỗi	Khóa chính		
2	MTG	Chuỗi	Khóa ngoại		
3	MNXB	Chuỗi	Khóa ngoại		
4	MLSACH	Chuỗi	Khóa ngoại		
5	MNN	Chuỗi	Khóa ngoại		
6	TenSach	Chuỗi			
7	Ngaymua	Ngày			
8	SoTrang	Số	>0		

Phiếu mượn

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MPHM	Chuỗi	Khóa chính		
2	NgayMuon	Ngày			

Chi tiết mượn

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MPHM	Chuỗi	Khóa ngoại		
2	MSACH	Chuỗi	Khóa ngoại		
3	NgayTra	Ngày			

Loại sách

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLSACH	Chuỗi	Khóa ngoại		
2	TenLoai	Chuỗi			
3	Ghi Chu	Chuỗi			

Loại độc giả

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MLDG	Chuỗi	Khóa chính		
2	Tenloai	Chuỗi			
3	GhiChu	Chuỗi			

Nhà xuất bản

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNXB	Chuỗi	Khóa chính		
2	Tenloai	Chuỗi			
3	GhiChu	Chuỗi			

Tác giả

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MTG	Chuỗi	Khóa chính		
2	Ten	Chuỗi			
3	GhiChu	Chuỗi			

Ngôn ngữ

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MNN	Chuỗi	Khóa chính		
2	Ten	Chuỗi			
3	GhiChu	Chuỗi			

❖ Mô hình chi tiết các thành phần trong sơ đồ lớp

Đối tượng Độc Giả

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MDG	Chuỗi			
2	Loại độc giả	Số	giá trị rời rạc		
3	HoTen	Chuỗi			

4	NgaySinh	Ngày	từ 19 đến 90		
5	DiaChi	Chuỗi			
6	DienThoai	Chuỗi			

Đối tượng Sách

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	MSACH	Chuỗi			
2	Loại sách	Số			
3	Tác giả	Chuỗi			
4	Nhà xuất bản	Chuỗi			
5	Ngày nhập	Chuỗi			
6	TenSach	Chuỗi			
7	Ngôn ngữ	Ngày			
8	SoTrang	Số	>0		

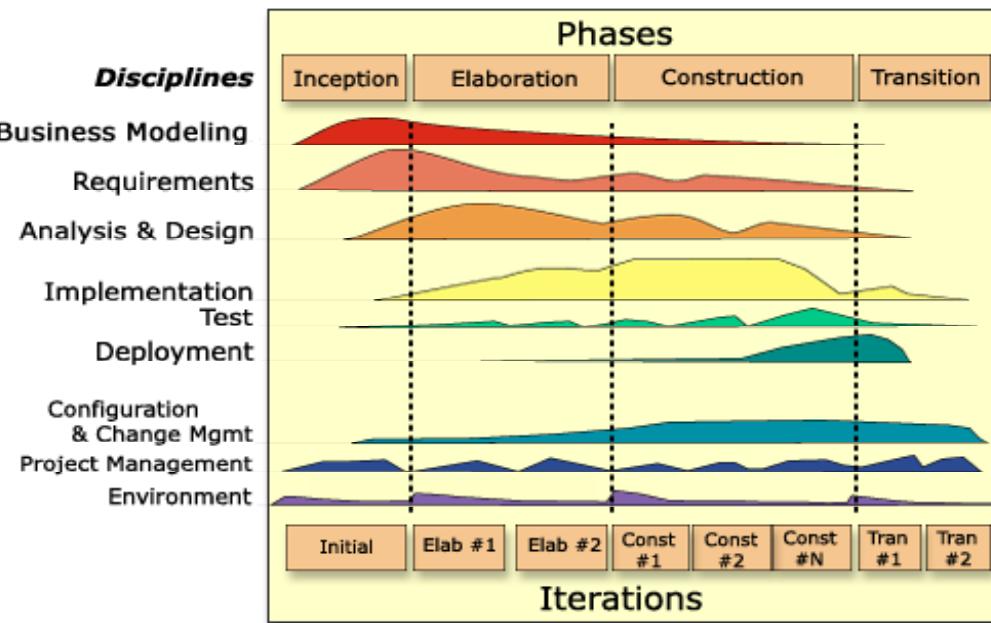
Quan hệ mượn

Stt	Thuộc tính	Kiểu	Miền giá trị	Ý nghĩa	Ghi chú
1	Ngày mượn	Ngay	>=Ngày nhập		
2	Ngày trả	Ngay	>=Ngày mượn		

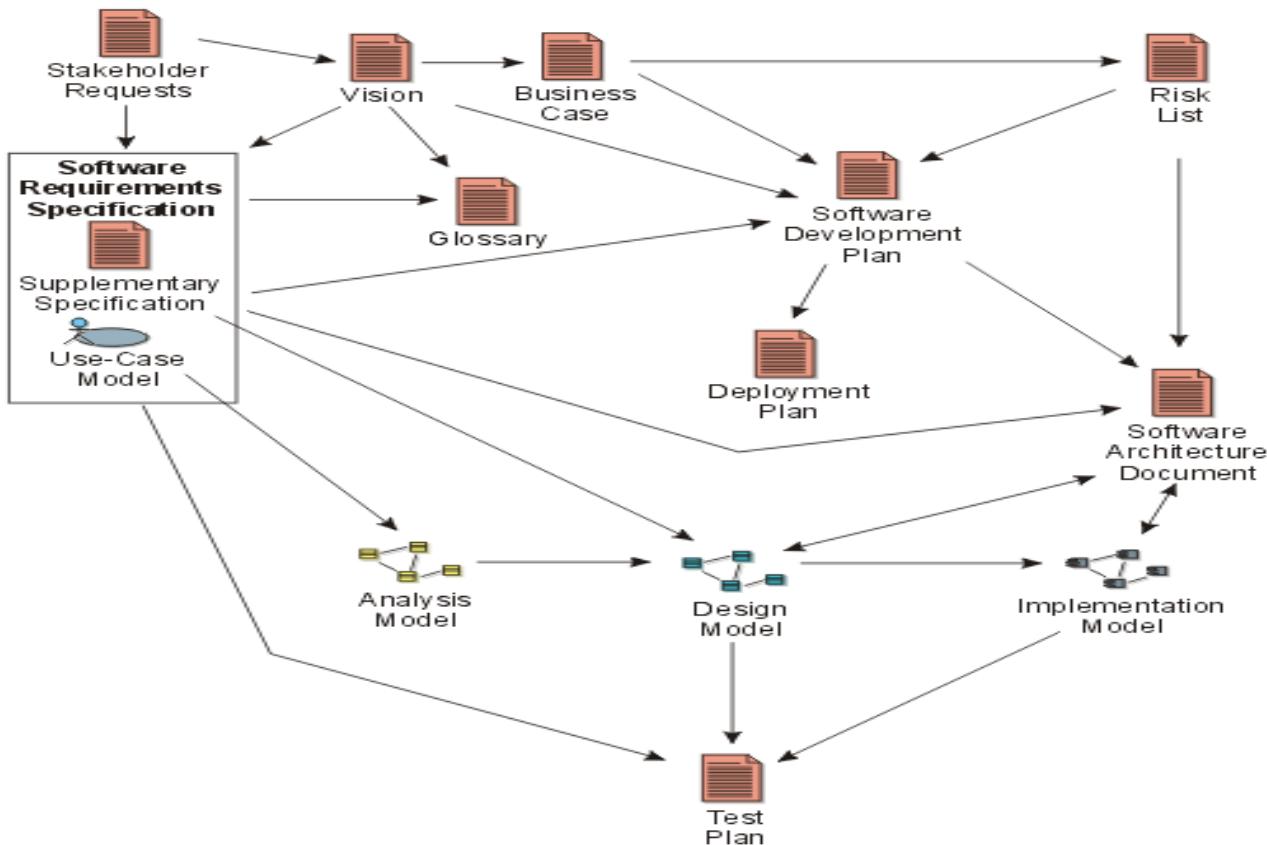
PHỤ LỤC C – QUY TRÌNH RUP

Một số hình ảnh minh họa về quy trình RUP (**Rational Unified Process**, <http://www.ibm.com/software/rational/rup>) phổ biến trên thế giới:

❖ Phần A – Quy trình chung

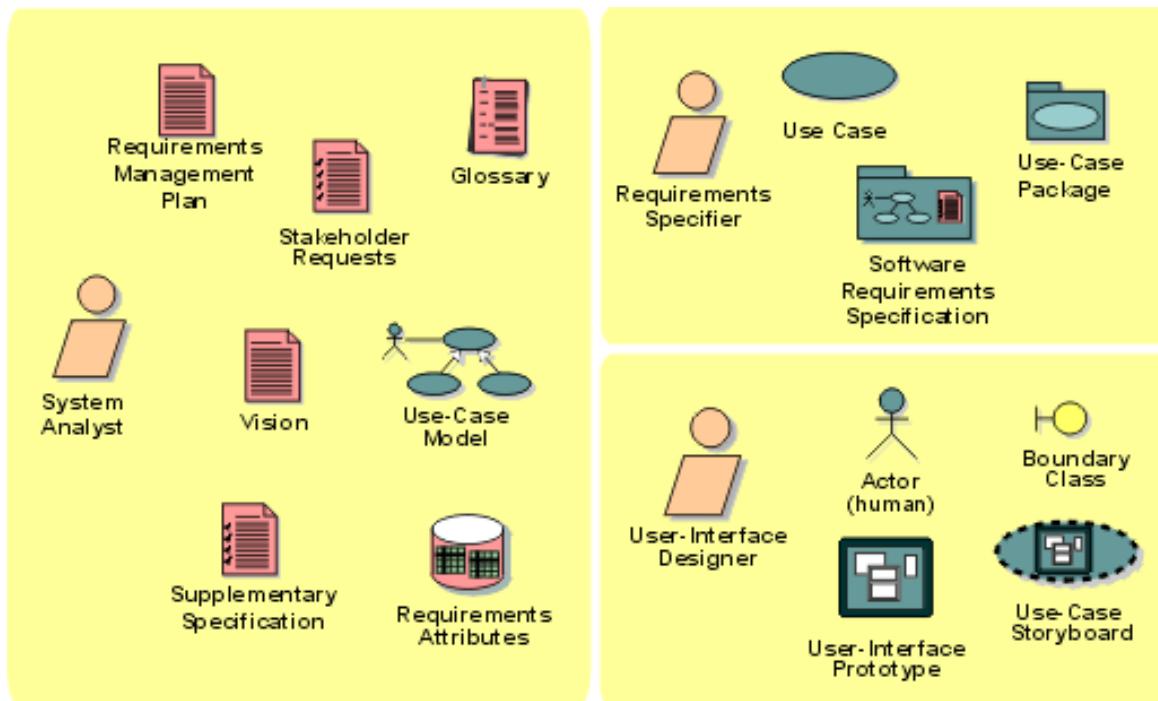


Hình 10.5: Major artifacts

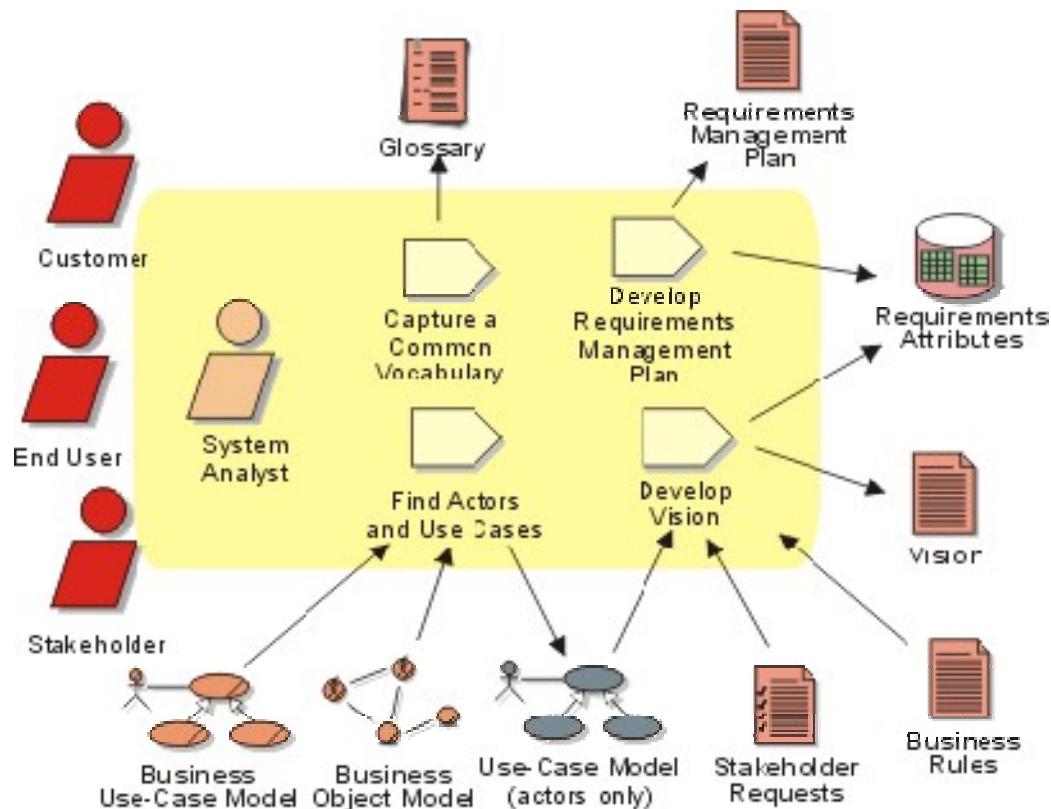


Hình 10.6: Information flow

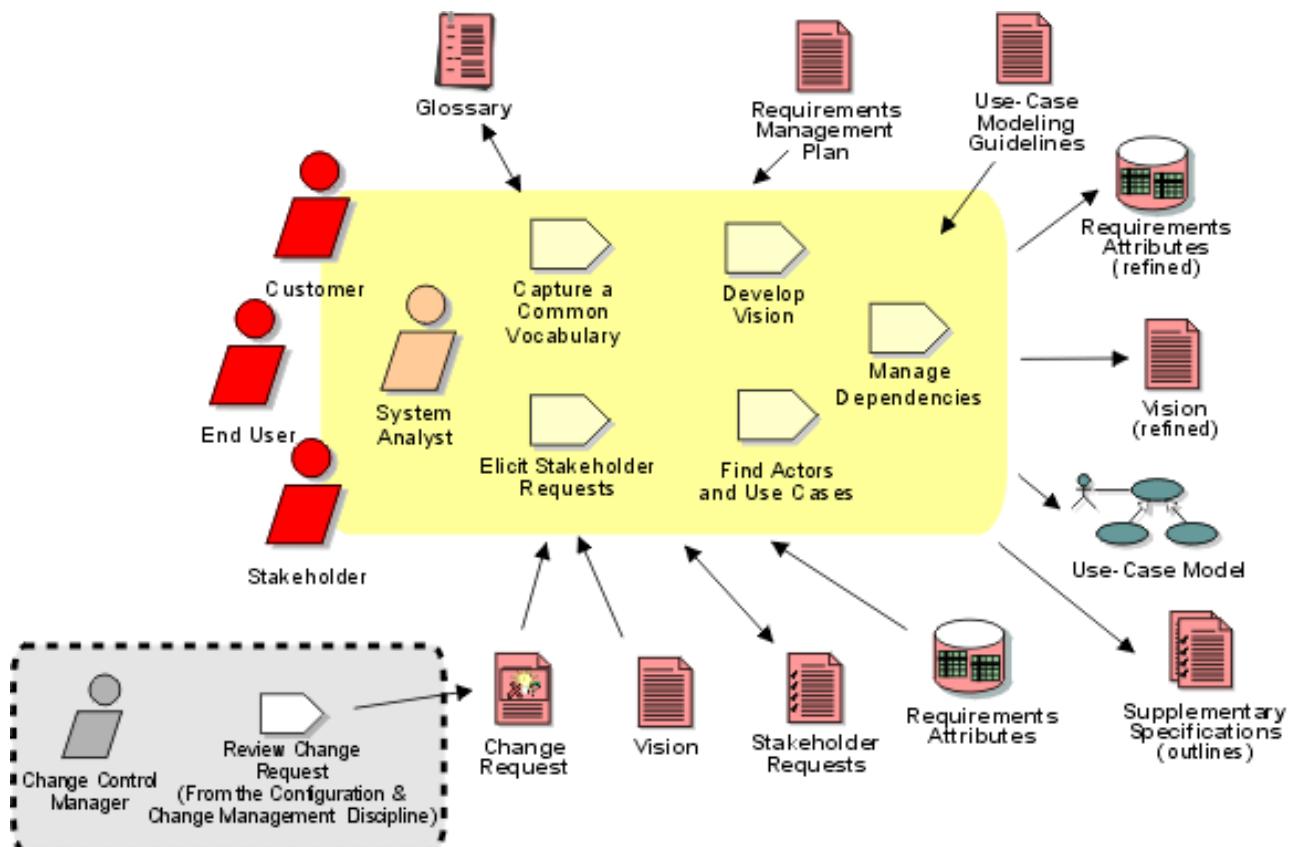
❖ Phần B – Quy trình phân tích yêu cầu



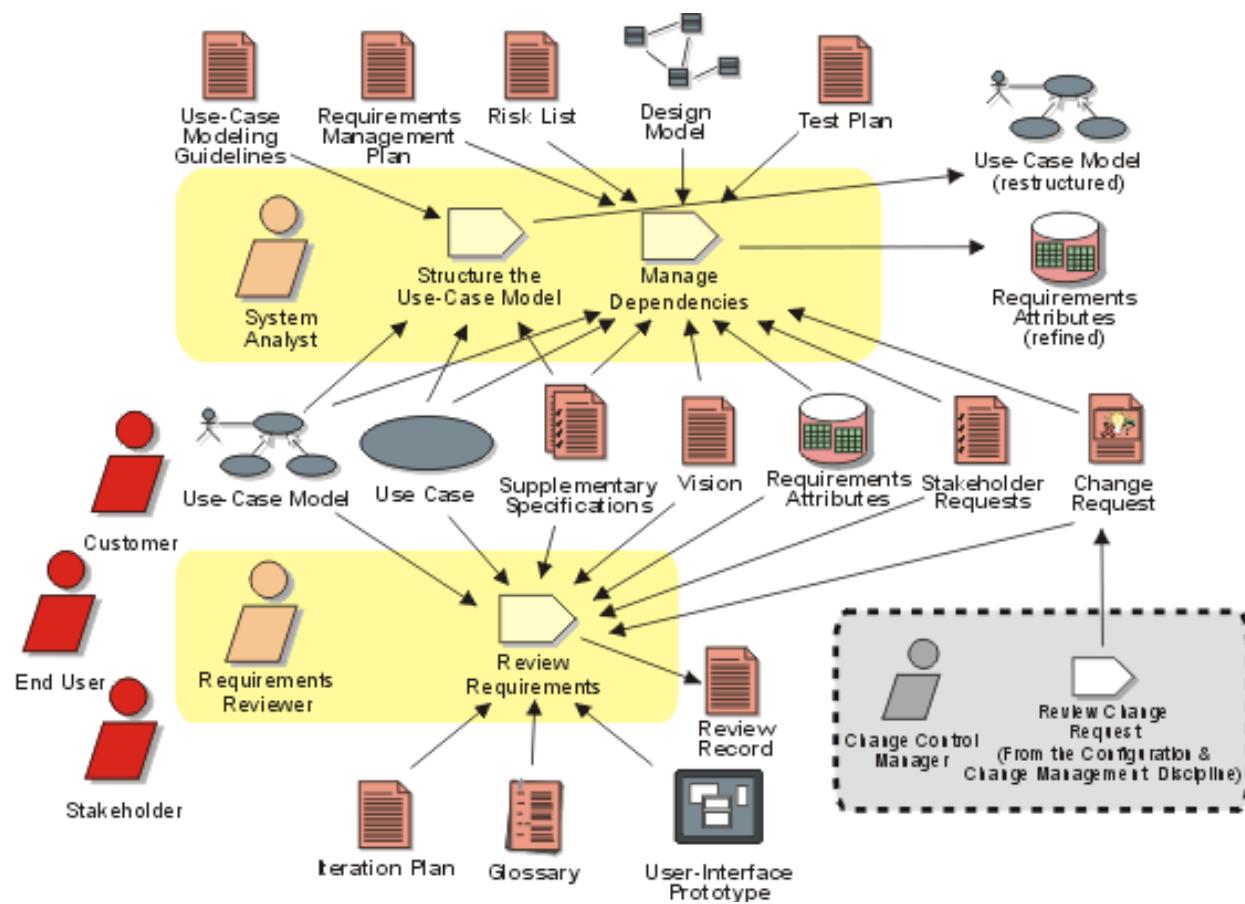
Hình 10.7: Requirements Artifact Set



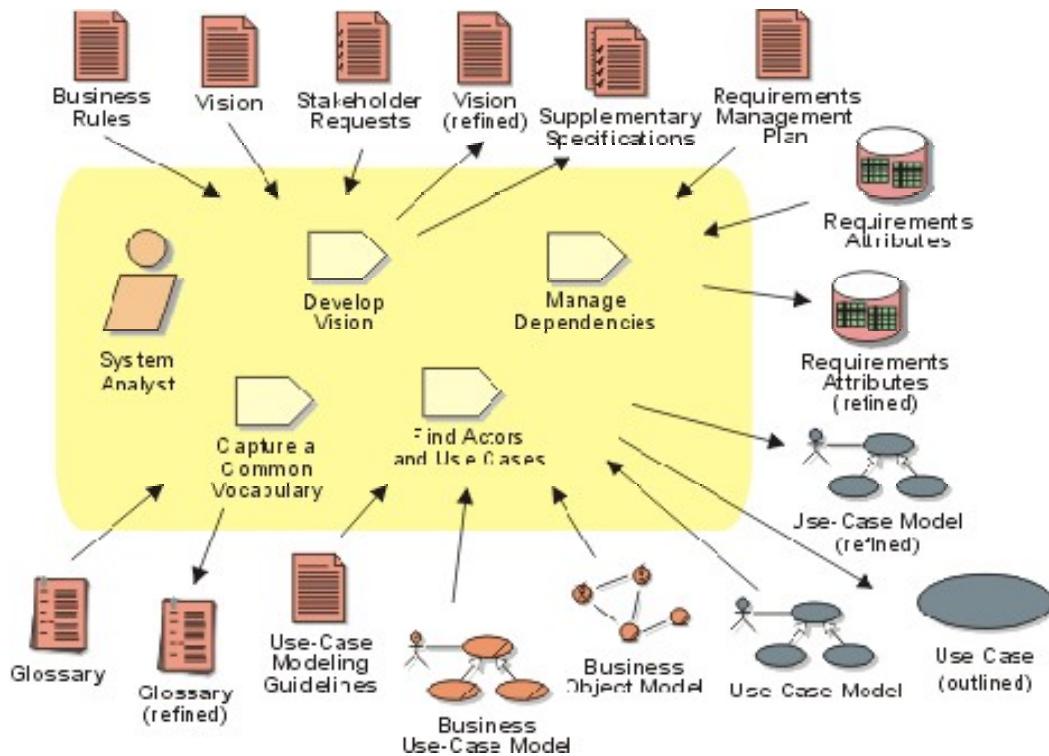
Hình 10.8: Workflow Detail: Analyze the Problem



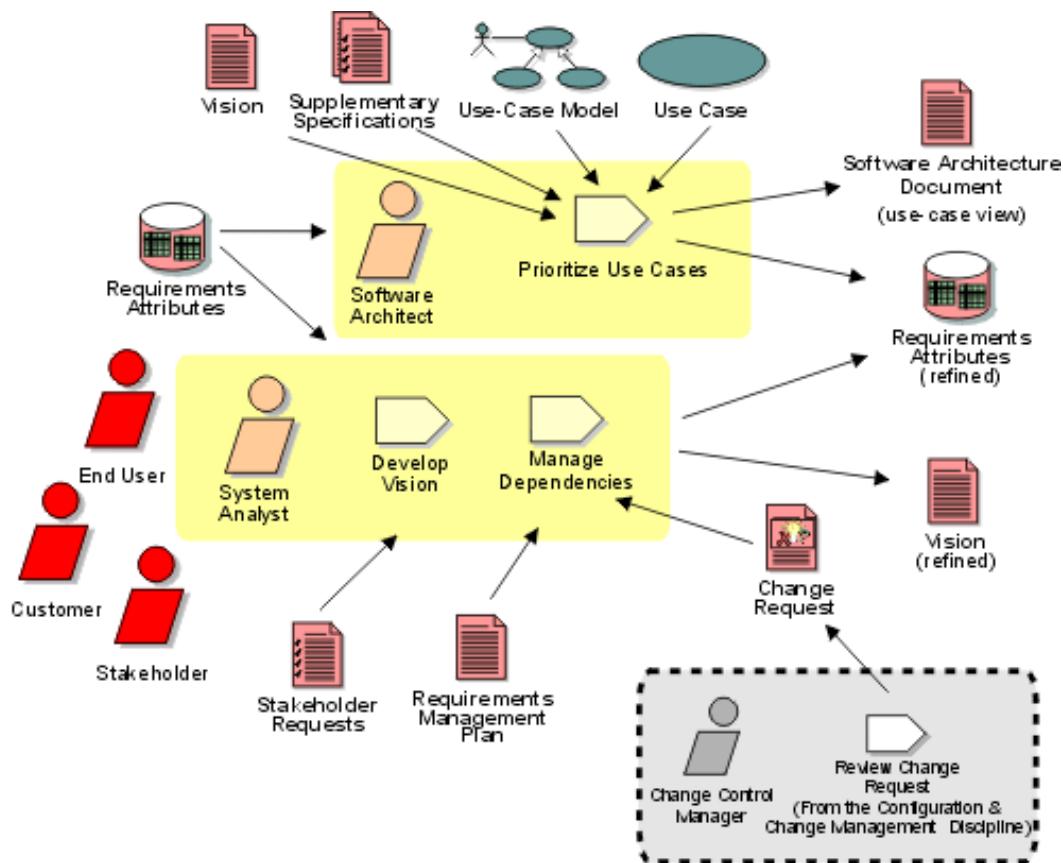
Hình 10.9: Workflow Detail: Understand Stakeholder Needs



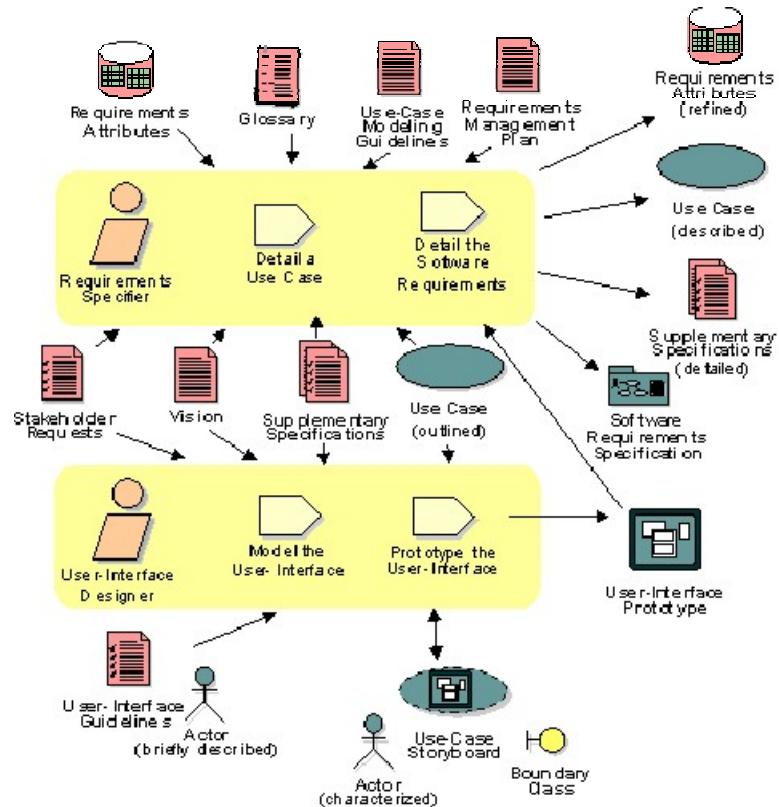
Hình 10.10: Workflow Detail: Manage Changing Requirements



Hình 10.11: Workflow Detail: Define the System

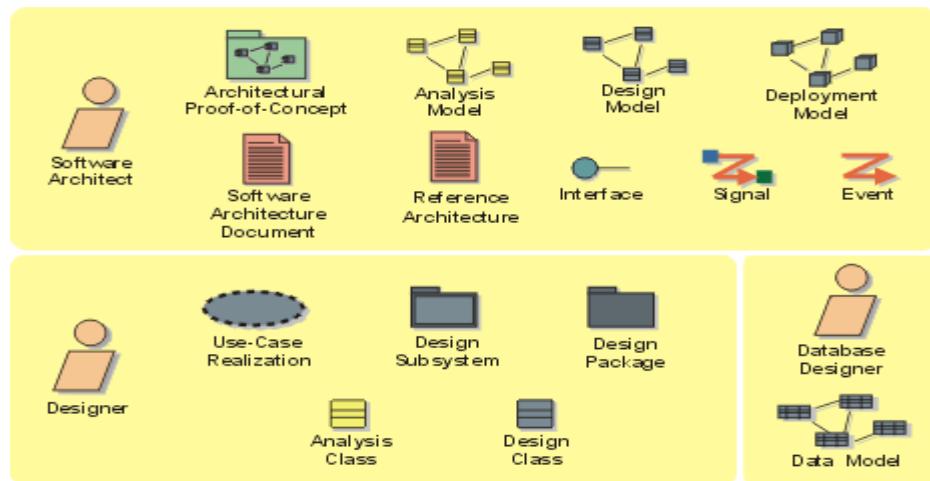


Hình 10.12: Workflow Detail: Manage the Scope of the System

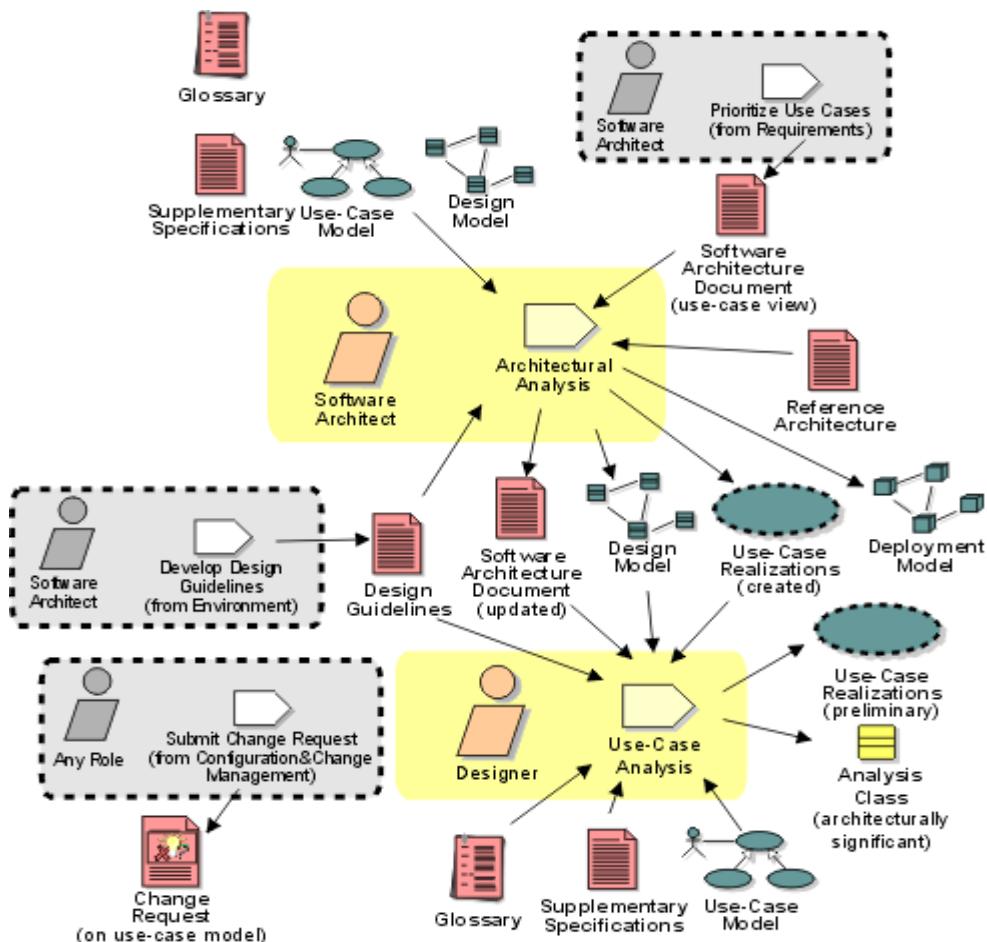


Hình 10.13: Workflow Detail: Refine the System Definition

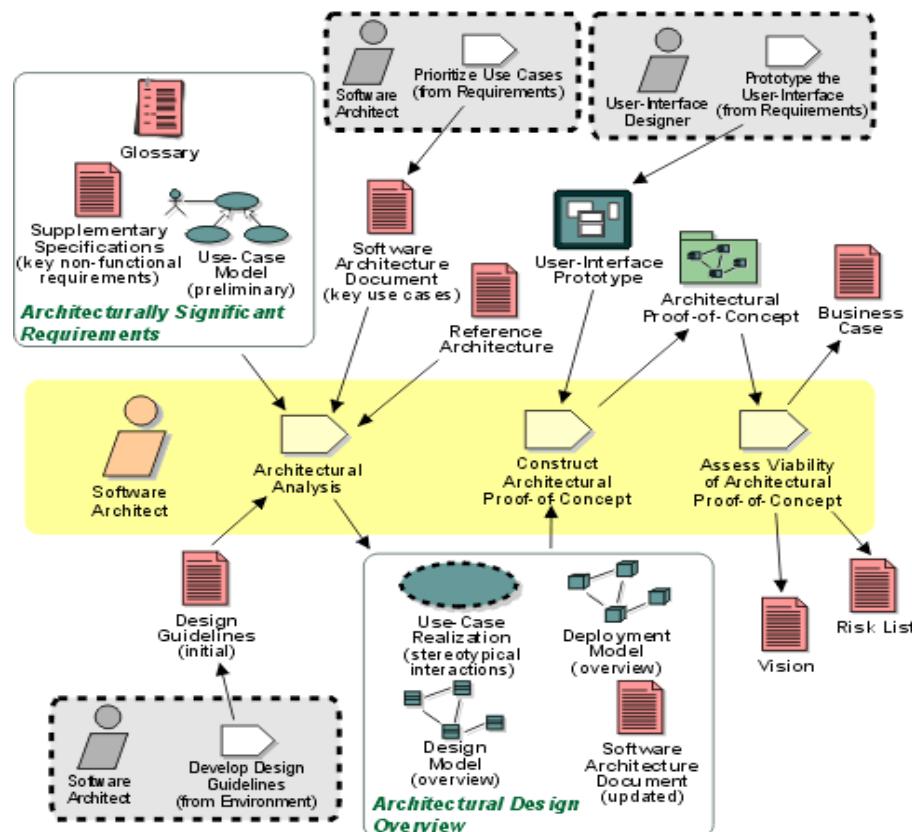
❖ **Phân C – Quy trình phân tích thiết kế**



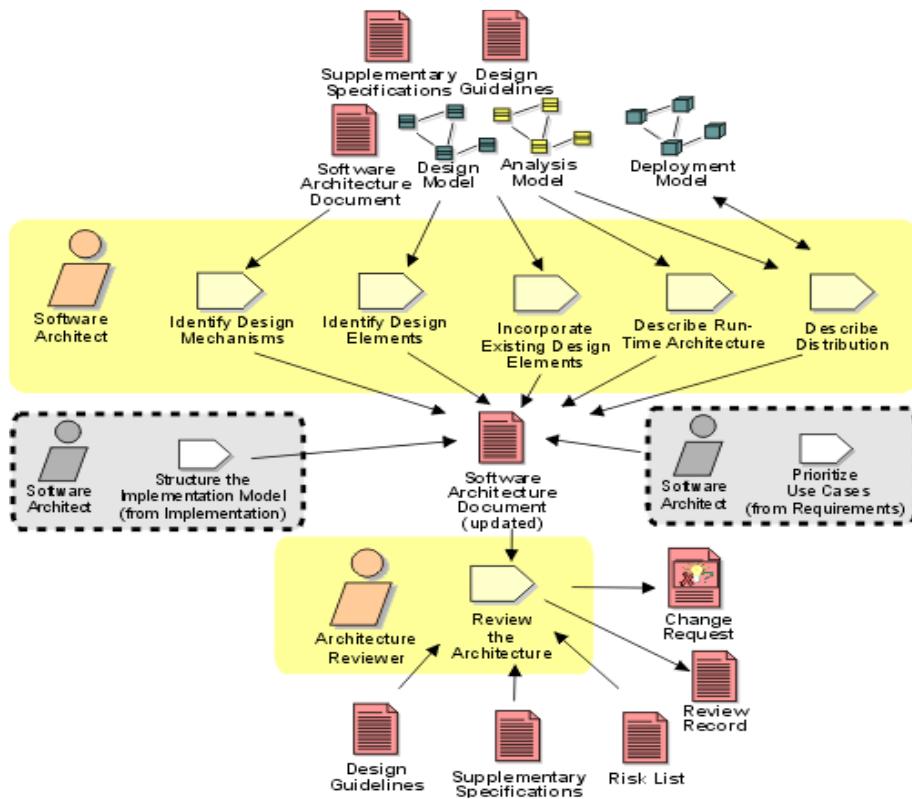
Hình 10.14: Analysis & Design Artifact Set



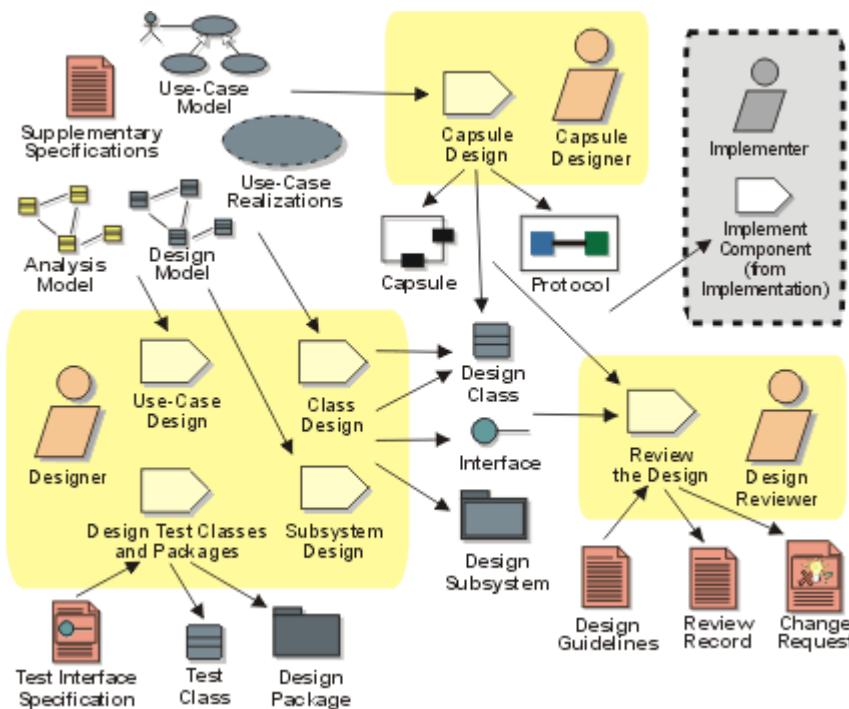
Hình 10.15: Workflow Detail: Define a Candidate Architecture



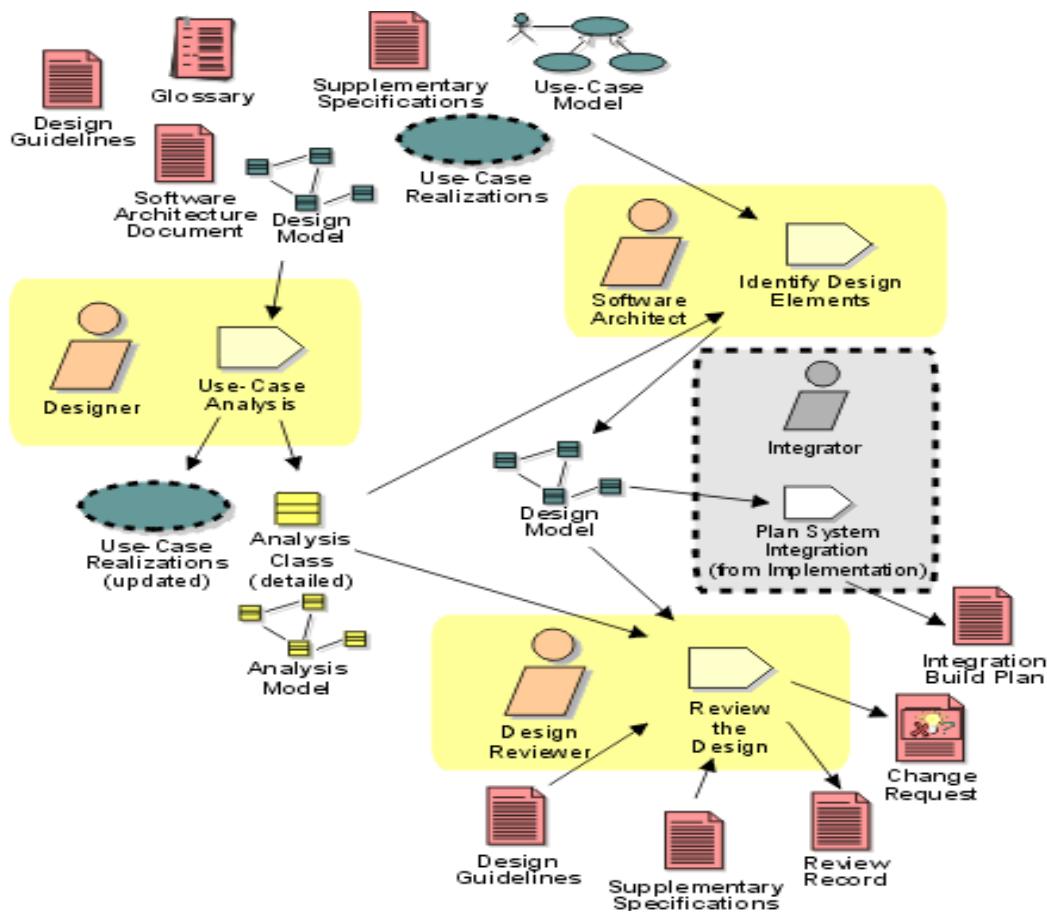
Hình 10.16: Workflow Detail: Perform Architectural Synthesis



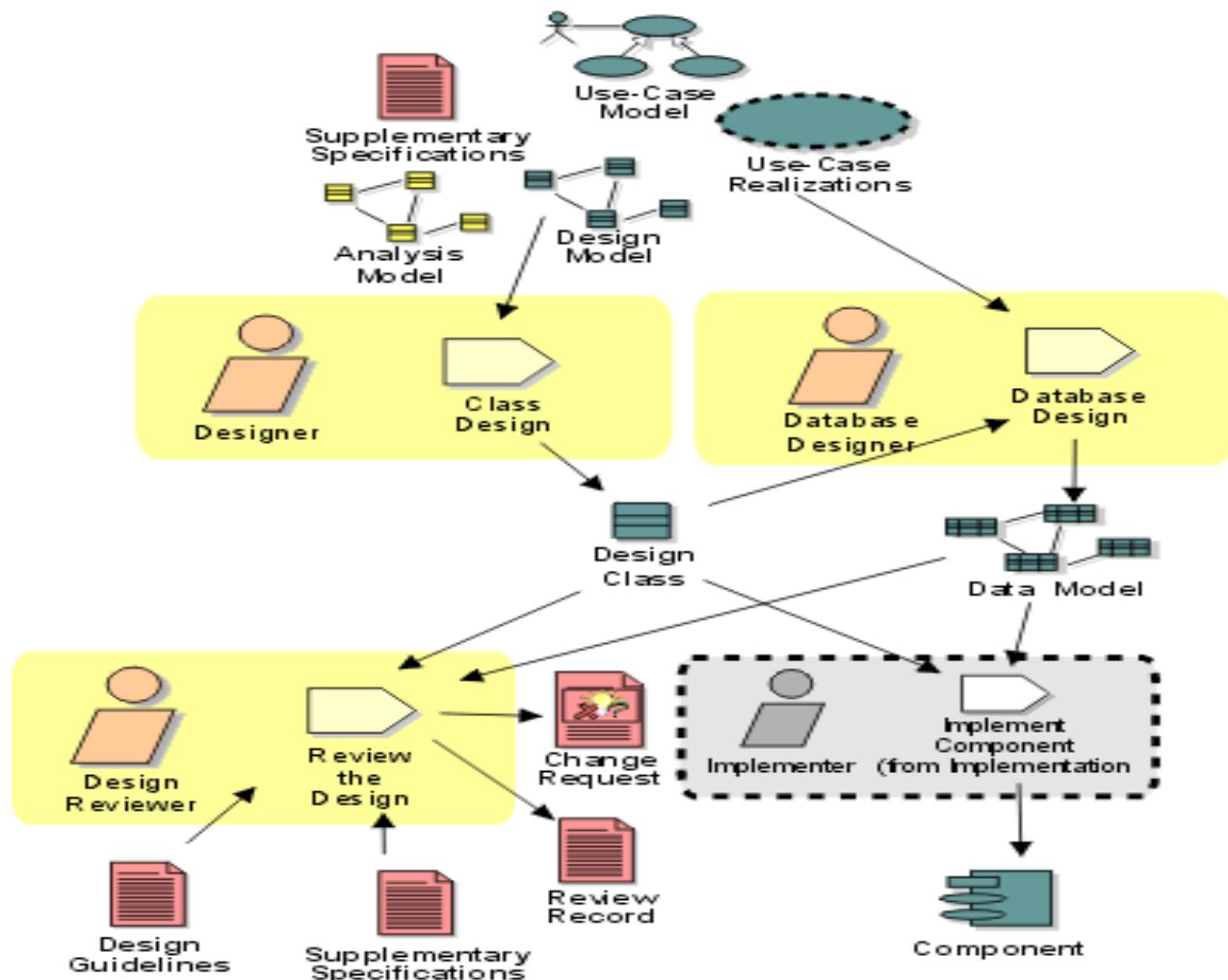
Hình 10.17: Workflow Detail: Refine the Architecture



Hình 10.18: Workflow Detail: Design Components

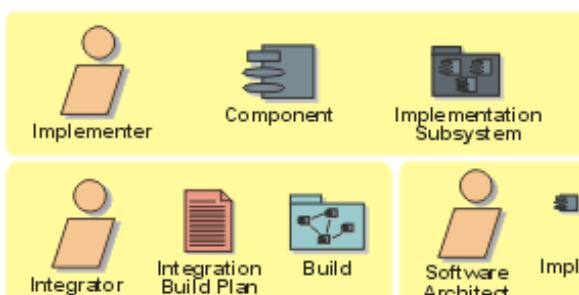


Hình 10.19: Workflow Detail: Analyze Behavior

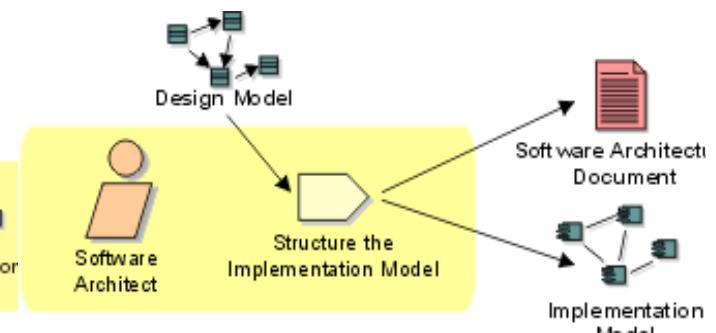


Hình 10.20: Workflow Detail: Design the Database

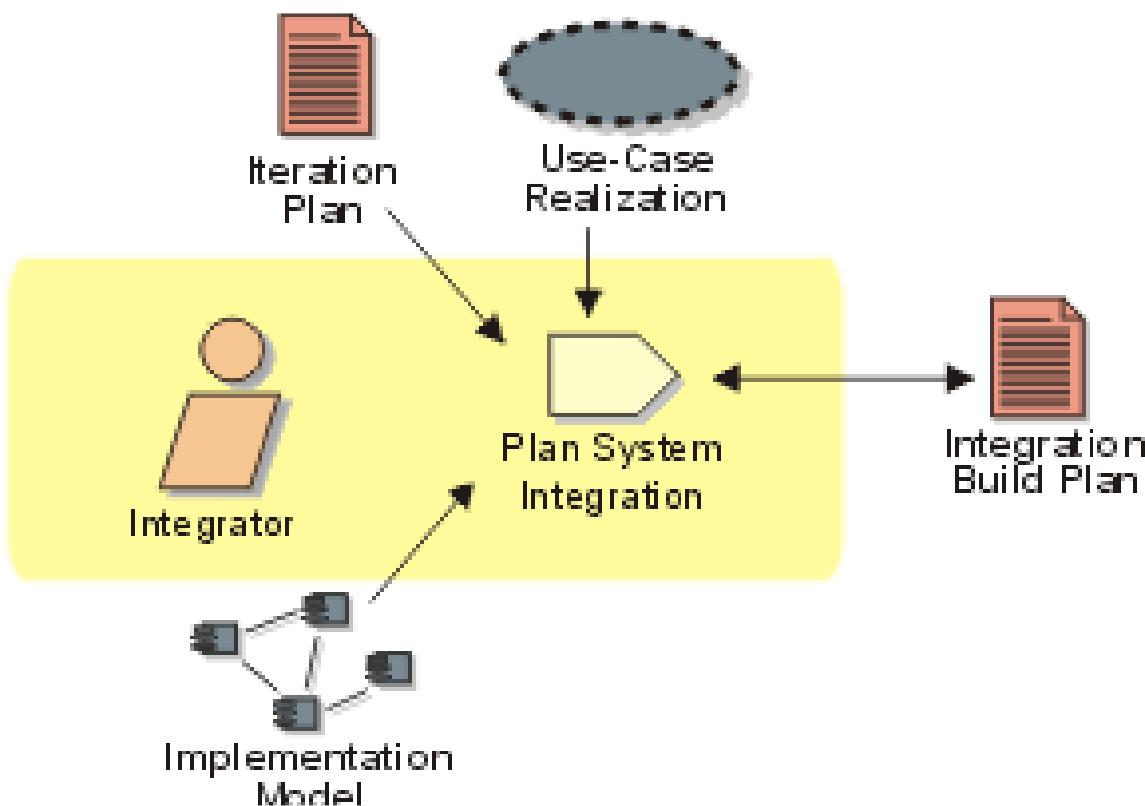
❖ Phần D – Quy trình hiện thực



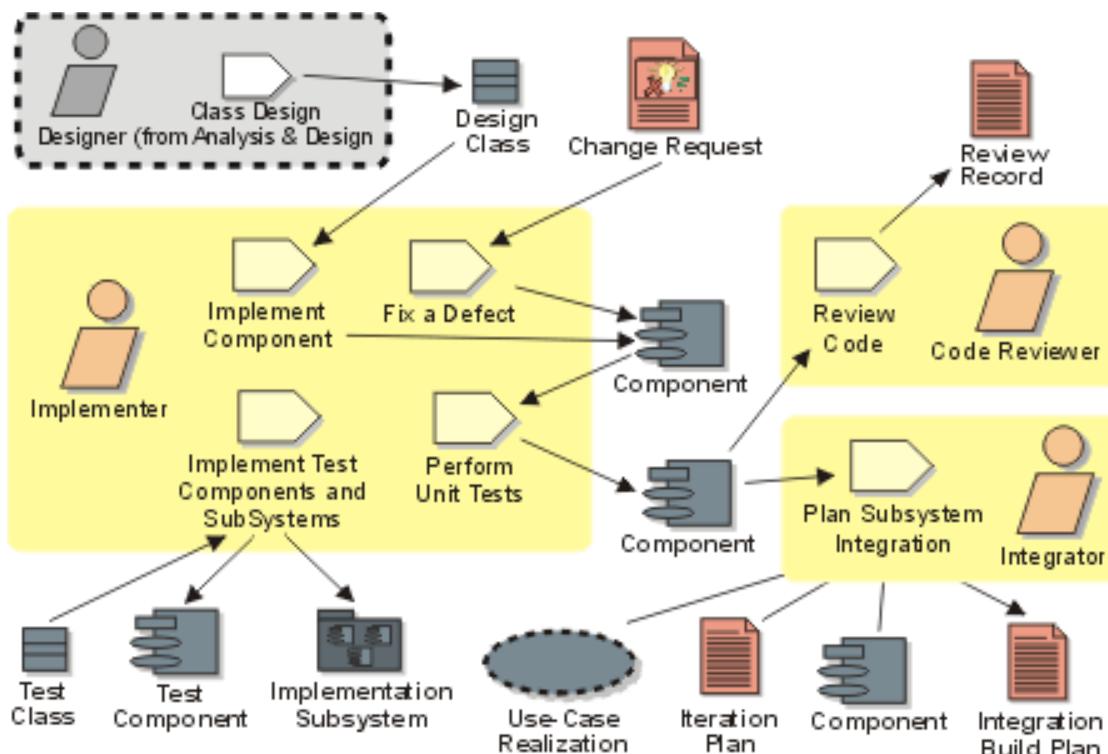
Hình 10.21: Implementation Artifact Set



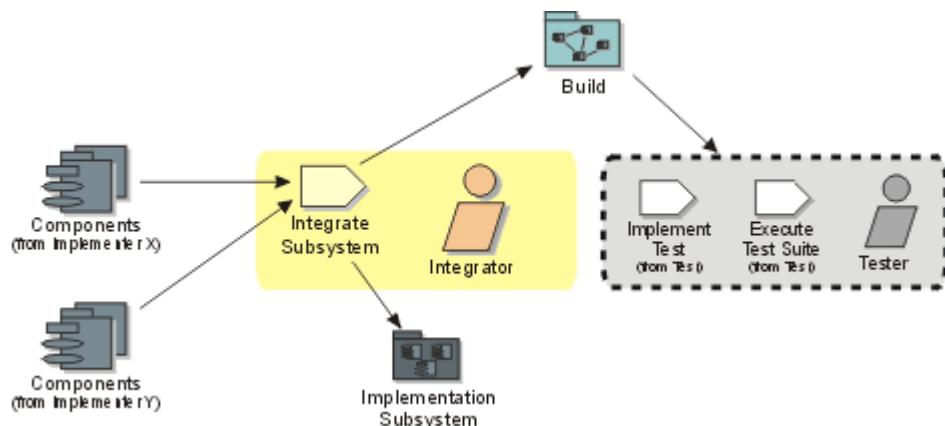
Hình 10.22: Workflow Detail: Structure the Implementation Model



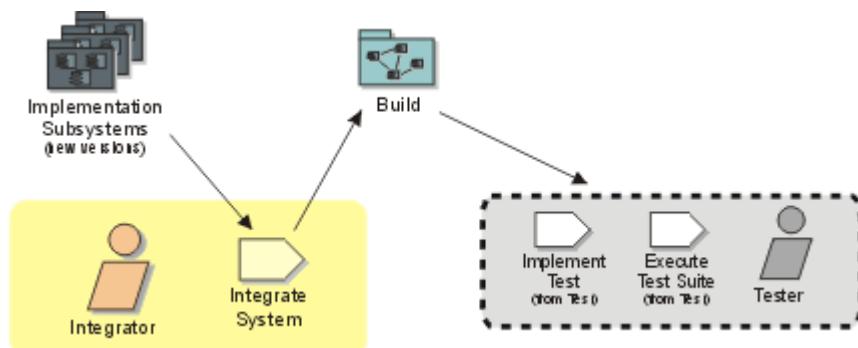
Hình 10.23: Workflow Detail: Plan the Integration



Hình 10.24: Workflow Detail: Implement Components

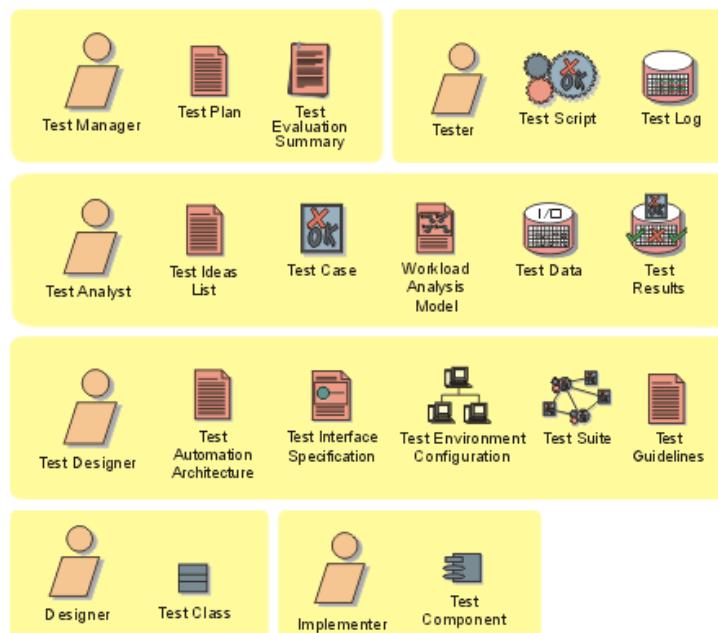


Hình 10.25: Workflow Detail: Integrate Each Subsystem

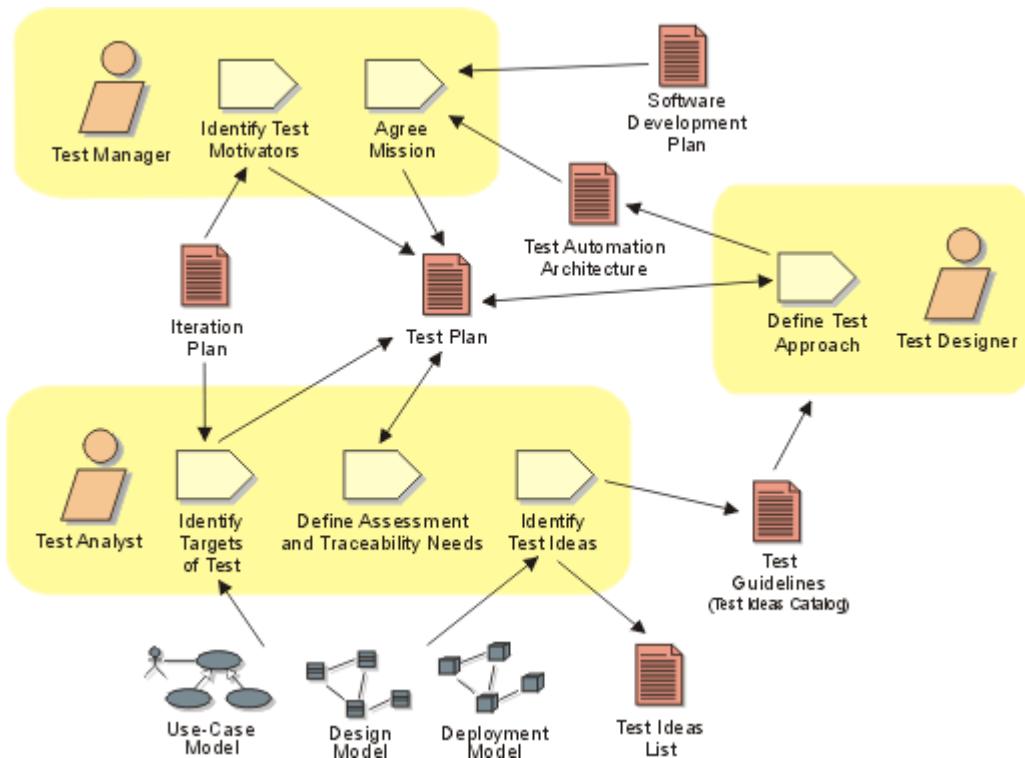


Hình 10.26: Workflow Detail: Integrate the System

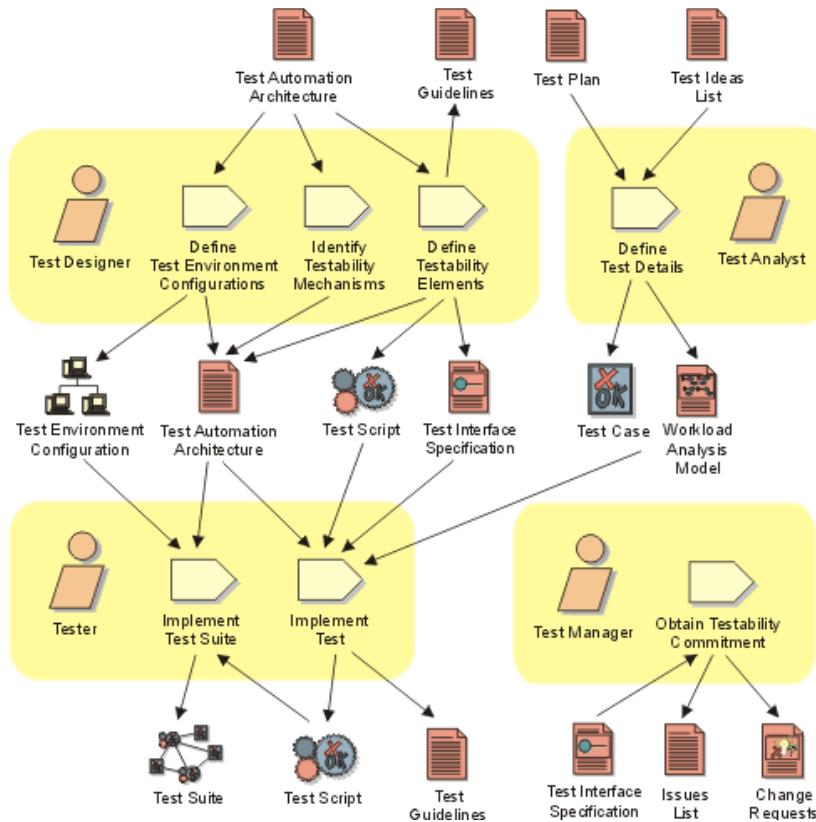
❖ Phần E – Quy trình kiểm thử



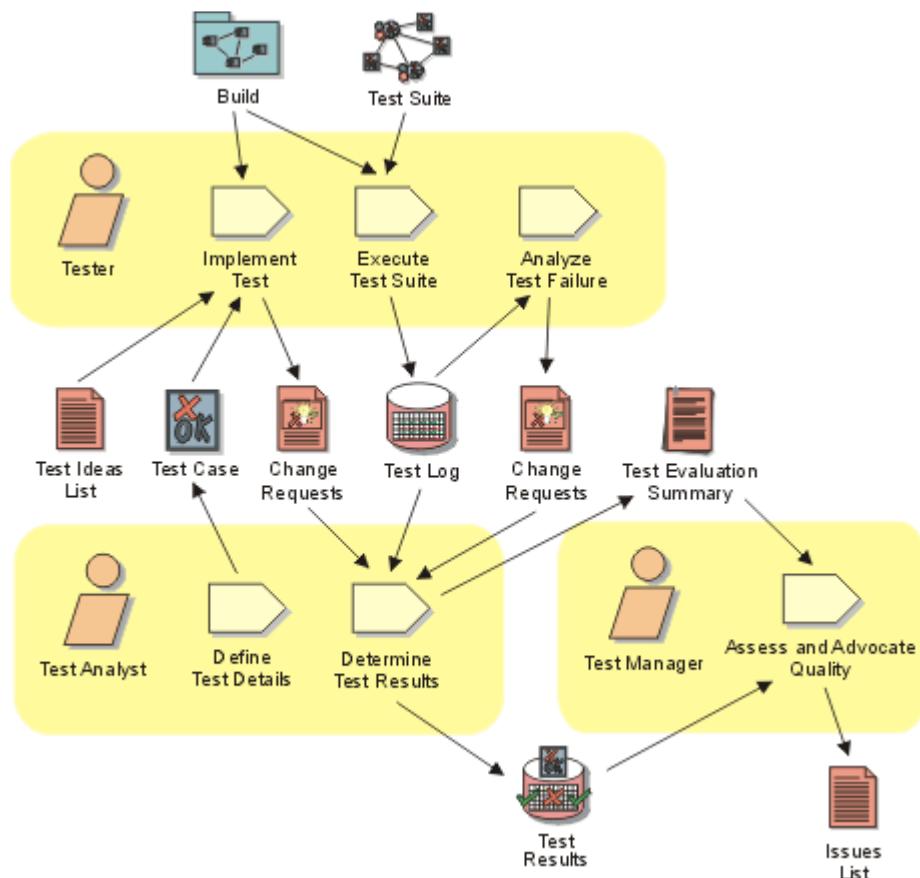
Hình 10.27: Test Artifact Set



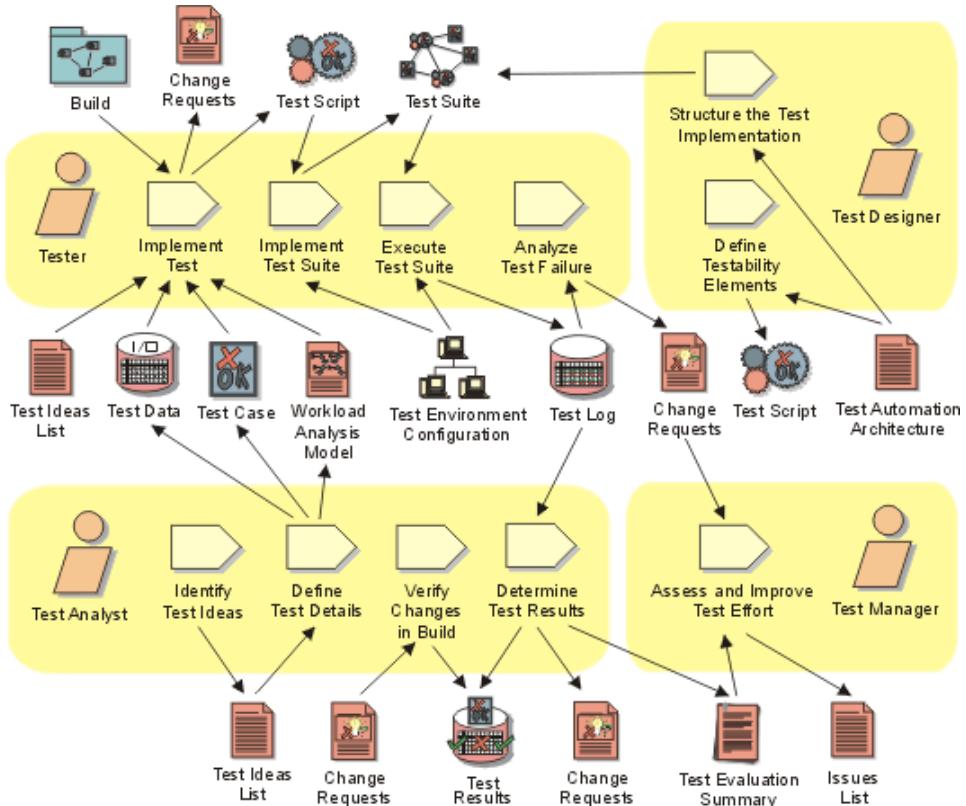
Hình 10.28: Workflow Detail: Define Evaluation Mission



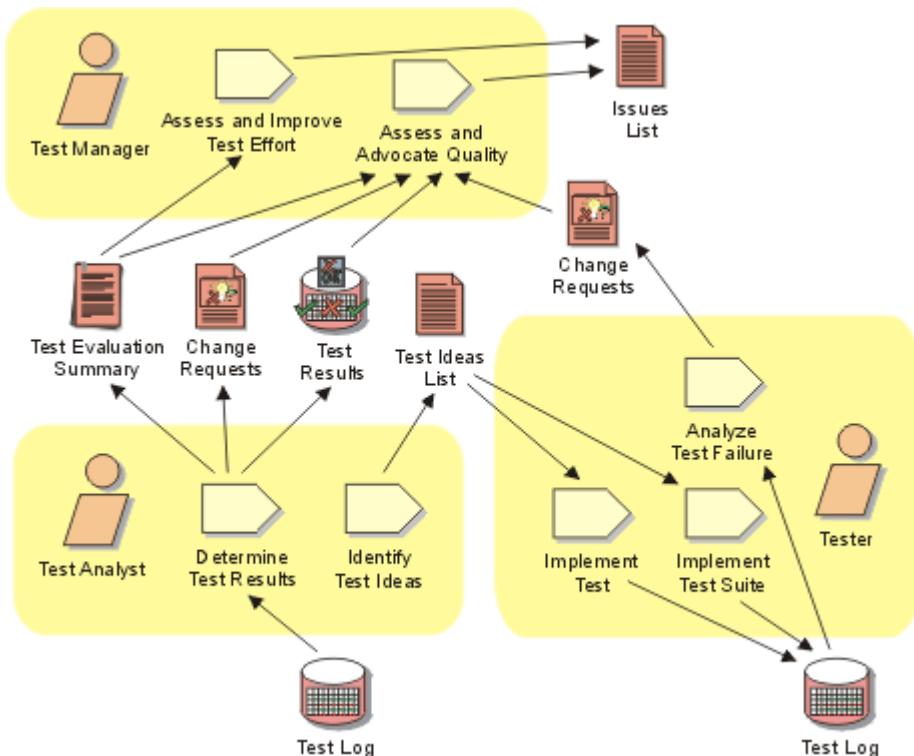
Hình 10.29: Workflow Detail: Verify Test Approach



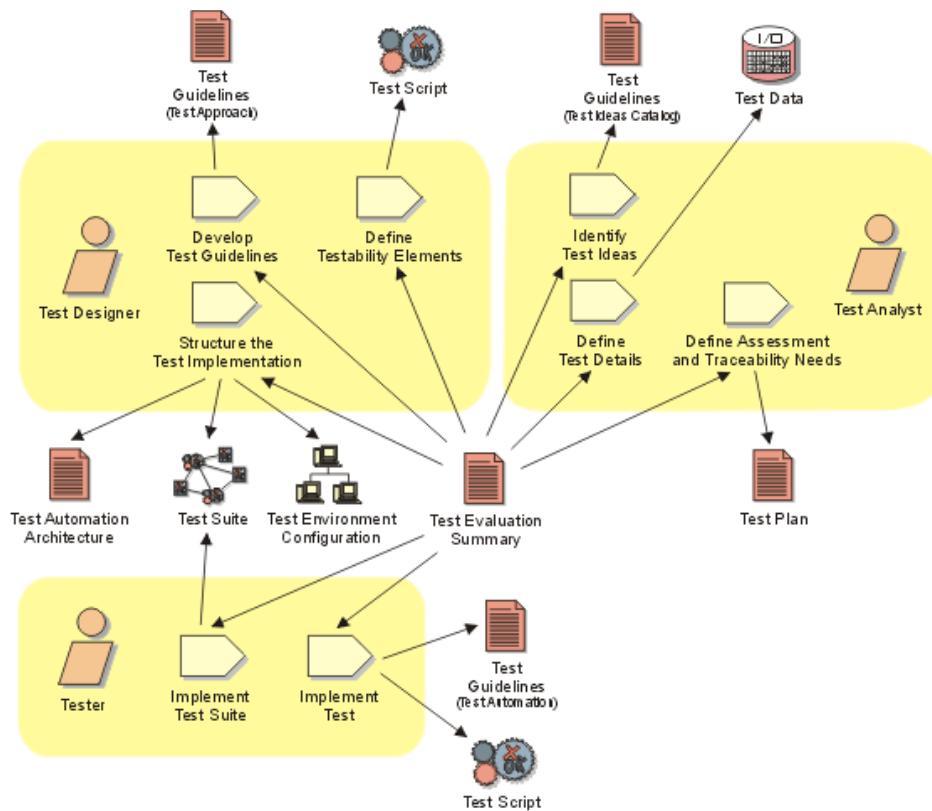
Hình 10.30: Workflow Detail: Validate Build Stability



Hình 10.31: Workflow Detail: Test and Evaluate



Hình 10.32: Workflow Detail: Achieve Acceptable Mission



Hình 10.33: Workflow Detail: Improve Test Assets

TÀI LIỆU THAM KHẢO

1. David Gustafson (2002). *Schaum's Outline of Software Engineering*. McGraw Hill Professional.
2. Roger S. Pressman (2010). *Software Engineering: A Practitioner's Approach*, 7/e. R. S. Pressman & Associates, Inc.
3. James Rumbaugh, Michael Blaha, Wiliam Premerlani, Frederick Eddy, Wiliam Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice-Hall International Editions.
4. Ian Sommerville (2010). *Software Engineering (9th)*. Addison-Wesley.
5. Hình ảnh minh họa từ nguồn Google Images.
6. Ian Lewis, Bruce Nielson (1999). *McSd Test Success: Analyzing Requirements and Defining Solution Architectures*. Sybex Inc.
7. Inc. Syngress Media. (1999). *MCSD Analyzing Requirements Study Guide*. Osborne McGraw-Hill.
8. Nguyễn Xuân Huy (1994). *Công nghệ phần mềm*. ĐH Tổng hợp Tp. HCM.
9. Nguyễn Tiến Huy. *Bài giảng Nhập môn Công nghệ Phần mềm*. ĐH Khoa học Tự nhiên Tp.HCM.
10. Nguyễn Tuân Huy (2003). *Quá trình phát triển phần mềm thống nhất* (biên dịch). Nhà xuất bản Thống kê.
11. Nguyễn Chánh Thành. *Bài giảng Công nghệ Phần mềm*. Đại học Kỹ thuật Công Nghệ TPHCM.
12. Đỗ Thị Thanh Tuyền. *Bài giảng Nhập môn Công nghệ Phần mềm*, Đại học Công nghệ Thông tin TPHCM.
13. Phạm Ngọc Hùng. *Bài giảng Công nghệ Phần mềm*, Đại học Công nghệ Hà nội.
14. Các giáo trình, bài giảng môn Công nghệ Phần mềm của Khoa Công nghệ Thông tin hay Tin học của các trường Đại học ở Việt nam và trên thế giới.