

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



PHÂN TÍCH THIẾT KẾ HỆ THỐNG THÔNG TIN

(Dùng cho sinh viên hệ đào tạo đại học từ xa)

Lưu hành nội bộ

cuu duong than cong . com

HÀ NỘI - 2007

PHÂN TÍCH & THIẾT KẾ HỆ THỐNG THÔNG TIN

Trần Đình Quế
Nguyễn Mạnh Sơn

MỤC LỤC

MỤC LỤC	ii
LỜI NÓI ĐẦU	1
CHƯƠNG 1 MỞ ĐẦU	3
1.2 KHÁI QUÁT VÒNG ĐỜI PHÁT TRIỂN HỆ THỐNG THÔNG TIN	4
1.3 CÁC CÁCH TIẾP CẬN PHÂN TÍCH THIẾT KẾ HỆ THỐNG	7
1.3.1 Phương pháp hướng cấu trúc	8
1.3.2 Phương pháp hướng đối tượng	9
1.5 CÁC KHÁI NIỆM CƠ BẢN CỦA HƯỚNG ĐỐI TƯỢNG	10
1.6 CÁC BƯỚC PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG	11
TỔNG KẾT CHƯƠNG 1	13
CÂU HỎI VÀ BÀI TẬP	13
CHƯƠNG 2 : UML VÀ CÔNG CỤ PHÁT TRIỂN HỆ THỐNG	15
2.1 GIỚI THIỆU VỀ UML	15
2.1.1 Lịch sử ra đời của UML	15
2.1.2 UML – Ngôn ngữ mô hình hoá hướng đối tượng	16
2.1.3 Các khái niệm cơ bản trong UML	17
2.2 CÁC BIỂU ĐỒ UML	20
2.2.1 Biểu đồ use case	22
2.2.2 Biểu đồ lớp	24
2.2.3 Biểu đồ trạng thái	30
2.2.4 Biểu đồ tương tác dạng tuần tự	33
2.2.5 Biểu đồ tương tác dạng cộng tác	35
2.2.6 Biểu đồ hoạt động	36
2.2.7 Biểu đồ thành phần	39
2.2.8 Biểu đồ triển khai hệ thống	40
2.3 GIỚI THIỆU CÔNG CỤ RATIONAL ROSE	41
TỔNG KẾT CHƯƠNG 2	44
CÂU HỎI – BÀI TẬP	45
CHƯƠNG 3: PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG	46
3.1 TỔNG QUAN VỀ PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG	46
3.1.1 Vai trò của pha phân tích	46
3.1.2 Các bước phân tích hướng đối tượng	47
3.1.3 Ví dụ	47
3.2 MÔ HÌNH USE CASE VÀ KỊCH BẢN	48
3.2.1 Vai trò của mô hình use case	48
3.2.2 Xây dựng biểu đồ use case	50
3.2.3 Xây dựng biểu đồ use case trong Rational Rose	57
3.3 MÔ HÌNH LỚP	63
3.3.1 Vấn đề xác định lớp	63
3.3.2 Xây dựng biểu đồ lớp trong pha phân tích	65
3.3.3 Biểu diễn biểu đồ lớp trong Rational Rose	67
3.4 MÔ HÌNH ĐỘNG DỰA TRÊN BIỂU ĐỒ TRẠNG THÁI	71
3.4.1 Khái quát về mô hình động	71

3.4.3 Xây dựng biểu đồ trạng thái	74
3.4.3 Biểu diễn biểu đồ trạng thái trong Rational Rose	75
TỔNG KẾT CHƯƠNG 3	78
CÂU HỎI – BÀI TẬP	79
CHƯƠNG 4: PHA THIẾT KẾ HƯỚNG ĐỐI TƯỢNG	83
4.1 TỔNG QUAN VỀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG	83
4.1.1 Vai trò của pha thiết kế	83
4.1.2 Các bước thiết kế hướng đối tượng	84
4.2 CÁC BIỂU ĐỒ TƯƠNG TÁC	84
4.2.2 Xây dựng biểu đồ tuần tự	84
4.2.3 Xây dựng biểu đồ cộng tác	88
4.2.4 Biểu diễn các biểu đồ tương tác trong Rational Rose	89
4.3 BIỂU ĐỒ LỚP CHI TIẾT	91
4.3.1 Xác định các phương thức cho mỗi lớp	91
4.3.2 Xác định mối quan hệ giữa các lớp	92
4.3.4 Hoàn chỉnh biểu đồ lớp chi tiết	93
4.3 THIẾT KẾ CHI TIẾT	95
4.3.1 Xây dựng biểu đồ hoạt động cho các phương thức	96
4.3.2 Xây dựng bảng thiết kế chi tiết	98
4.4 BIỂU ĐỒ THÀNH PHẦN VÀ BIỂU ĐỒ TRIỂN KHAI	99
4.4.1 Xây dựng biểu đồ thành phần	99
4.4.2 Xây dựng biểu đồ triển khai	100
4.4.3 Biểu diễn biểu đồ thành phần và triển khai trong Rational Rose	102
TỔNG KẾT CHƯƠNG 4	104
CÂU HỎI – BÀI TẬP	104
PHỤ LỤC PHÂN TÍCH THIẾT KẾ HỆ THỐNG THƯ VIỆN ĐIỆN TỬ	108
1. GIỚI THIỆU HỆ THỐNG	108
1.1 Hoạt động nghiệp vụ thư viện	108
1.2 Yêu cầu hệ thống	109
2 PHA PHÂN TÍCH	110
2.1 Xây dựng biểu đồ use case	110
2.2 Xây dựng biểu đồ lớp phân tích	113
2.3 Biểu đồ trạng thái	113
3. PHA THIẾT KẾ	114
3.1 Các biểu đồ tuần tự	115
3.2 Biểu đồ lớp chi tiết	121
3.3 Thiết kế riêng từng chức năng	122
3.4 Biểu đồ hoạt động	126
3.5 Biểu đồ triển khai hệ thống	127
GỢI Ý TRẢ LỜI CÁC BÀI TẬP	129
TÀI LIỆU THAM KHẢO	133

LỜI NÓI ĐẦU

Phương pháp luận phát triển các hệ thống thông tin luôn là một trong những chủ đề quan trọng nhất của công nghệ thông tin. Trải qua một giai đoạn tiến hoá lâu dài, phát triển theo cách tiếp cận hướng đối tượng đã dần dần chiếm ưu thế và ngày càng trở nên phổ biến và đã được chuẩn hoá trong công nghiệp phần mềm.

Cùng với sự ra đời của ngôn ngữ mô hình hoá thống nhất UML và nhiều công cụ hỗ trợ như Rational Rose, AgroUML... phương pháp luận phát triển phần mềm hướng đối tượng đã được áp dụng rộng rãi trong công nghiệp phần mềm trên khắp thế giới. Ngôn ngữ UML hiện thời vẫn đang được phát triển để đáp ứng cho nhiều yêu cầu và nhiều dạng hệ thống khác nhau như hệ phân tán, hệ nhúng...

Tài liệu này nhằm giới thiệu cho sinh viên các khái niệm cơ bản của hướng đối tượng và UML, sau đó trình bày các bước phân tích thiết kế hệ thống thông tin dựa trên UML và công cụ Rational Rose. Nội dung của tài liệu gồm 4 chương và phần Phụ lục:

Chương 1: Mở đầu. Giới thiệu các dạng hệ thống thông tin và các khái niệm cơ bản của cách tiếp cận hướng đối tượng; vòng đời phát triển hệ thống và so sánh các cách tiếp cận phát triển hệ thống.

Chương 2: UML và Công cụ phát triển hệ thống. Trình bày các khái niệm cơ bản của UML, các biểu đồ, các ký hiệu UML và các bước phát triển hệ thống sử dụng các biểu đồ đó. Chương này cũng giới thiệu công cụ Rational Rose cho phân tích thiết kế hệ thống thông tin.

Chương 3: Pha phân tích hướng đối tượng. Trình bày các bước phân tích hệ thống theo các biểu đồ UML bao gồm: xây dựng mô hình use case, xây dựng mô hình lớp và biểu đồ trạng thái. Tài liệu cũng đưa ra những gợi ý cho từng bước và hướng dẫn sử dụng công cụ Rational Rose cho các bước đó.

Chương 4: Pha thiết kế hướng đối tượng. Trình bày các bước thiết kế hệ thống bao gồm: xây dựng các biểu đồ tương tác, biểu đồ lớp chi tiết, thiết kế chi tiết và xây dựng biểu đồ triển khai hệ thống. Tài liệu cũng có những gợi ý cho từng bước của pha thiết kế.

Phần Phụ lục. Trình bày toàn bộ quá trình phân tích thiết kế hệ thống quản lý thư viện và phát sinh mã cho hệ thống này.

Mỗi chương đều có phần câu hỏi, bài tập để giúp sinh viên hiểu rõ hơn kiến thức được học và kiểm tra khả năng áp dụng kiến thức của sinh viên vào các bài toán thực tế.

Tài liệu này được xây dựng nhằm đáp ứng nhu cầu học tập của sinh viên từ xa của Học viện Công nghệ Bưu chính Viễn thông. Do thời gian có hạn nên phiên bản đầu tiên này chắc chắn còn nhiều hạn chế và thiếu sót. Các tác giả rất mong nhận được những đóng góp ý kiến của các đồng nghiệp và các bạn sinh viên.

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Km10 Đường Nguyễn Trãi, Hà Nội
Tel: (04) 52 79 21 21 - Fax: (04) 52 79 55 67
Website: <http://www.e-ptit.edu.vn>; E-mail: info@e-ptit.edu.vn

[cuuduongthancong . com](http://cuuduongthancong.com)

CHƯƠNG TRÌNH
PTIT
ĐÀO TẠO ĐẠI HỌC TỪ XA
[cuuduongthancong . com](http://cuuduongthancong.com)

CHƯƠNG 1

MỞ ĐẦU

Chương này tập trung trình bày các nội dung sau đây:

- Các hệ thống thông tin và vấn đề phát triển hệ thống thông tin
- Khái quát vòng đời phát triển hệ thống thông tin
- Các cách tiếp cận phân tích và thiết kế hệ thống
- Các khái niệm cơ bản của hướng đối tượng

1.1 CÁC HỆ THỐNG THÔNG TIN

Ngày nay, hệ thống thông tin đã được ứng dụng trong mọi lĩnh vực khác nhau của đời sống xã hội. Tùy theo quan điểm mà có thể phân loại các hệ thống thông tin theo các tiêu chí khác nhau. Xét về mặt ứng dụng, hệ thống thông tin có thể được phân chia thành một số dạng như sau:

Hệ thống thông tin quản lý: Bao gồm các hệ thống thông tin hỗ trợ các hoạt động nghiệp vụ và quản lý của các doanh nghiệp, các tổ chức. Ví dụ các hệ thống quản lý nhân sự, hệ thống kế toán, hệ thống tính cước và chăm sóc khách hàng, hệ thống quản lý thư viện, hệ thống đào tạo trực tuyến ...

Các hệ thống Website: là các hệ thống có nhiệm vụ cung cấp thông tin cho người dùng trên môi trường mạng Internet. Các hệ thống Website có đặc điểm là thông tin cung cấp cho người dùng có tính đa dạng (có thể là tin tức hoặc các dạng file đa phương tiện) và được cập nhật thường xuyên.

Hệ thống thương mại điện tử: Là các hệ thống website đặc biệt phục vụ việc trao đổi mua bán hàng hoá, dịch vụ trên môi trường Internet. Hệ thống thương mại điện tử bao gồm cả các nền tảng hỗ trợ các giao thức mua bán, các hình thức thanh toán, chuyển giao hàng hoá ...

Hệ thống điều khiển: là các hệ thống phần mềm gắn với các thiết bị phần cứng hoặc các hệ thống khác nhằm mục đích điều khiển và giám sát hoạt động của thiết bị hay hệ thống đó.

Mỗi loại hệ thống thông tin có những đặc trưng riêng và cũng đặt ra những yêu cầu riêng cho việc phát triển hệ thống. Ví dụ, các hệ thống điều khiển đòi hỏi những yêu cầu về môi trường phát triển, hệ điều hành và ngôn ngữ lập trình riêng;

các hệ website thực thi các chức năng trên môi trường mạng phân tán đòi hỏi cách phát triển riêng...Do vậy, không có một phương pháp luận chung cho tất cả các dạng hệ thống thông tin.

Phạm vi của tài liệu này nhằm giới thiệu một số khái niệm cơ bản của UML cho phát triển các hệ thống và để dễ dàng minh họa chúng ta sẽ xem xét vấn đề phát triển dạng hệ thống thông tin phổ biến nhất là hệ thống thông tin quản lý.

1.2 KHÁI QUÁT VÒNG ĐỜI PHÁT TRIỂN HỆ THỐNG THÔNG TIN

Việc phát triển các hệ thống thông tin không chỉ đơn giản là lập trình mà luôn được xem như một tiến trình hoàn chỉnh.

Tiến trình phần mềm là phương cách sản xuất ra phần mềm với các thành phần chủ yếu bao gồm: mô hình vòng đời phát triển phần mềm, các công cụ hỗ trợ cho phát triển phần mềm và những người trong nhóm phát triển phần mềm.

Như vậy, tiến trình phát triển phần mềm nói chung là sự kết hợp cả hai khía cạnh kỹ thuật (vòng đời phát triển, phương pháp phát triển, các công cụ và ngôn ngữ sử dụng, ...) và khía cạnh quản lý (quản lý dự án phần mềm).

Mô hình vòng đời phần mềm là các bước phát triển một sản phẩm phần mềm cụ thể. Một vòng đời phát triển phần mềm thường có các pha cơ bản sau:

Pha xác định yêu cầu: khám phá các khái niệm liên quan đến việc phát triển phần mềm, xác định chính xác yêu cầu và các ràng buộc của khách hàng với sản phẩm phần mềm đó.

Pha phân tích: mô tả chức năng của sản phẩm, các input của sản phẩm và các output được yêu cầu; khám phá các khái niệm trong miền quan tâm của sản phẩm và bước đầu đưa ra giải pháp xây dựng hệ thống.

Pha thiết kế: xác định cụ thể phần mềm sẽ được xây dựng như thế nào. Pha thiết kế bao gồm hai mức là thiết kế kiến trúc và thiết kế chi tiết.

Pha cài đặt tích hợp: cài đặt chi tiết và tích hợp hệ thống phần mềm dựa trên kết quả của pha thiết kế.

Pha bảo trì: tiến hành sửa chữa phần mềm khi có các thay đổi. Đây là pha rất quan trọng, tiêu tốn nhiều thời gian và chi phí nhất trong tiến trình phát triển phần mềm.

Pha loại bỏ: thực hiện loại bỏ phần mềm hoặc thay thế phần mềm bởi một phần mềm hoàn toàn mới.

Thông thường hai quá trình không thể thiếu được trong vòng đời phát triển phần mềm là *viết tài liệu và kiểm thử*. Các quá trình này không trở thành một pha riêng biệt mà được tiến hành song song với tất cả các pha khác trong tiến trình phần mềm nghĩa là tất cả các pha đều phải viết tài liệu và kiểm thử với các mức độ khác nhau.

Có rất nhiều mô hình vòng đời phần mềm nhưng hai mô hình đơn giản và được sử dụng rộng rãi nhất là mô hình thác nước và mô hình làm bản mẫu nhanh.

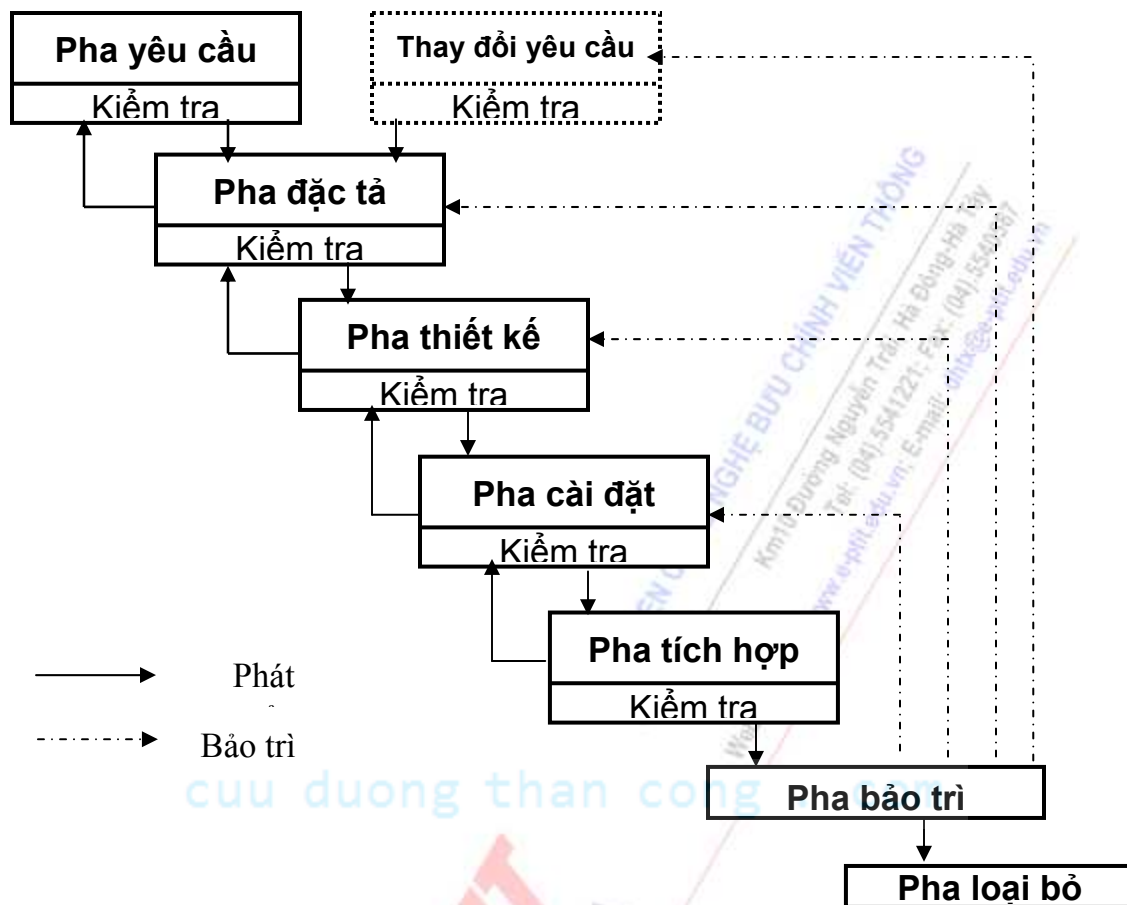
1.2.1 Mô hình thác nước

Theo mô hình thác nước, sau khi yêu cầu của hệ thống đã được xác định và kiểm tra bởi nhóm SQA, pha phân tích sẽ được tiến hành để xây dựng tài liệu. Sau khi tài liệu phân tích được khách hàng chấp nhận, nhóm phát triển sẽ tiến hành lập kế hoạch và lịch biểu cho các quá trình phát triển tiếp theo. Sau đó, các pha thiết kế, cài đặt và tích hợp sẽ lần lượt được tiến hành; mỗi pha này đều có phần kiểm tra để khi cần có thể quay lại sửa đổi tài liệu của pha trước đó. Khi phần mềm đã được triển khai và chuyển sang pha bảo trì; nếu có lỗi hoặc thay đổi xảy ra, nhóm thiết kế sẽ phải quay trở lại sửa đổi tài liệu cho một trong các pha trước đó và nếu cần có thể quay trở lại thay đổi một số yêu cầu ban đầu của hệ thống.

Vì các pha cứ nối tiếp nhau một cách liên tục như một thác nước nên mô hình này được gọi là mô hình thác nước. Tiến trình phần mềm theo mô hình thác nước được biểu diễn như trong Hình 1.1. Mô hình thác nước có một số ưu điểm như sau:

- Có vòng lặp, cho phép trở về pha trước trong vòng đời phần mềm để sửa chữa khi phát hiện lỗi hoặc khi có thay đổi.
- Hướng tài liệu: tất cả các pha trong vòng đời phần mềm theo mô hình thác nước đều được viết tài liệu cẩn thận và được kiểm tra bởi nhóm SQA trước khi chuyển sang pha tiếp theo. Do vậy, hệ thống sẽ dễ dàng bảo trì khi có những thay đổi.

Tuy nhiên, mô hình thác nước cũng có nhược điểm là sản phẩm phần mềm cuối cùng có thể không thỏa mãn nhu cầu thực sự của khách hàng. Lý do là khách hàng chỉ được trao đổi một lần duy nhất và chưa được hình dung sản phẩm nên rất có thể các pha tiếp theo sẽ không thực hiện đúng những gì khách hàng cần.



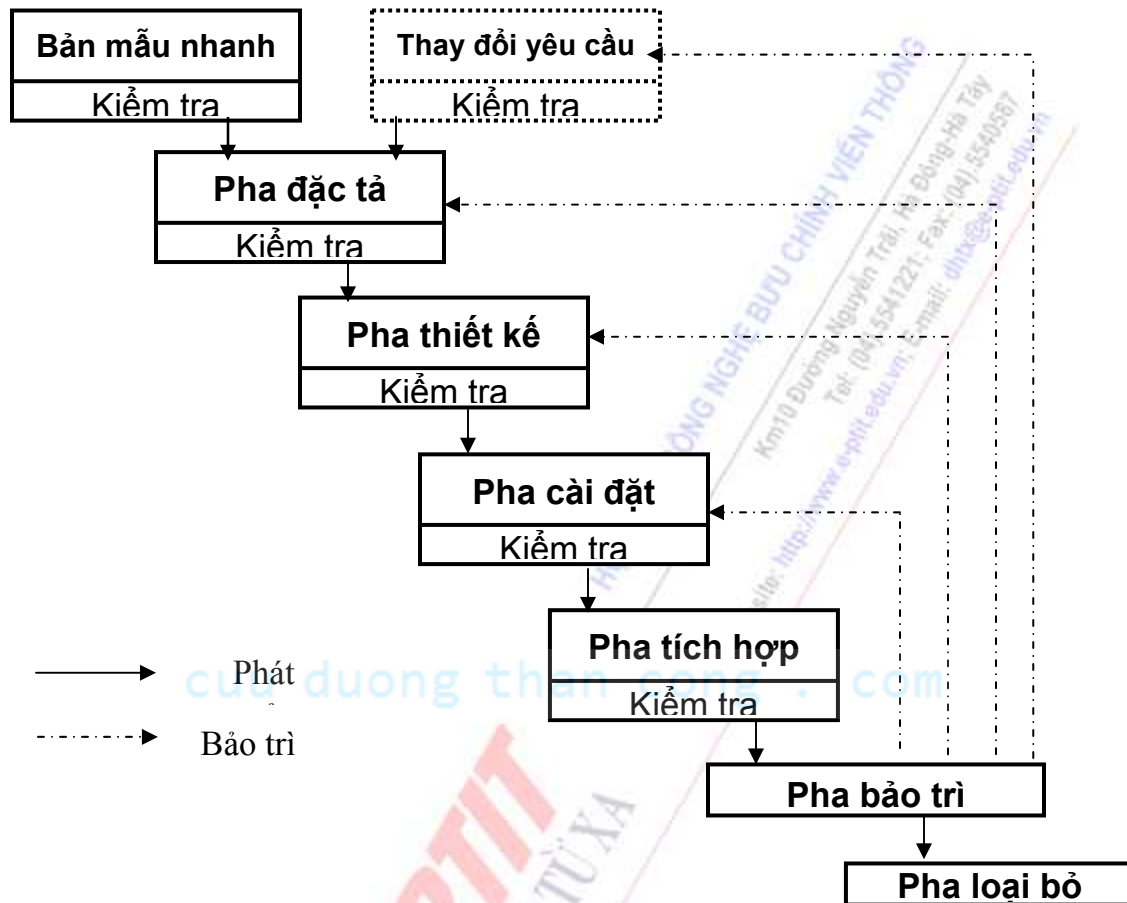
Hình 1.1: Tiến trình phần mềm theo mô hình thác nước

1.2.2 Mô hình làm bản mẫu nhanh

Trong mô hình làm bản mẫu nhanh, bước đầu tiên là nhóm phát triển sẽ xây dựng một bản mẫu và giao cho khách hàng và người sử dụng hệ thống dùng thử. Khi khách hàng đồng ý chấp nhận bản mẫu thì nhóm phát triển mới tiếp tục tiến hành các pha khác của vòng đời phần mềm. Trong các pha tiếp theo, do đã có bản mẫu nên các pha sẽ được tiến hành liên tục và không có bước quay về pha trước đó. Chỉ khi hệ thống đã triển khai và chuyển sang pha bảo trì, nếu có thay đổi hay phát hiện lỗi thì nhóm phát triển mới quay lại một trong những pha trước đó, nhưng không quay lại pha làm bản mẫu vì bản mẫu đã được chấp nhận.

Ưu điểm chính của mô hình này là “nhanh” và hơn nữa do sản phẩm phần mềm được tạo ra từ mô hình làm bản mẫu nên có khả năng cao là đảm bảo thỏa mãn yêu cầu thực sự của khách hàng. Tuy nhiên, mô hình làm bản mẫu nhanh

cũng có nhược điểm do các pha được tiến hành liên tục mà không được viết tài liệu. Mô hình làm bản mẫu nhanh được biểu diễn như trong Hình 1.2.



Hình 1.2: Vòng đời phát triển phần mềm theo mô hình làm bản mẫu nhanh

1.3 CÁC CÁCH TIẾP CẬN PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Trong những năm 70 - 80, phương pháp hướng cấu trúc được coi là phương pháp chuẩn để phát triển phần mềm. Tuy nhiên, phương pháp này tỏ ra không phù hợp trong phát triển các hệ phần mềm lớn và đặc biệt là kém hiệu quả trong sử dụng lại - một yêu cầu quan trọng trong công nghiệp phần mềm. Thập niên 90 chứng kiến sự nở rộ trong nghiên cứu và xây dựng phương pháp luận phát triển phần mềm hướng đối tượng và nhanh chóng trở thành phổ biến trong công nghiệp phần mềm ngày nay. Để hiểu rõ phần nào sự khác biệt này phần này dành so sánh một số khác biệt giữa hai phương pháp này.

1.3.1 Phương pháp hướng cấu trúc

Đặc trưng của phương pháp hướng cấu trúc là phân chia chương trình chính thành nhiều chương trình con, mỗi chương trình con nhằm đến thực hiện một công việc xác định.

Trong phương pháp hướng cấu trúc, phần mềm được thiết kế dựa trên một trong hai hướng : hướng dữ liệu và hướng hành động.

- Cách tiếp cận hướng dữ liệu xây dựng phần mềm dựa trên việc phân rã phần mềm theo các chức năng cần đáp ứng và dữ liệu cho các chức năng đó. Cách tiếp cận hướng dữ liệu sẽ giúp cho những người phát triển hệ thống dễ dàng xây dựng ngân hàng dữ liệu.
- Cách tiếp cận hướng hành động lại tập trung phân tích hệ phần mềm dựa trên các hoạt động thực thi các chức năng của phần mềm đó.

Cách thức thực hiện của phương pháp hướng cấu trúc là **phương pháp thiết kế từ trên xuống (top-down)**. Phương pháp này tiến hành phân rã bài toán thành các bài toán nhỏ hơn, rồi tiếp tục phân rã các bài toán con cho đến khi nhận được các bài toán có thể cài đặt được ngay sử dụng các hàm của ngôn ngữ lập trình hướng cấu trúc.

Phương pháp hướng cấu trúc có ưu điểm là tư duy phân tích thiết kế rõ ràng, chương trình sáng sủa dễ hiểu. Tuy nhiên, phương pháp này có một số nhược điểm sau:

- Không hỗ trợ việc sử dụng lại. Các chương trình hướng cấu trúc phụ thuộc chặt chẽ vào cấu trúc dữ liệu và bài toán cụ thể, do đó không thể dùng lại một modul nào đó trong phần mềm này cho phần mềm mới với các yêu cầu về dữ liệu khác.
- Không phù hợp cho phát triển các phần mềm lớn. Nếu hệ thống thông tin lớn, việc phân ra thành các bài toán con cũng như phân các bài toán con thành các modul và quản lý mối quan hệ giữa các modul đó sẽ là không phải là dễ dàng và dễ gây ra các lỗi trong phân tích và thiết kế hệ thống, cũng như khó kiểm thử và bảo trì.

1.3.2 Phương pháp hướng đối tượng

Khác với phương pháp hướng cấu trúc chỉ tập trung hoặc vào dữ liệu hoặc vào hành động, phương pháp hướng đối tượng tập trung vào cả hai khía cạnh của hệ thống là dữ liệu và hành động.

Cách tiếp cận hướng đối tượng là một lối tư duy theo cách ánh xạ các thành phần trong bài toán vào các đối tượng ngoài đời thực. Với cách tiếp cận này, một hệ thống được chia tương ứng thành các thành phần nhỏ gọi là các *đối tượng*, mỗi đối tượng bao gồm đầy đủ cả dữ liệu và hành động liên quan đến đối tượng đó. Các đối tượng trong một hệ thống tương đối độc lập với nhau và phần mềm sẽ được xây dựng bằng cách kết hợp các đối tượng đó lại với nhau thông qua các mối quan hệ và tương tác giữa chúng. Các nguyên tắc cơ bản của phương pháp hướng đối tượng bao gồm :

- **Trừu tượng hóa (abstraction):** trong phương pháp hướng đối tượng, các thực thể phần mềm được mô hình hóa dưới dạng các đối tượng. Các đối tượng này được trừu tượng hóa ở mức cao hơn dựa trên thuộc tính và phương thức mô tả đối tượng để tạo thành các lớp. Các lớp cũng sẽ được trừu tượng hóa ở mức cao hơn nữa để tạo thành một sơ đồ các lớp được kế thừa lẫn nhau. Trong phương pháp hướng đối tượng có thể tồn tại những lớp không có đối tượng tương ứng, gọi là *lớp trừu tượng*. Như vậy, nguyên tắc cơ bản để xây dựng các khái niệm trong hướng đối tượng là sự trừu tượng hóa theo các mức độ khác nhau.
- **Tính đóng gói (encapsulation) và ẩn giấu thông tin:** các đối tượng có thể có những phương thức hoặc thuộc tính riêng (*kiểu private*) mà các đối tượng khác không thể sử dụng được. Dựa trên nguyên tắc ẩn giấu thông tin này, cài đặt của các đối tượng sẽ hoàn toàn độc lập với các đối tượng khác, các lớp độc lập với nhau và cao hơn nữa là cài đặt của hệ thống hoàn toàn độc lập với người sử dụng cũng như các hệ thống khác sử dụng kết quả của nó.
- **Tính modul hóa (modularity):** các bài toán sẽ được phân chia thành những vấn đề nhỏ hơn, đơn giản và quản lý được.
- **Tính phân cấp (hierarchy):** cấu trúc chung của một hệ thống hướng đối tượng là dạng phân cấp theo các mức độ trừu tượng từ cao đến thấp.

Ưu điểm nổi bật của phương pháp hướng đối tượng là đã giải quyết được các vấn đề nảy sinh với phương pháp hướng cấu trúc:

- *Hỗ trợ sử dụng lại mã nguồn* : Chương trình lập trình theo phương pháp hướng đối tượng thường được chia thành các gói là các nhóm của các lớp đối tượng khác nhau. Các gói này hoạt động tương đối độc lập và hoàn toàn có thể sử dụng lại trong các hệ thống thông tin tương tự.
- *Phù hợp với các hệ thống lớn*: Phương pháp hướng đối tượng không chia bài toán thành các bài toán nhỏ mà tập trung vào việc xác định các đối tượng, dữ liệu và hành động gắn với đối tượng và mối quan hệ giữa các đối tượng. Các đối tượng hoạt động độc lập và chỉ thực hiện hành động khi nhận được yêu cầu từ các đối tượng khác. Vì vậy, phương pháp này hỗ trợ phân tích, thiết kế và quản lý một hệ thống lớn, có thể mô tả các hoạt động nghiệp vụ phức tạp bởi quá trình phân tích thiết kế không phụ thuộc vào số biến dữ liệu hay số lượng thao tác cần thực hiện mà chỉ quan tâm đến các đối tượng tồn tại trong hệ thống đó.

1.5 CÁC KHÁI NIỆM CƠ BẢN CỦA HƯỚNG ĐỐI TƯỢNG

Một số khái niệm cơ bản trong hướng đối tượng bao gồm:

- **Đối tượng (object)**: một đối tượng biểu diễn một thực thể vật lý, một thực thể khái niệm hoặc một thực thể phần mềm. Có thể định nghĩa một đối tượng là một khái niệm, sự trừu tượng hoặc một vật với giới hạn rõ ràng và có ý nghĩa với một ứng dụng cụ thể.
- **Lớp (Class)**: là mô tả của một nhóm đối tượng có chung các thuộc tính, hành vi và các mối quan hệ. Như vậy, một đối tượng là thể hiện của một lớp và một lớp là một định nghĩa trừu tượng của đối tượng.
- **Thành phần (component)**: là một phần của hệ thống hoạt động độc lập và giữ một chức năng nhất định trong hệ thống.
- **Gói (package)**: là một cách tổ chức các thành phần, phần tử trong hệ thống thành các nhóm. Nhiều gói có thể được kết hợp với nhau để trở thành một hệ thống con (subsystem).
- **Kế thừa**: Trong phương pháp hướng đối tượng, một lớp có thể có sử dụng lại các thuộc tính và phương thức của một hoặc nhiều lớp khác. Kiểu quan hệ này gọi là quan hệ kế thừa, được xây dựng dựa trên mối quan hệ kế thừa trong bài toán thực tế. Ví dụ, giả sử ta có lớp *Người* gồm các thuộc tính : *tên, ngày sinh, quê quán, giới tính* ; Lớp *Nhân Viên* có quan hệ kế thừa từ lớp *Người* sẽ có tất

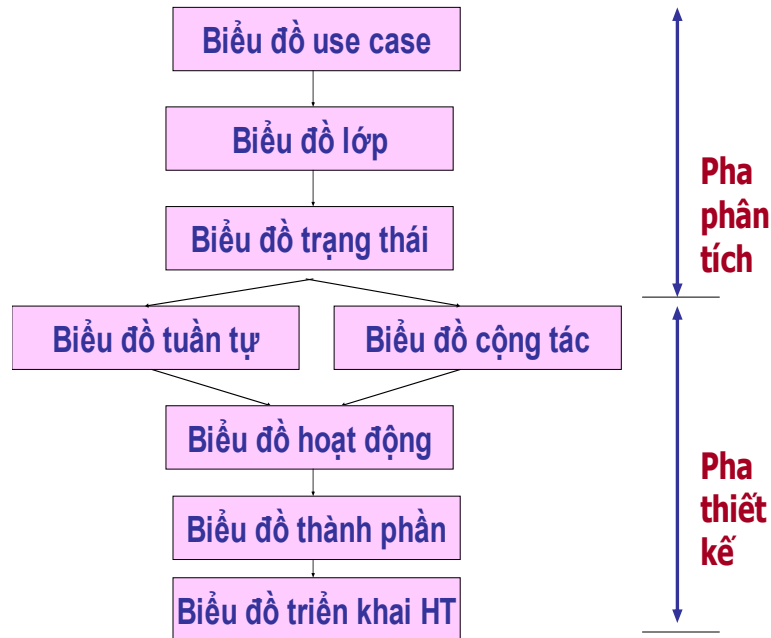
cả các thuộc tính trên và bổ sung thêm các thuộc tính mới gồm : *chức vụ, lương*.

Vòng đời phát triển phần mềm hướng đối tượng cũng có các pha tương tự như các vòng đời phát triển phần mềm nói chung. Các pha cơ bản đặc trưng trong phát triển phần mềm hướng đối tượng bao gồm:

- **Phân tích hướng đối tượng:** xây dựng một mô hình chính xác để mô tả hệ thống cần xây dựng là gì. Thành phần của mô hình này là các đối tượng gắn với hệ thống thực.
- **Thiết kế hướng đối tượng:** Là giai đoạn tổ chức chương trình thành các tập hợp đối tượng cộng tác, mỗi đối tượng trong đó là thực thể của một lớp. Kết quả của pha thiết kế cho biết hệ thống sẽ được xây dựng như thế nào qua các bản thiết kế kiến trúc và thiết kế chi tiết.
- **Lập trình và tích hợp:** Thực hiện bản thiết kế hướng đối tượng bằng cách sử dụng các ngôn ngữ lập trình hướng đối tượng (C++, Java, ...).

1.6 CÁC BƯỚC PHÂN TÍCH THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Các bước phân tích thiết kế hướng đối tượng được xây dựng dựa trên biểu đồ các ký hiệu UML. Đó là ngôn ngữ mô hình hoá thống nhất được xây dựng để mô hình hoá quá trình phát triển hệ thống phần mềm hướng đối tượng. Các vấn đề cơ bản về UML sẽ được giới thiệu chi tiết trong Chương 2. Phần này chỉ nhằm giới thiệu một cách khái quát các bước trong phân tích và thiết kế hướng đối tượng.



Hình 1.3: Các bước phát triển hệ thống hướng đối tượng

Pha phân tích

Xây dựng Biểu đồ use case: Dựa trên tập yêu cầu ban đầu, người phân tích tiến hành xác định các tác nhân, use case và các quan hệ giữa các use case để mô tả lại các chức năng của hệ thống. Một thành phần quan trọng trong biểu đồ use case là các kịch bản mô tả hoạt động của hệ thống trong mỗi use case cụ thể.

Xây dựng Biểu đồ lớp: Xác định tên các lớp, các thuộc tính của lớp, một số phương thức và mối quan hệ cơ bản trong sơ đồ lớp.

Xây dựng biểu đồ trạng thái: Mô tả các trạng thái và chuyển tiếp trạng thái trong hoạt động của một đối tượng thuộc một lớp nào đó.

Trong Pha thiết kế

Xây dựng các biểu đồ tương tác (gồm biểu đồ cộng tác và biểu đồ tuần tự): mô tả chi tiết hoạt động của các use case dựa trên các scenario đã có và các lớp đã xác định trong pha phân tích.

Xây dựng biểu đồ lớp chi tiết: tiếp tục hoàn thiện biểu đồ lớp bao gồm bổ sung các lớp còn thiếu, dựa trên biểu đồ trạng thái để bổ sung các thuộc tính, dựa trên biểu đồ tương tác để xác định các phương thức và mối quan hệ giữa các lớp.

Xây dựng biểu đồ hoạt động: mô tả hoạt động của các phương thức phức tạp trong mỗi lớp hoặc các hoạt động hệ thống có sự liên quan của nhiều lớp. Biểu đồ hoạt động là cơ sở để cài đặt các phương thức trong các lớp.

Xây dựng biểu đồ thành phần: xác định các gói, các thành phần và tổ chức phần mềm theo các thành phần đó.

Xây dựng biểu đồ triển khai hệ thống: xác định các thành phần và các thiết bị cần thiết để triển khai hệ thống, các giao thức và dịch vụ hỗ trợ.

TỔNG KẾT CHƯƠNG 1

Chương này đã trình bày các nội dung mở đầu cho phân tích thiết kế hệ thống hướng đối tượng. Các nội dung cơ bản cần nhớ gồm :

- Có nhiều loại hệ thống thông tin khác nhau như : hệ thống thông tin quản lý, các Website, các hệ thống thương mại, các hệ thống điều khiển ... Mỗi loại hệ thống thông tin sẽ tương ứng với một phương pháp phát triển riêng.
- Việc phát triển các hệ thống thông tin nói chung được xem như một vòng đời với các pha : Xác định yêu cầu, đặc tả, thiết kế, cài đặt tích hợp, bảo trì và loại bỏ. Có hai mô hình vòng đời đơn giản và hay dùng nhất là mô hình thác nước và mô hình làm bản mẫu nhanh.
- Phương pháp phát triển phần mềm hướng đối tượng tỏ ra có nhiều ưu điểm hơn so với phương pháp hướng cấu trúc. Các pha đặc trưng trong vòng đời phát triển phần mềm hướng đối tượng là phân tích hướng đối tượng, thiết kế hướng đối tượng và lập trình hướng đối tượng.
- Các bước phát triển phần mềm hướng đối tượng được xây dựng dựa trên các biểu đồ trong ngôn ngữ mô hình hoá thống nhất UML. Chương 2 sẽ trình bày chi tiết về UML và tập ký hiệu cho các bước phát triển hệ thống.

CÂU HỎI VÀ BÀI TẬP

1. Kể tên một số ví dụ cho các loại hệ thống thông tin: hệ thống thông tin quản lý, hệ thống website thương mại điện tử, hệ thống điều khiển ...
2. Vì sao nói tiến trình phần mềm là sự kết hợp khía cạnh kỹ thuật và khía cạnh quản lý.
3. So sánh ưu, nhược điểm của phương pháp phát triển phần mềm hướng cấu trúc và hướng đối tượng.

4. Trình bày các khái niệm trong hướng đối tượng : lớp, đối tượng, gói, thành phần, kế thừa. Cho ví dụ.

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Km10 Đường Nguyễn Trãi, Hà Đông-Hà Tây
Tel: (04) 5541221; Fax: (04) 5540567
Website: <http://www.e-ptit.edu.vn>; E-mail: dhtr@e-ptit.edu.vn

cuuduongthancong.com

CHƯƠNG TRÌNH
PTIT
ĐẠO TẠO ĐẠI HỌC TỪ XA

cuuduongthancong.com

CHƯƠNG 2

UML VÀ CÔNG CỤ PHÁT TRIỂN HỆ THỐNG

Chương này nhằm giới thiệu về ngôn ngữ mô hình hoá thống nhất UML và công cụ phát triển phần mềm hướng đối tượng. Nội dung cụ thể bao gồm:

- Giới thiệu UML
- Các biểu đồ trong UML
- Các bước phân tích thiết kế hướng đối tượng sử dụng UML
- Giới thiệu bộ công cụ Rational Rose

2.1 GIỚI THIỆU VỀ UML

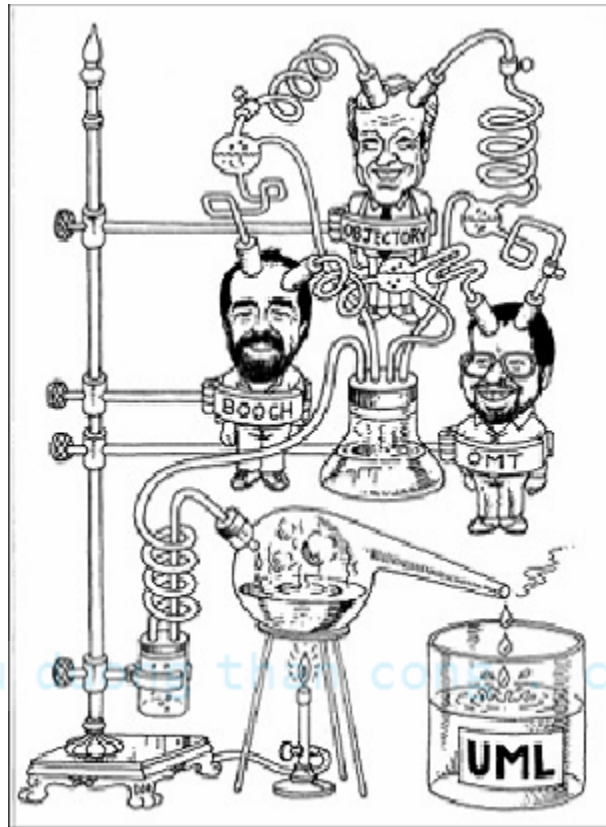
2.1.1 Lịch sử ra đời của UML

Việc áp dụng rộng rãi phương pháp hướng đối tượng đã đặt ra yêu cầu cần phải xây dựng một phương pháp mô hình hóa để có thể sử dụng như một chuẩn chung cho những người phát triển phần mềm hướng đối tượng trên khắp thế giới. Trong khi các ngôn ngữ hướng đối tượng ra đời khá sớm, ví dụ như Simula-67 (năm 1967), Smalltalk (đầu những năm 1980), C++, CLOS (giữa những năm 1980)...thì những phương pháp luận cho phát triển hướng đối tượng lại ra đời khá muộn. Cuối những năm 80, đầu những năm 1990, một loạt các phương pháp luận và ngôn ngữ mô hình hóa hướng đối tượng mới ra đời, như Booch của Grady Booch, OMT của James Rumbaugh, OOSE của Ivar Jacobson, hay OOA and OOD của Coad và Yordon.

Mỗi phương pháp luận và ngôn ngữ trên đều có hệ thống ký hiệu riêng, phương pháp xử lý riêng và công cụ hỗ trợ riêng. Chính điều này đã thúc đẩy những người tiên phong trong lĩnh vực mô hình hoá hướng đối tượng ngồi lại cùng nhau để tích hợp những điểm mạnh của mỗi phương pháp và đưa ra một mô hình thống nhất chung. Nỗ lực thống nhất đầu tiên bắt đầu khi Rumbaugh gia nhập nhóm nghiên cứu của Booch tại tập đoàn Rational năm 1994 và sau đó Jacobson cũng gia nhập nhóm này vào năm 1995.

James Rumbaugh, Grady Booch và Ivar Jacobson đã cùng cố gắng xây dựng được một Ngôn Ngữ Mô Hình Hoá Thống Nhất và đặt tên là UML (Unified Modeling Language).

Modeling Language) (Hình 2.1). UML đầu tiên được đưa ra năm 1997 và sau đó được chuẩn hoá để trở thành phiên bản 1.0. Hiện nay chúng ta đang sử dụng ngôn ngữ UML phiên bản 2.0.



Hình 2.1: Sự ra đời của UML

2.1.2 UML – Ngôn ngữ mô hình hoá hướng đối tượng

UML (Unified Modelling Language) là ngôn ngữ mô hình hoá tổng quát được xây dựng để đặc tả, phát triển và viết tài liệu cho các khía cạnh trong phát triển phần mềm hướng đối tượng. UML giúp người phát triển hiểu rõ và ra quyết định liên quan đến phần mềm cần xây dựng. UML bao gồm một tập các khái niệm, các ký hiệu, các biểu đồ và hướng dẫn.

UML hỗ trợ xây dựng hệ thống hướng đối tượng dựa trên việc nắm bắt khía cạnh cấu trúc tĩnh và các hành vi động của hệ thống.

- Các cấu trúc tĩnh định nghĩa các kiểu đối tượng quan trọng của hệ thống, nhằm cài đặt và chỉ ra mối quan hệ giữa các đối tượng đó.
- Các hành vi động (dynamic behavior) định nghĩa các hoạt động của các đối tượng theo thời gian và tương tác giữa các đối tượng hướng tới đích.

Các mục đích của ngôn ngữ mô hình hoá thống nhất UML:

- Mô hình hoá các hệ thống sử dụng các khái niệm hướng đối tượng.
- Thiết lập sự liên hệ từ nhận thức của con người đến các sự kiện cần mô hình hoá.
- Giải quyết vấn đề về mức độ thừa kế trong các hệ thống phức tạp với nhiều ràng buộc khác nhau.
- Tạo một ngôn ngữ mô hình hoá có thể sử dụng được bởi người và máy.

UML quy định một loạt các ký hiệu và quy tắc để mô hình hoá các pha trong quá trình phát triển phần mềm hướng đối tượng dưới dạng các biểu đồ.

2.1.3 Các khái niệm cơ bản trong UML

a) Khái niệm mô hình

Mô hình là một biểu diễn của sự vật hay một tập các sự vật trong một lĩnh vực áp dụng nào đó theo một cách khác. Mô hình nhằm nắm bắt các khía cạnh quan trọng của sự vật, bỏ qua các khía cạnh không quan trọng và biểu diễn theo một tập ký hiệu và quy tắc nào đó. Các mô hình thường được xây dựng sao cho có thể vẽ được thành các biểu đồ dựa trên tập ký hiệu và quy tắc đã cho.

Khi xây dựng các hệ thống, mô hình được sử dụng nhằm thoả mãn các mục đích sau:

- Nắm bắt chính xác yêu cầu và tri thức miền mà hệ thống cần phát triển
- Thể hiện tư duy về thiết kế hệ thống
- Trợ giúp ra quyết định thiết kế dựa trên việc phân tích yêu cầu
- Tổ chức, tìm kiếm, lọc, kiểm tra và sửa đổi thông tin về các hệ thống lớn.
- Làm chủ được các hệ thống phức tạp

Các thành phần trong một mô hình bao gồm:

- Ngữ nghĩa và biểu diễn: Ngữ nghĩa là nhằm đưa ra ý nghĩa, bản chất và các tính chất của tập các ký hiệu. Biểu diễn là phương pháp thể hiện mô hình theo cách sao cho có thể nhìn thấy được.
- Ngữ cảnh: mô tả tổ chức bên trong, cách sử dụng mô hình trong tiến trình phần mềm ...

b) Các hướng nhìn (View) trong UML

Các mô hình trong UML nhằm mục đích hỗ trợ phát triển các hệ thống phần mềm hướng đối tượng. Trong phương pháp luận hướng đối tượng không có sự phân biệt rạch ròi giữa các pha hay các bước. Tuy nhiên, thông thường UML vẫn được chia thành một số hướng nhìn và nhiều loại biểu đồ.

Một hướng nhìn trong UML là một tập con các biểu đồ UML được xây dựng để biểu diễn một khía cạnh nào đó của hệ thống.

Sự phân biệt giữa các hướng nhìn là rất linh hoạt. Có thể có những biểu đồ UML có mặt trong cả hai hướng nhìn. Các hướng nhìn cùng các biểu đồ tương ứng được mô tả trong bảng sau:

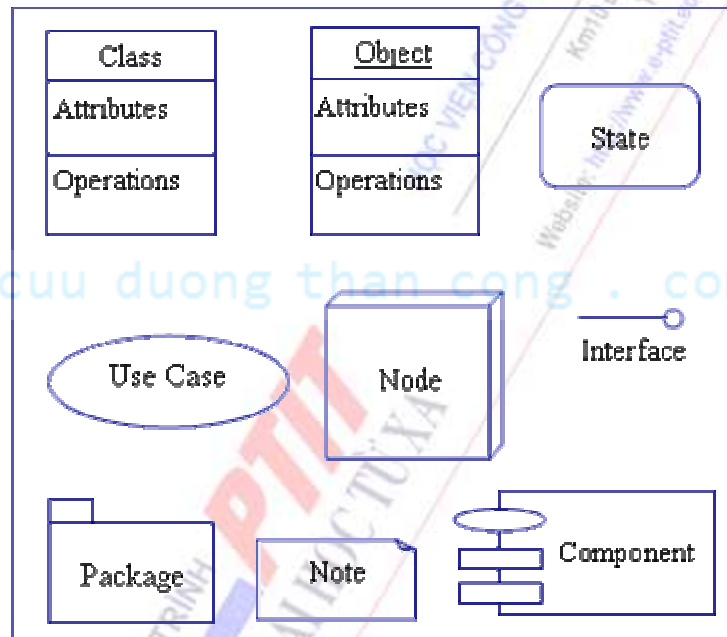
Khía cạnh chính	Hướng nhìn	Các biểu đồ	Các khái niệm chính
<i>Khía cạnh cấu trúc hệ thống</i>	Hướng nhìn tĩnh (static view)	Biểu đồ lớp	lớp, liên hệ, kế thừa, phụ thuộc, giao diện
	Hướng nhìn use case (Use case view)	Biểu đồ use case	Use case, tác nhân, liên hệ, extend, include ...
	Hướng nhìn cài đặt (implementation view)	Biểu đồ thành phần	Thành phần, giao diện, quan hệ phụ thuộc ...
	Hướng nhìn triển khai (deployment view)	Biểu đồ triển khai	Node, thành phần, quan hệ phụ thuộc, vị trí (location)
<i>Khía cạnh động</i>	Hướng nhìn máy trạng thái (state machine view)	Biểu đồ trạng thái	Trạng thái, sự kiện, chuyển tiếp, hành động
	Hướng nhìn hoạt động (activity view)	Biểu đồ động	Trạng thái, sự kiện, chuyển tiếp, kết hợp, đồng bộ ...
	Hướng nhìn tương tác (interaction view)	Biểu đồ tuần tự	Tương tác, đối tượng, thông điệp, kích hoạt ...
		Biểu đồ cộng tác	Cộng tác, vai trò cộng tác, thông điệp ...
<i>Khía cạnh quản lý mô hình</i>	Hướng nhìn quản lý mô hình	Biểu đồ lớp	Gói, hệ thống con, mô hình

<i>Khía cạnh khả năng mở rộng</i>	Tất cả	Tất cả	Các ràng buộc, stereotype, ...
-----------------------------------	--------	--------	--------------------------------

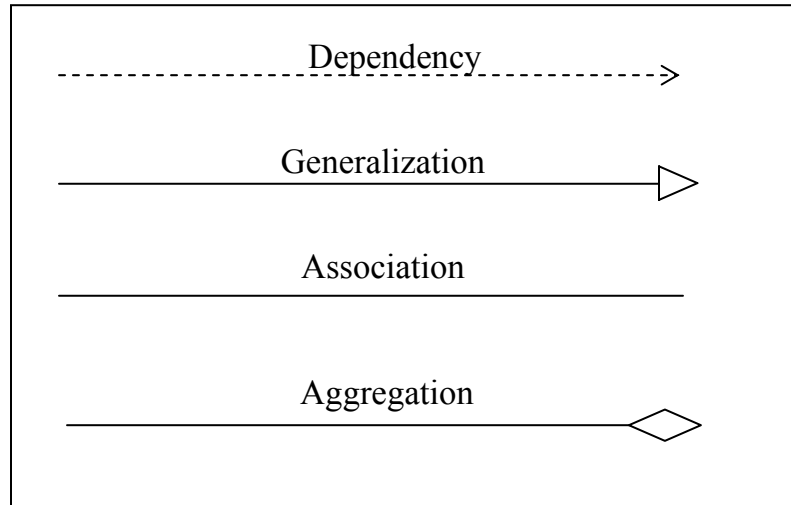
Bảng 2.1: Các hướng nhìn trong UML

c) Các phần tử mô hình và các quan hệ

Một số ký hiệu để mô hình hướng đối tượng thường gặp trong UML được biểu diễn trong Hình 2.2. Đi kèm với các phần tử mô hình này là các *quan hệ*. Các quan hệ này có thể xuất hiện trong bất cứ mô hình nào của UML dưới các dạng khác nhau (như quan hệ giữa các use case, quan hệ trong biểu đồ lớp ...) (Hình 2.3).



Hình 2.2: Một số phần tử mô hình thường gặp trong UML



Hình 2.3: Một số dạng quan hệ trong UML

Ý nghĩa của các phần tử mô hình và các quan hệ sẽ được giải thích cụ thể hơn trong các chương sau.

2.2 CÁC BIỂU ĐỒ UML

Thành phần mô hình chính trong UML là các biểu đồ:

- **Biểu đồ use case** biểu diễn sơ đồ chức năng của hệ thống. Từ tập yêu cầu của hệ thống, biểu đồ use case sẽ phải chỉ ra hệ thống cần thực hiện điều gì để thoả mãn các yêu cầu của người dùng hệ thống đó. Đi kèm với biểu đồ use case là các kịch bản.
- **Biểu đồ lớp** chỉ ra các lớp đối tượng trong hệ thống, các thuộc tính và phương thức của từng lớp và các mối quan hệ giữa những lớp đó.
- **Biểu đồ trạng thái** tương ứng với mỗi lớp sẽ chỉ ra các trạng thái mà đối tượng của lớp đó có thể có và sự chuyển tiếp giữa những trạng thái đó.
- **Các biểu đồ tương tác** biểu diễn mối liên hệ giữa các đối tượng trong hệ thống và giữa các đối tượng với các tác nhân bên ngoài. Có hai loại biểu đồ tương tác:
 - **Biểu đồ tuần tự:** Biểu diễn mối quan hệ giữa các đối tượng và giữa các đối tượng và tác nhân theo thứ tự thời gian.
 - **Biểu đồ cộng tác:** Biểu diễn mối quan hệ giữa các đối tượng và giữa các đối tượng và tác nhân nhưng nhấn mạnh đến vai trò của các đối tượng trong tương tác.

- **Biểu đồ hoạt động** biểu diễn các hoạt động và sự đồng bộ, chuyển tiếp các hoạt động, thường được sử dụng để biểu diễn các phương thức phức tạp của các lớp.
- **Biểu đồ thành phần** định nghĩa các thành phần của hệ thống và mối liên hệ giữa các thành phần đó.
- **Biểu đồ triển khai** mô tả hệ thống sẽ được triển khai như thế nào, thành phần nào được cài đặt ở đâu, các liên kết vật lý hoặc giao thức truyền thông nào được sử dụng.

Dựa trên tính chất của các biểu đồ, UML chia các biểu đồ thành hai lớp mô hình¹:

- **Biểu đồ mô hình cấu trúc (Structural Modeling Diagrams):** biểu diễn các cấu trúc tĩnh của hệ thống phần mềm được mô hình hoá. Các biểu đồ trong mô hình tĩnh tập trung biểu diễn khía cạnh tĩnh của hệ thống, liên quan đến cấu trúc cơ bản cũng như các phần tử chính trong miền quan tâm của bài toán. Các biểu đồ trong mô hình tĩnh bao gồm:
 - Biểu đồ gói
 - Biểu đồ đối tượng và lớp
 - Biểu đồ thành phần
 - Biểu đồ triển khai
- **Biểu đồ mô hình hành vi (Behavioral Modeling Diagrams):** Nắm bắt đến các hoạt động và hành vi của hệ thống, cũng như tương tác giữa các phần tử bên trong và bên ngoài hệ thống. Các dạng biểu đồ trong mô hình động bao gồm:
 - Biểu đồ use case
 - Biểu đồ tương tác dạng tuần tự
 - Biểu đồ tương tác dạng cộng tác
 - Biểu đồ trạng thái
 - Biểu đồ động

Chúng ta sẽ lần lượt xem xét chi tiết các biểu đồ UML, mỗi biểu đồ sẽ được trình bày ý nghĩa của nó, tập kí hiệu UML cho biểu đồ đó và một ví dụ.

¹ Tham khảo http://www.sparxsystems.com.au/resources/uml2_tutorial/

2.2.1 Biểu đồ use case

a) Ý nghĩa

Biểu đồ use case biểu diễn sơ đồ chức năng của hệ thống. Từ tập yêu cầu của hệ thống, biểu đồ use case sẽ phải chỉ ra hệ thống cần thực hiện điều gì để thoả mãn các yêu cầu của người dùng hệ thống đó. Đi kèm với biểu đồ use case là các kịch bản (scenario). Có thể nói, biểu đồ use case chỉ ra sự tương tác giữa các *tác nhân* và hệ thống thông qua các use case.

Mỗi *use case* mô tả một chức năng mà hệ thống cần phải có xét từ quan điểm người sử dụng. *Tác nhân* là con người hay hệ thống thực khác cung cấp thông tin hay tác động tới hệ thống.

Một *biểu đồ use case* là một tập hợp các tác nhân, các use case và các mối quan hệ giữa chúng. Các use case trong biểu đồ use case có thể được phân rã theo nhiều mức khác nhau.

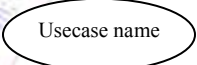

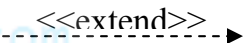
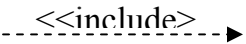
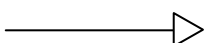

b) Tập ký hiệu UML cho biểu đồ use case

Một biểu đồ Use Case chứa các phần tử mô hình biểu thị hệ thống, tác nhân cũng như các trường hợp sử dụng và các mối quan hệ giữa các Use Case. Chúng ta sẽ lần lượt xem xét các phần tử mô hình này:

- a) *Hệ thống*: Với vai trò là thành phần của biểu đồ use case, hệ thống biểu diễn ranh giới giữa bên trong và bên ngoài của một chủ thể trong phần mềm chúng ta đang xây dựng. Chú ý rằng một hệ thống ở trong biểu đồ use case không phải bao giờ cũng nhất thiết là một hệ thống phần mềm; nó có thể là một chiếc máy, hoặc là một hệ thống thực (như một doanh nghiệp, một trường đại học, ...).
- b) *Tác nhân (actor)*: là người dùng của hệ thống, một tác nhân có thể là một người dùng thực hoặc các hệ thống máy tính khác có vai trò nào đó trong hoạt động của hệ thống. Như vậy, tác nhân thực hiện các use case. Một tác nhân có thể thực hiện nhiều use case và ngược lại một use case cũng có thể được thực hiện bởi nhiều tác nhân.
- c) *Các use case*: Đây là thành phần cơ bản của biểu đồ use case. Các use case được biểu diễn bởi các hình elip. Tên các use case thể hiện một chức năng xác định của hệ thống.
- d) *Mối quan hệ giữa các use case*: giữa các use case có thể có các mối quan hệ như sau:

- *Include*: use case này sử dụng lại chức năng của use case kia.
- *Extend*: use case này mở rộng từ use case kia bằng cách thêm vào một chức năng cụ thể.
- *Generalization*: use case này được kế thừa các chức năng từ use case kia.

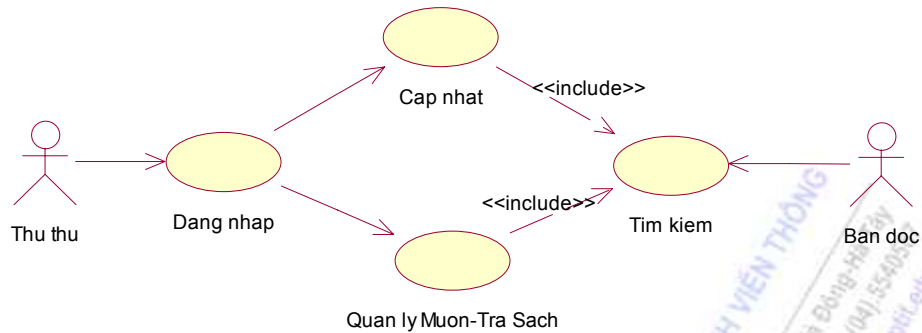
Các phần tử mô hình use case cùng với ý nghĩa và cách biểu diễn của nó được tổng kết trong bảng 2.2.

Phần tử mô hình	Ý nghĩa	Cách biểu diễn	Ký hiệu trong biểu đồ
Use case	Biểu diễn một chức năng xác định của hệ thống	Hình ellip chứa tên của use case	
Tác nhân	Là một đối tượng bên ngoài hệ thống tương tác trực tiếp với các use case	Biểu diễn bởi một lớp kiểu actor (hình người tượng trưng)	
Mối quan hệ giữa các use case	Tùy từng dạng quan hệ	Extend và include có dạng các mũi tên đứt nét Generalization có dạng mũi tên tam giác.	  
Biên của hệ thống	Tách biệt phần bên trong và bên ngoài hệ thống	Được biểu diễn bởi một hình chữ nhật rỗng.	

Bảng 2.2: Các phần tử mô hình trong biểu đồ use case

c) Ví dụ biểu đồ use case

Dưới đây là một use case cho hệ thống quản lý thư viện đơn giản. Người quản trị thư viện (thủ thư) thông qua đăng nhập để thực hiện Cập nhật thông tin và Quản lý các giao dịch mượn - trả sách. Bạn đọc chỉ có thể tìm kiếm, tra cứu thông tin sách. Chức năng tìm kiếm sách được dùng như một phần trong chức năng Cập nhật và Quản lý mượn sách nên chúng ta sử dụng quan hệ include. Chi tiết hơn về cách xây dựng biểu đồ này sẽ trình bày trong chương 3.



Hình 2.4: Biểu đồ use case tổng quát trong hệ thống quản lý thư viện

2.2.2 Biểu đồ lớp

a) Ý nghĩa

Trong phương pháp hướng đối tượng, *một nhóm đối tượng có chung một số thuộc tính và phương thức tạo thành một lớp*. Mỗi tương tác giữa các đối tượng trong hệ thống sẽ được biểu diễn thông qua mối quan hệ giữa các lớp.

Các lớp (bao gồm cả các thuộc tính và phương thức) cùng với các mối quan hệ sẽ tạo thành biểu đồ lớp. Biểu đồ lớp là một biểu đồ dạng mô hình tĩnh nhằm mô tả hướng nhìn tĩnh về một hệ thống bằng các khái niệm lớp, các thuộc tính, phương thức của lớp và mối quan hệ giữa chúng với nhau.

b) Tập ký hiệu UML cho biểu đồ lớp

Trong phần này, tài liệu sẽ xem xét các vấn đề liên quan đến biểu diễn sơ đồ lớp trong UML. Cuối phần này sẽ là một bảng tổng kết các ký hiệu UML sử dụng trong sơ đồ lớp.

- **Kí hiệu lớp:** trong UML, mỗi lớp được biểu diễn bởi hình chữ nhật gồm 3 phần: tên lớp, các thuộc tính và các phương thức.
- **Thuộc tính:** các thuộc tính trong biểu đồ lớp được biểu diễn theo cấu trúc chung như sau:

phạm_vi_tên : kiểu_số_đối_tượng = mặc_định (Giá_trị_giới_hạn)

Trong đó:

phạm_vi cho biết phạm vi truy nhập của thuộc tính. Có ba kiểu xác định thuộc tính phổ biến là:

+: thuộc tính kiểu public

#: thuộc tính kiểu protected

-: thuộc tính kiểu private.

~: thuộc tính được phép truy nhập tới từ các lớp trong cùng package

Các phạm vi của thuộc tính có thể được biểu diễn dưới dạng ký hiệu (+, #, -, ~) hoặc biểu diễn dưới dạng các từ khoá (public, protected, private).

Tên: là chuỗi ký tự biểu diễn tên thuộc tính.

kiểu: là kiểu dữ liệu của thuộc tính.

số_đối_tượng: chỉ ra số đối tượng khai báo cho thuộc tính ứng với một

mặc_định: là giá trị khởi đầu mặc định (nếu có) của thuộc tính.

Giá_trị_giới_hạn: là giới hạn các giá trị cho thuộc tính (thông tin này không bắt buộc).

Ví dụ một khai báo thuộc tính đầy đủ:

purchaseDate:Date[1] = "01-01-2000" (Saturday)

- **Phương thức (method)**: các phương thức trong UML được biểu diễn theo cấu trúc chung như sau [UNG]:

phạm_vi *tên(danh_sách_tham_số)*: *kiểu_trả_lại* { *ki_ều_phương_thức* }

Trong đó:

visibility biểu diễn phạm vi cho phương thức. Giống như đối với thuộc tính, có ba dạng kiểu xác định cơ bản cho phương thức là:

- +: phương thức kiểu public
- #: phương thức kiểu protected
- -: phương thức kiểu private
- ~: phương thức được phép truy nhập tới từ các lớp trong cùng package

tên là chuỗi ký tự xác định tên của phương thức.

kiểu_trả_lại: chỉ ra kiểu giá trị trả về của phương thức.

danh_sách_tham_số: biểu diễn danh sách các tham số trong khai báo của phương thức. Mỗi tham số được biểu diễn dưới dạng chung: *tên_tham_số*:

kiểu_giá_trị = *giá_trị_mặc_định*.

ki_ều_phương_thức: không bắt buộc, cho biết kiểu phương thức. Phương thức có thể nhận một trong các kiểu đặc biệt sau:

abstract: phương thức kiểu trừu tượng

query: phương thức kiểu truy vấn.

Ví dụ một khai báo phương thức cho một lớp:

generatePurchaseList(prodID:int): String

• **Các kiểu lớp trong UML**

UML định nghĩa một số kiểu lớp đặc biệt dựa trên vai trò của nó trong biểu đồ lớp. Ngoài kiểu lớp thông thường đã trình bày ở trên, UML còn định nghĩa một số kiểu lớp bổ sung gồm:

- *Lớp thực thể*: là lớp đại diện cho các thực thể chứa thông tin về các đối tượng xác định nào đó.
- *Lớp biên (lớp giao diện)*: là lớp nằm ở ranh giới giữa hệ thống với môi trường bên ngoài, thực hiện vai trò nhận yêu cầu trực tiếp từ các tác nhân và chuyển các yêu cầu đó cho các lớp bên trong hệ thống.
- *Lớp điều khiển*: thực hiện các chức năng điều khiển hoạt động của hệ thống ứng với các chức năng cụ thể nào đó với một nhóm các lớp biên hoặc lớp thực thể xác định.

STT	Kiểu lớp	Kí hiệu UML
1	<i>Lớp thực thể</i>	
2	<i>Lớp biên (lớp giao diện)</i>	
3	<i>Lớp điều khiển</i>	

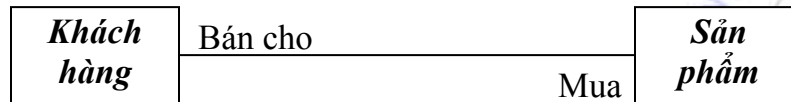
Bảng 2.3: Các kiểu lớp trong UML

• **Các mối quan hệ trong biểu đồ lớp**

Giữa các lớp có các dạng quan hệ cơ bản như sau:

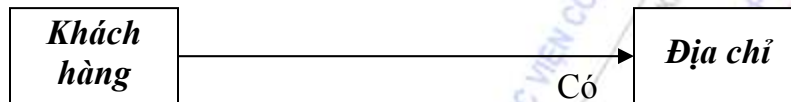
- *Quan hệ kết hợp (Association)*: Một kết hợp (association) là một sự nối kết giữa các lớp, cũng có nghĩa là sự nối kết giữa các đối tượng của các lớp này. Trong UML, một quan hệ được các định nhằm mô tả một tập hợp các liên kết (links), tức là một sự liên quan về ngữ nghĩa (semantic connection) giữa một nhóm các đối tượng được biểu diễn bởi các lớp tương ứng.

Mặc định, quan hệ kết hợp được biểu diễn bởi đoạn thẳng 2 chiều nối 2 đối tượng và có thể kèm theo ngữ nghĩa của quan hệ tại hai đầu của đoạn thẳng. Xem ví dụ Hình 2.5. Lớp khách hàng có quan hệ kết hợp với lớp sản phẩm. Ngữ nghĩa của quan hệ này thể hiện ở chỗ: khách hàng *mua* sản phẩm, còn sản phẩm *được bán cho* khách hàng.



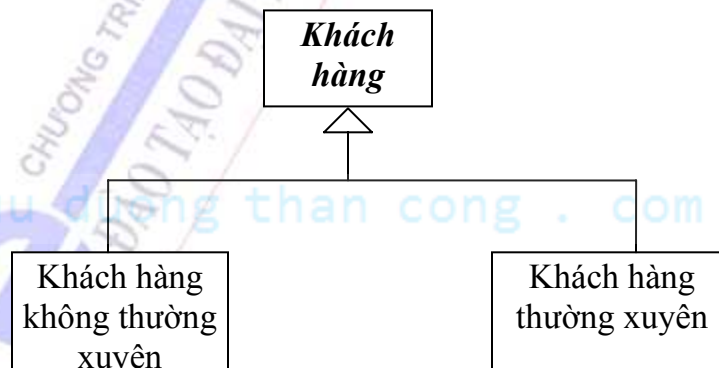
Hình 2.5: Quan hệ kết hợp

Quan hệ kết hợp cũng có thể có dạng một chiều. Xem ví dụ Hình 2.6.



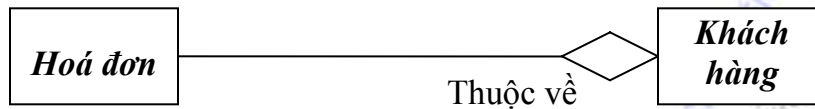
Hình 2.6: Quan hệ kết hợp một chiều

- **Khái quát hóa (Generalization):** Khái quát hóa là mối quan hệ giữa một lớp có các đặc trưng mang tính khái quát cao hơn và một lớp có tính chất đặc biệt hơn. Trong sơ đồ lớp, mối quan hệ khái quát hóa chính là sự kế thừa của một lớp từ lớp khác. Quan hệ khái quát hoá được biểu diễn bằng một mũi tên có tam giác rỗng gắn ở đầu. Xem ví dụ Hình 2.7.



Hình 2.7: Quan hệ khái quát hoá

- *Quan hệ cộng hợp (Aggregation)*: là dạng quan hệ mô tả một lớp A là một phần của lớp B và lớp A có thể tồn tại độc lập. Quan hệ cộng hợp được biểu diễn bằng một mũi tên gắn hình thoi rỗng ở đầu hướng về lớp bao hàm. Xem ví dụ Hình 2.8. Lớp Hoá đơn là một phần của lớp Khách hàng nhưng đối tượng Hoá đơn vẫn có thể tồn tại độc lập với đối tượng khách hàng.



Hình 2.8: Quan hệ cộng hợp

- *Quan hệ gộp (Composition)*: Một quan hệ gộp biểu diễn một quan hệ kiểu tổng thể-bộ phận. Lớp A có quan hệ gộp với lớp B nếu lớp A là một phần của lớp B và sự tồn tại của đối tượng lớp B điều khiển sự tồn tại của đối tượng lớp A. Quan hệ này được biểu diễn bởi một mũi tên gắn hình thoi đặc ở đầu. Xem ví dụ Hình 2.9.



Hình 2.9: Quan hệ gộp

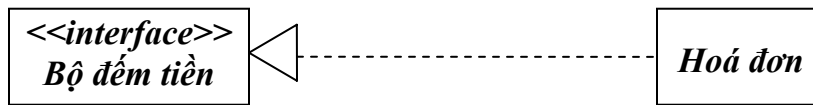
- *Quan hệ phụ thuộc (Dependency)*: Phụ thuộc là mối quan hệ giữa hai lớp đối tượng: một lớp đối tượng A có tính độc lập và một lớp đối tượng B phụ thuộc vào A; một sự thay đổi của A sẽ ảnh hưởng đến lớp phụ thuộc B. Xem ví dụ Hình 2.10.



Hình 2.10: Quan hệ phụ thuộc

- *Quan hệ thực thi (Realization)*: biểu diễn mối quan hệ ngữ nghĩa giữa các thành phần của biểu đồ lớp, trong đó một thành phần mô tả một công việc

dạng hợp đồng và thành phần còn lại thực hiện hợp đồng đó. Thông thường lớp *thực hiện hợp đồng* có thể là các giao diện. Xem ví dụ Hình 2.11.



Hình 2.11: Quan hệ thực thi

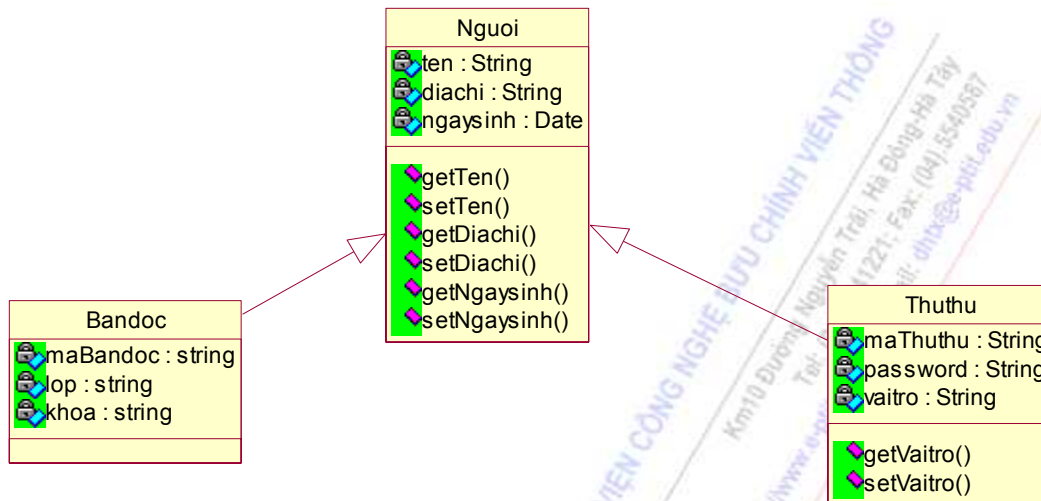
Bảng 2.4 tổng kết các phần tử mô hình UML được sử dụng trong mô hình lớp, ý nghĩa và ký hiệu tương ứng trong các biểu đồ.

Phần tử mô hình	Ý nghĩa	Cách biểu diễn	Ký hiệu trong biểu đồ
Lớp (class)	Biểu diễn tên lớp, các thuộc tính và phương thức của lớp đó.	Một hình chữ nhật gồm 3 phần tách biệt.	<div> <div>Tên lớp</div> <div>Các thuộc tính</div> <div>Các phương thức</div> </div>
Quan hệ kiểu kết hợp	Biểu diễn quan hệ giữa hai lớp độc lập, có liên quan đến nhau.	Một đường kẻ liền nét (có tên xác định) nối giữa hai lớp.	<div> <div></div> <div>Tên</div> <div></div> </div>
Quan hệ gộp	Biểu diễn quan hệ kiểu bộ phận – tổng thể.	Đường kẻ liền nét có hình thoi ở đầu.	<div> <div></div> <div></div> </div>
Quan hệ khái quát hoá (kế thừa)	Lớp này thừa hưởng các thuộc tính - phương thức của lớp kia	Mũi tên tam giác.	<div> <div></div> <div></div> </div>
Quan hệ phụ thuộc.	Các lớp phụ thuộc lẫn nhau trong hoạt động của hệ thống.	Mũi tên đứt nét.	<div> <div></div> <div></div> </div>

Bảng 2.4: Tóm tắt các phần tử mô hình UML trong biểu đồ lớp

c) Ví dụ biểu đồ lớp

Dưới đây là ví dụ một phần của biểu đồ lớp trong hệ thống quản lý thư viện trong đó các lớp Thủ thư (người quản lý thư viện) và Bạn đọc kế thừa từ lớp Person.



Hình 2.12: Biểu đồ lớp ví dụ

2.2.3 Biểu đồ trạng thái

a) Ý nghĩa

Biểu đồ trạng thái được sử dụng để biểu diễn các trạng thái và sự chuyển tiếp giữa các trạng thái của các đối tượng trong một lớp xác định. Thông thường, mỗi lớp sẽ có một biểu đồ trạng thái (trừ lớp trừu tượng là lớp không có đối tượng).

Biểu đồ trạng thái được biểu diễn dưới dạng máy trạng thái hữu hạn với các trạng thái và sự chuyển tiếp giữa các trạng thái đó. Có hai dạng biểu đồ trạng thái:

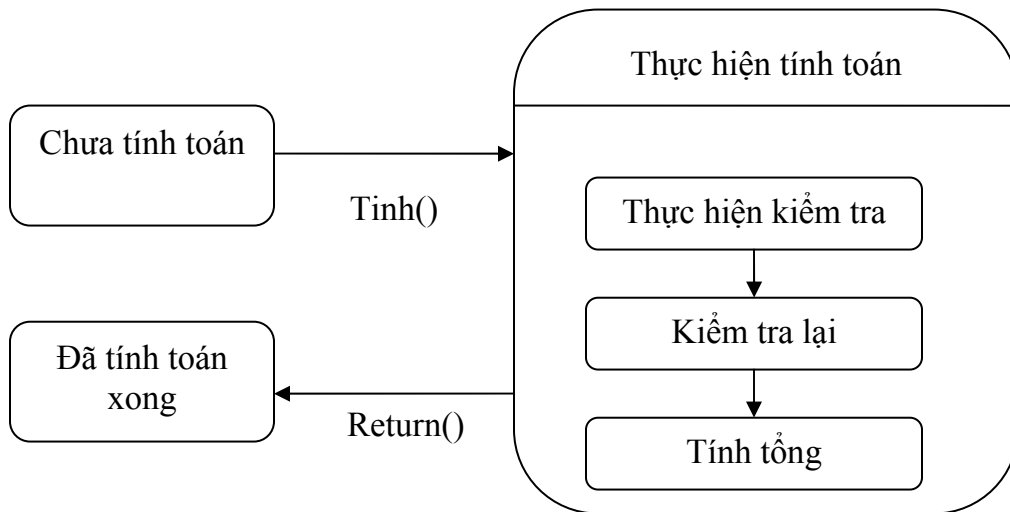
- Biểu đồ trạng thái cho một use case: mô tả các trạng thái và chuyển tiếp trạng thái của một đối tượng thuộc một lớp nào đó trong hoạt động của một use case cụ thể.
- Biểu đồ trạng thái hệ thống mô tả tất cả các trạng thái của một đối tượng trong toàn bộ hoạt động của cả hệ thống.

b) Tập ký hiệu UML cho biểu đồ trạng thái

Các thành phần trong một biểu đồ trạng thái bao gồm:

- **Trạng thái (state).** Bên trong các trạng thái có thể miêu tả các biến trạng thái hoặc các hành động (action) tương ứng với trạng thái đó.

- **Trạng thái con (substate):** là một trạng thái chứa bên trong một trạng thái khác. Trạng thái có nhiều trạng thái con gọi là trạng thái tổ hợp. Xem xét một ví dụ có trạng thái con trong Hình 2.13.



Hình 2.13: Biểu đồ trạng thái có trạng thái con

- **Trạng thái khởi đầu (initial state):** trạng thái đầu tiên khi kích hoạt đối tượng.
- **Trạng thái kết thúc (final state):** kết thúc vòng đời đối tượng.
- **Các chuyển tiếp (transition):** biểu diễn các chuyển đổi giữa các trạng thái.
- **Sự kiện (event):** sự kiện tác động gây ra sự chuyển đổi trạng thái. Mỗi sự kiện được đi kèm với các điều kiện (guard) và các hành động (action).

Trong biểu đồ trạng thái của UML, một số loại sự kiện sau đây sẽ được xác định:

- **Sự kiện gọi (call event):** Yêu cầu thực hiện một hành động (một phương thức)
- **Sự kiện tín hiệu (signal event):** Gửi thông điệp (chứa các giá trị thuộc tính tham số liên quan) giữa các trạng thái.
- **Sự kiện thời gian (time event):** Biểu diễn quá trình chuyển tiếp theo thời gian, thường kèm theo từ mô tả thời gian cụ thể.

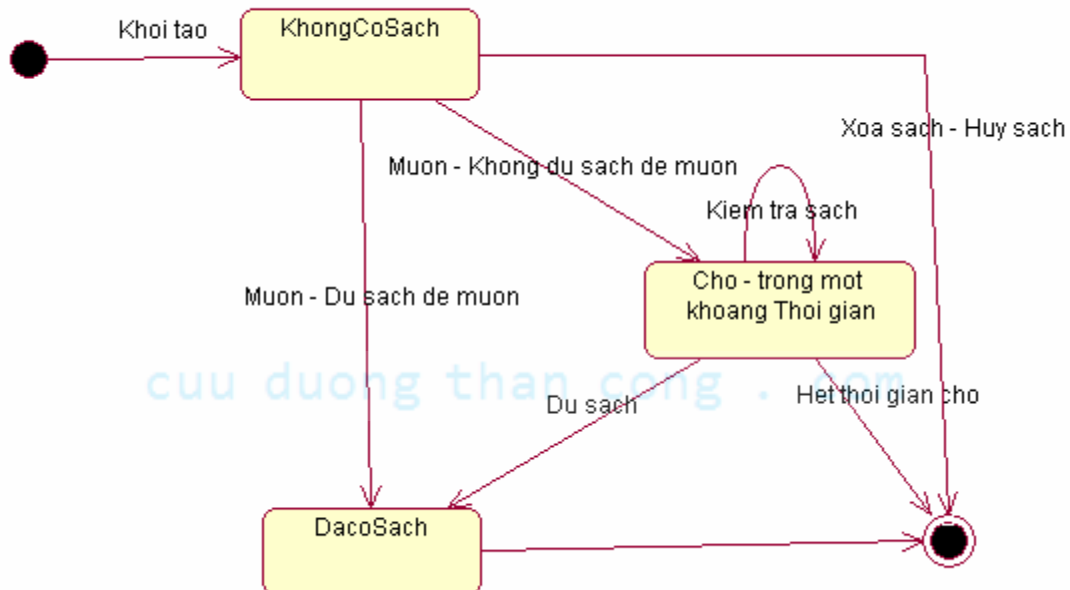
Các phần tử mô hình UML và ký hiệu tương ứng cho biểu đồ trạng thái được tổng kết như trong Bảng 2.5.

Phần tử mô hình	Ý nghĩa	Biểu diễn	Ký hiệu trong biểu đồ
<i>Trạng thái</i>	Biểu diễn một trạng thái của đối tượng trong vòng đời của đối tượng đó.	Hình chữ nhật vòng ở góc, gồm 3 phần: tên, các biến, và các hoạt động.	
<i>Trạng thái khởi đầu</i>	Khởi đầu vòng đời của đối tượng.	Hình tròn đặc	
<i>Trạng thái kết thúc</i>	Kết thúc vòng đời của đối tượng.	Hai hình tròn lồng nhau	
<i>Chuyển tiếp (transition)</i>	Chuyển từ trạng thái này sang trạng thái khác	Mũi tên liền nét với tên gọi là biểu diễn của chuyển tiếp đó.	

Bảng 2.5: Các phần tử mô hình UML trong biểu đồ trạng thái

c) Ví dụ Biểu đồ trạng thái

Dưới đây là biểu đồ trạng thái của lớp thẻ mượn trong chức năng Quản lý mượn sách. Chi tiết về xây dựng biểu đồ trạng thái này sẽ được trình bày trong Chương 3.



Hình 2.14: Ví dụ biểu đồ trạng thái

2.2.4 Biểu đồ tương tác dạng tuần tự

Các biểu đồ tương tác biểu diễn mối liên hệ giữa các đối tượng trong hệ thống và giữa các đối tượng với các tác nhân bên ngoài. Có hai loại biểu đồ tương tác: Biểu đồ tuần tự và biểu đồ cộng tác.

a) Ý nghĩa

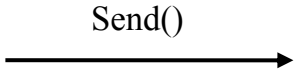
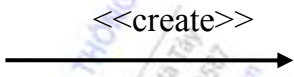
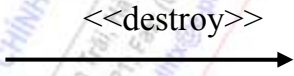
Biểu đồ tuần tự: Biểu diễn mối quan hệ giữa các đối tượng, giữa các đối tượng và tác nhân theo thứ tự thời gian. Biểu đồ tuần tự nhấn mạnh thứ tự thực hiện của các tương tác.

b) Tập ký hiệu UML cho biểu đồ tuần tự

Các thành phần cơ bản của một biểu đồ tuần tự là:

- *Các đối tượng (object):* được biểu diễn bởi các hình chữ nhật, bên trong là tên của đối tượng. Cách viết chung của đối tượng là: *tên đối tượng: tên lớp*. Nếu chỉ viết *:tên_lớp* thì có nghĩa là bất cứ đối tượng nào của lớp tương ứng đó. Trong biểu đồ tuần tự, không phải các đối tượng đều xuất hiện ở trên cùng của biểu đồ mà chúng chỉ xuất hiện (về mặt thời gian) khi thực sự tham gia vào tương tác.
- *Các message:* được biểu diễn bằng các mũi tên hướng từ đối tượng gửi sang đối tượng nhận. Tên các message có thể biểu diễn dưới dạng phi hình thức (như các thông tin trong kịch bản) hoặc dưới dạng hình thức (với dạng giống như các phương thức). Biểu đồ tuần tự cho phép có các message từ một đối tượng tới chính bản thân nó.
- Trong biểu đồ tuần tự có thể có nhiều loại message khác nhau tùy theo mục đích sử dụng và tác động của message đến đối tượng. Các dạng message được tổng kết trong Bảng 2.6 dưới đây:

STT	Loại message	Mô tả	Biểu diễn
1	Gọi (call)	Mô tả một lời gọi từ đối tượng này đến đối tượng kia.	Method() →
2	Trả về (return)	Trả về giá trị ứng với lời gọi	← -- _Giá trị trả về_

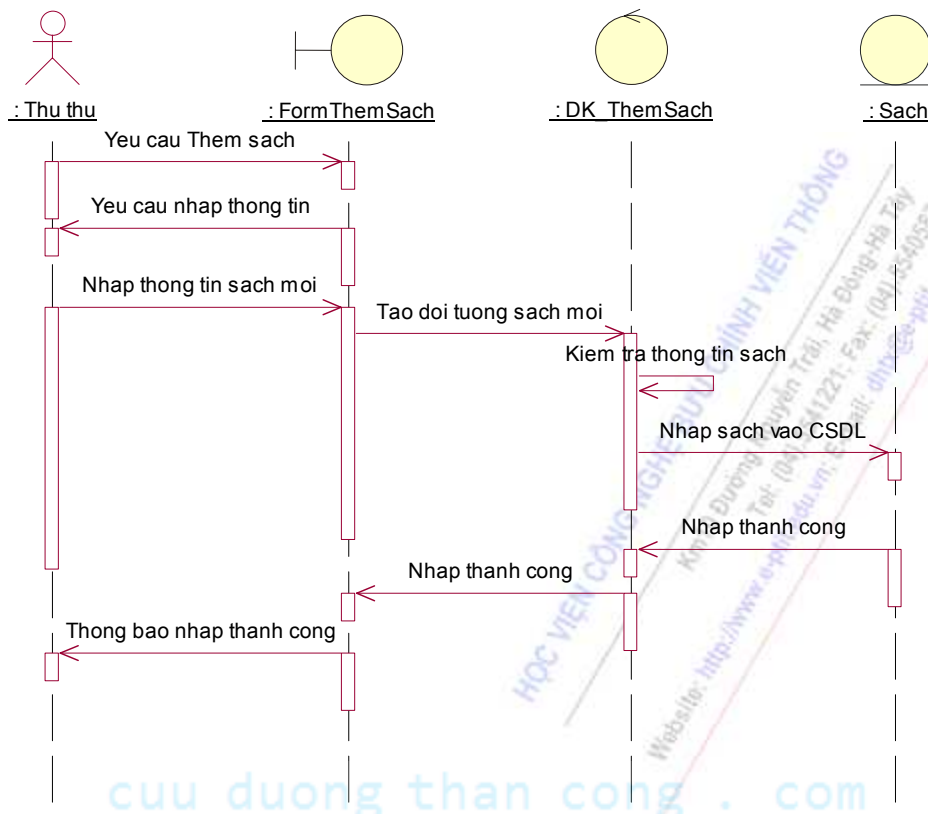
3	<i>Gửi (send)</i>	Gửi một tín hiệu tới một đối tượng	
4	<i>Tạo (create)</i>	Tạo một đối tượng	
5	<i>Hủy (destroy)</i>	Hủy một đối tượng	

Bảng 2.6: Các dạng message trong biểu đồ tuần tự

- *Đường lifeline*: là một đường kẻ nối dài phía dưới đối tượng, mô tả quá trình của đối tượng trong tương tác thuộc biểu đồ.
- *Chú thích*: biểu đồ tuần tự cũng có thể có chú thích để người đọc dễ dàng hiểu được nội dung của biểu đồ đó.

c) Ví dụ biểu đồ tương tác dạng tuần tự

Dưới đây là một ví dụ cho biểu đồ tuần tự cho chức năng Thêm sách trong hệ thống quản lý thư viện. Trong biểu đồ này có đối tượng giao diện FormThemSach, đối tượng điều khiển DK Thêm sách và đối tượng thực thể Sach. Chi tiết về xây dựng biểu đồ này sẽ được trình bày trong Chương 4.



Hình 2.15: Ví dụ biểu đồ tuần tự

2.2.5 Biểu đồ tương tác dạng cộng tác

a) Ý nghĩa

Biểu đồ cộng tác: Là biểu đồ tương tác biểu diễn mối quan hệ giữa các đối tượng; giữa các đối tượng và tác nhân nhấn mạnh đến vai trò của các đối tượng trong tương tác.

Biểu đồ cộng tác cũng có các message với nội dung tương tự như trong biểu đồ tuần tự. Tuy nhiên, các đối tượng được đặt một cách tự do trong không gian của biểu đồ và không có đường life line cho mỗi đối tượng. Các message được đánh số thể hiện thứ tự thời gian.

b) Tập ký hiệu UML cho biểu đồ cộng tác

Các thành phần cơ bản của một biểu đồ cộng tác là:

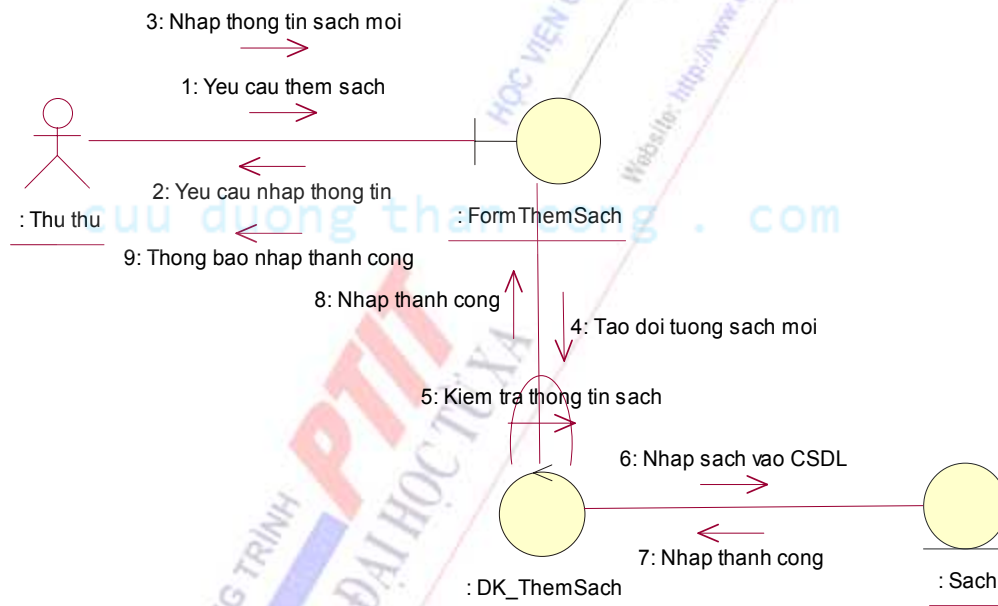
- *Các đối tượng:* được biểu diễn bởi các hình chữ nhật, bên trong là tên của đối tượng. Cách viết chung của đối tượng là: *tên đối tượng: tên lớp*. Trong

biểu đồ cộng tác, các đối tượng tham gia tương tác luôn xuất hiện tại một vị trí xác định.

- *Các liên kết*: giữa hai đối tượng có tương tác sẽ có một liên kết nối 2 đối tượng đó. Liên kết này không có chiều.
- *Các message*: được biểu diễn bằng các mũi tên hướng từ đối tượng gửi sang đối tượng nhận bên cạnh liên kết giữa 2 đối tượng đó. Trong biểu đồ cộng tác, các message được đánh số thứ tự theo thứ tự xuất hiện trong kịch bản mô tả use case tương ứng.

c) Ví dụ biểu đồ cộng tác

Dưới đây là một biểu đồ cộng tác mô tả chức năng Thêm sách trong hệ thống Quản lý thư viện.



Hình 2.16: Ví dụ Biểu đồ cộng tác

2.2.6 Biểu đồ hoạt động

a) Ý nghĩa

Biểu đồ hoạt động biểu diễn các hoạt động và sự đồng bộ, chuyển tiếp các hoạt động của hệ thống trong một lớp hoặc kết hợp giữa các lớp với nhau trong một chức năng cụ thể.

Biểu đồ hoạt động có thể được sử dụng cho nhiều mục đích khác nhau, ví dụ như:

- Để xác định các hành động phải thực hiện trong phạm vi một phương thức.
- Để xác định công việc cụ thể của một đối tượng.
- Để chỉ ra một nhóm hành động liên quan của các đối tượng được thực hiện như thế nào và chúng sẽ ảnh hưởng đến những đối tượng nằm xung quanh.

b) Tập ký hiệu UML

Các phần tử mô hình UML cho biểu đồ hoạt động bao gồm:

- *Hoạt động (Activity)*: là một quy trình được định nghĩa rõ ràng, có thể được thực hiện bởi một hàm hoặc một nhóm đối tượng. Hoạt động được thể hiện bằng hình chữ nhật tròn cạnh.
- *Thanh đồng bộ hóa (Synchronisation bar)*: cho phép ta mở ra hoặc là đóng lại các nhánh chạy song song trong tiến trình.








Hình 2.17: Thanh đồng bộ hóa trong biểu đồ động

- *Điều kiện (Guard Condition)*: các biểu thức logic có giá trị hoặc đúng hoặc sai. Điều kiện được thể hiện trong ngoặc vuông, ví dụ: [Customer existing].
- *Các luồng (swimlane)*: Mỗi biểu đồ động có thể biểu diễn sự phối hợp hoạt động trong nhiều lớp khác nhau. Khi đó mỗi lớp được phân tách bởi một luồng (swimlane) riêng biệt. Các luồng này được biểu diễn đơn giản là các ô khác nhau trong biểu đồ.

Các ký hiệu UML cho biểu đồ hoạt động được tổng kết trong Bảng sau:

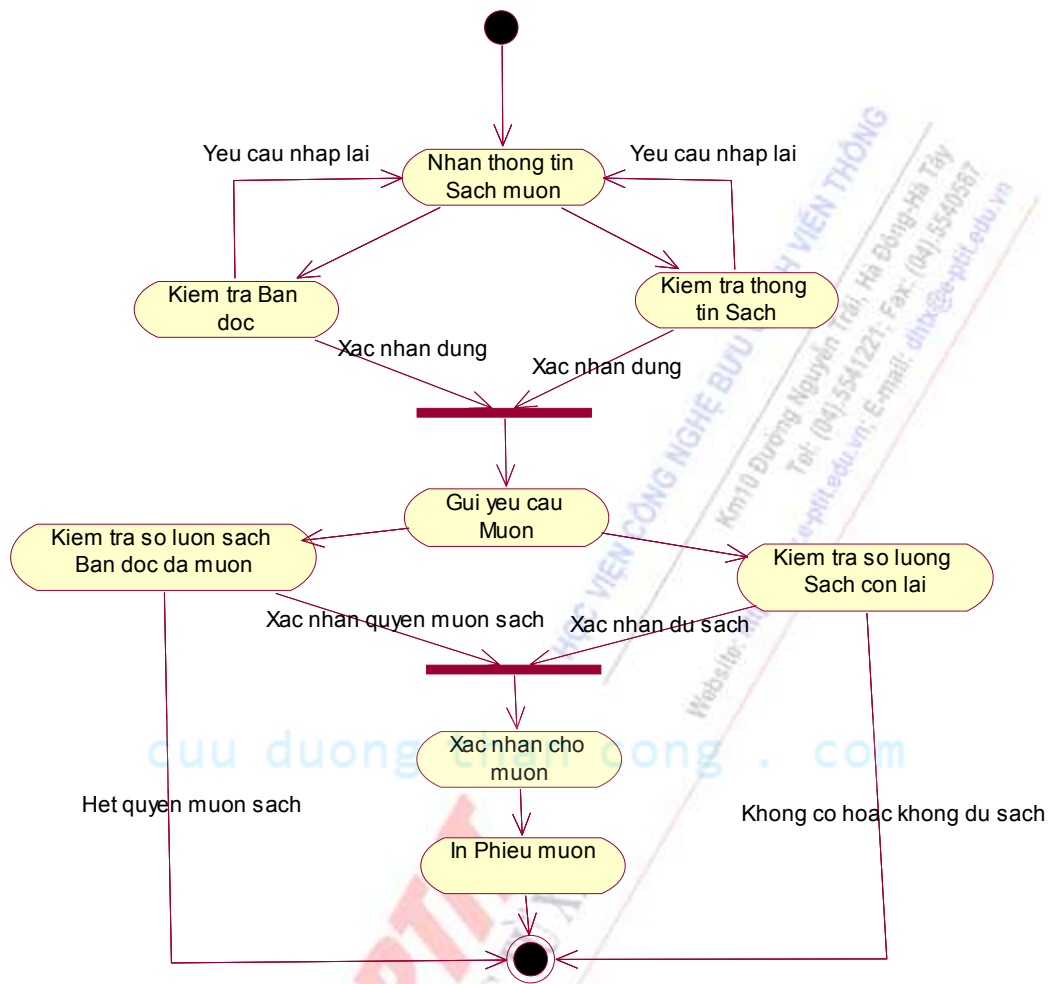
Phần tử mô hình	Ý nghĩa	Ký hiệu trong biểu đồ
<i>Hoạt động</i>	Mô tả một hoạt động gồm tên hoạt động và đặc tả của nó.	
<i>Trạng thái khởi đầu</i>		

<i>Trạng thái kết thúc</i>		
<i>Thanh đồng bộ ngang</i>	Mô tả thanh đồng bộ nằm ngang	
<i>Thanh đồng bộ hoá dọc</i>	Mô tả thanh đồng bộ theo chiều thẳng đứng	
<i>Chuyển tiếp</i>		
<i>Quyết định</i>	Mô tả một lựa chọn điều kiện.	
<i>Các luồng (swimlane)</i>	Phân tách các lớp đối tượng khác nhau tồn tại trong biểu đồ hoạt động	Phân cách nhau bởi một đường kẻ dọc từ trên xuống dưới biểu đồ

Bảng 2.6: Các phần tử của biểu đồ hoạt động

c) Ví dụ biểu đồ hoạt động

Dưới đây là ví dụ biểu đồ hoạt động của hàm thực hiện chức năng mượn sách trong lớp Thẻ mượn (Hệ thống quản lý thư viện). Chi tiết về biểu đồ này sẽ được trình bày trong chương 4.



Hình 2.18: Ví dụ biểu đồ hoạt động

2.2.7 Biểu đồ thành phần

a) Ý nghĩa

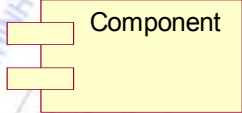
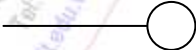
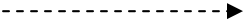
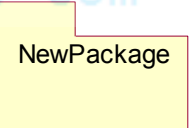
Biểu đồ thành phần được sử dụng để biểu diễn các thành phần phần mềm cấu thành nên hệ thống. Một hệ phần mềm có thể được xây dựng từ đầu bằng cách sử dụng mô hình lớp như đã trình bày trong các phần trước của tài liệu, hoặc cũng có thể được tạo nên từ các thành phần sẵn có.

Mỗi thành phần có thể coi như một phần mềm nhỏ hơn, cung cấp một khối dạng hộp đen trong quá trình xây dựng phần mềm lớn. Nói cách khác, các thành phần là các gói được xây dựng cho quá trình triển khai hệ thống. Các thành phần có thể là các gói ở mức cao như JavaBean, các gói thư viện liên kết động dll, hoặc

các phần mềm nhỏ được tạo ra từ các thành phần nhỏ hơn như các lớp và các thư viện chức năng.

b) Tập ký hiệu UML

Tập ký hiệu UML cho biểu đồ thành phần được tổng kết trong bảng sau:

Phần tử mô hình	Ý nghĩa	Ký hiệu trong biểu đồ
Thành phần	Mô tả một thành phần của biểu đồ, mỗi thành phần có thể chứa nhiều lớp hoặc nhiều chương trình con.	 Component
Giao tiếp	Mô tả giao tiếp gắn với mỗi thành phần. Các thành phần trao đổi thông tin qua các giao tiếp.	
Mối quan hệ phụ thuộc giữa các thành phần	Mối quan hệ giữa các thành phần (nếu có).	
Gói (package)	Được sử dụng để nhóm một số thành phần lại với nhau.	 NewPackage

Bảng 2.7: Các ký hiệu của biểu đồ thành phần

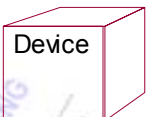


2.2.8 Biểu đồ triển khai hệ thống

a) Ý nghĩa

Biểu đồ triển khai biểu diễn kiến trúc cài đặt và triển khai hệ thống dưới dạng các nodes và các mối quan hệ giữa các node đó. Thông thường, các nodes được kết nối với nhau thông qua các liên kết truyền thông như các kết nối mạng, liên kết TCP-IP, microwave... và được đánh số theo thứ tự thời gian tương tự như trong biểu đồ cộng tác.

b) Tập ký hiệu UML cho biểu đồ triển khai

Tập ký hiệu UML cho biểu đồ triển khai hệ thống được biểu diễn trong Bảng sau:

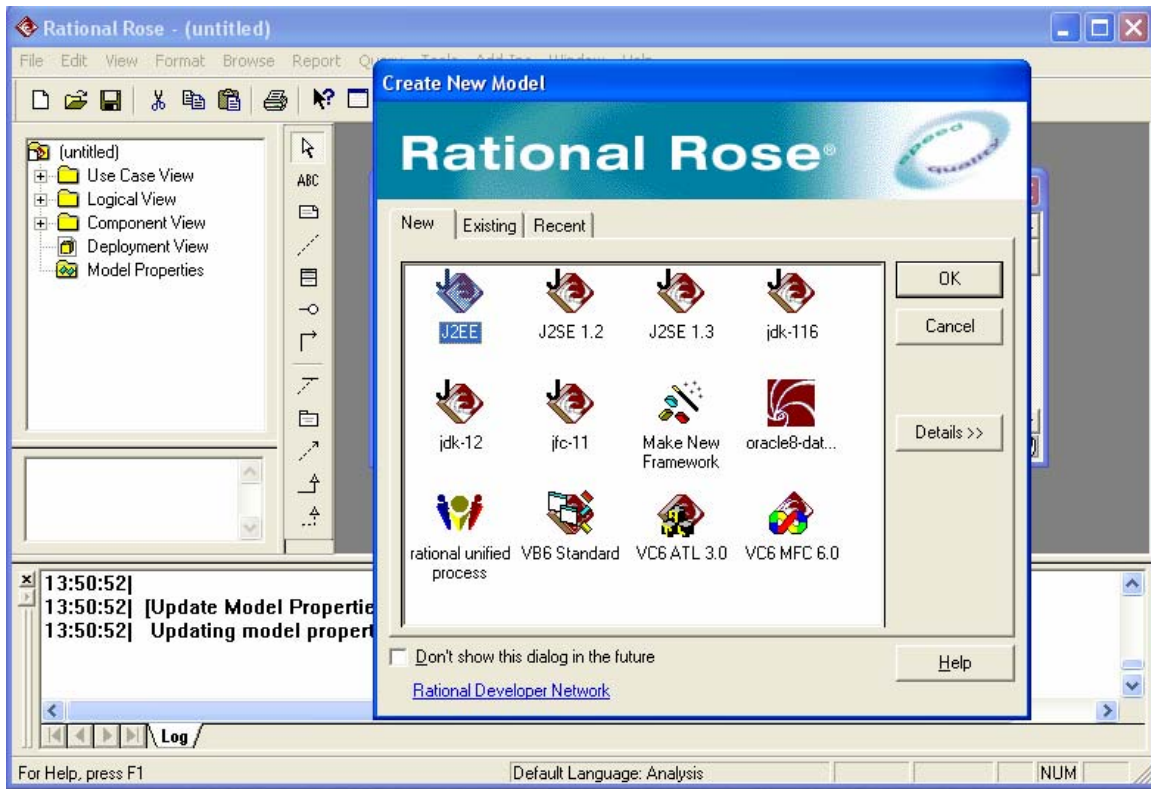
Phần tử mô hình	Ý nghĩa	Ký hiệu trong biểu đồ
<i>Các nodes (hay các thiết bị)</i>	Biểu diễn các thành phần không có bộ vi xử lý trong biểu đồ triển khai hệ thống	
<i>Các bộ xử lý</i>	Biểu diễn các thành phần có bộ vi xử lý trong biểu đồ triển khai hệ thống	
<i>Các liên kết truyền thông</i>	Nối các thành phần của biểu đồ triển khai hệ thống. Thường mô tả một giao thức truyền thông cụ thể.	

Bảng 2.8: Các ký hiệu của biểu đồ triển khai hệ thống

2.3 GIỚI THIỆU CÔNG CỤ RATIONAL ROSE

Rational Rose là một bộ công cụ được sử dụng cho phát triển các hệ phần mềm hướng đối tượng theo ngôn ngữ mô hình hóa UML. Với chức năng của một bộ công cụ trực quan, Rational Rose cho phép chúng ta tạo, quan sát, sửa đổi và quản lý các biểu đồ. Tập ký hiệu mà Rational Rose cung cấp thống nhất với các ký hiệu trong UML. Ngoài ra, Rational Rose còn cung cấp chức năng hỗ trợ quản lý dự án phát triển phần mềm, cung cấp các thư viện để hỗ trợ sinh khung mã cho hệ thống theo một ngôn ngữ lập trình nào đó.

Màn hình khởi động của Rational Rose phiên bản 2002 như trong Hình 2.19. Người sử dụng sẽ có thể chọn thư viện dự định sẽ cài đặt hệ thống, Rational Rose sẽ tải về các gói tương ứng trong thư viện đó. Các gói này (cùng các lớp tương ứng) sẽ xuất hiện trong biểu đồ lớp, người sử dụng sẽ tiếp tục phân tích, thiết kế hệ thống của mình dựa trên thư viện đó. Nếu sử dụng Rational Rose để xây dựng hệ thống từ đầu thì người sử dụng nên bỏ qua chức năng này.

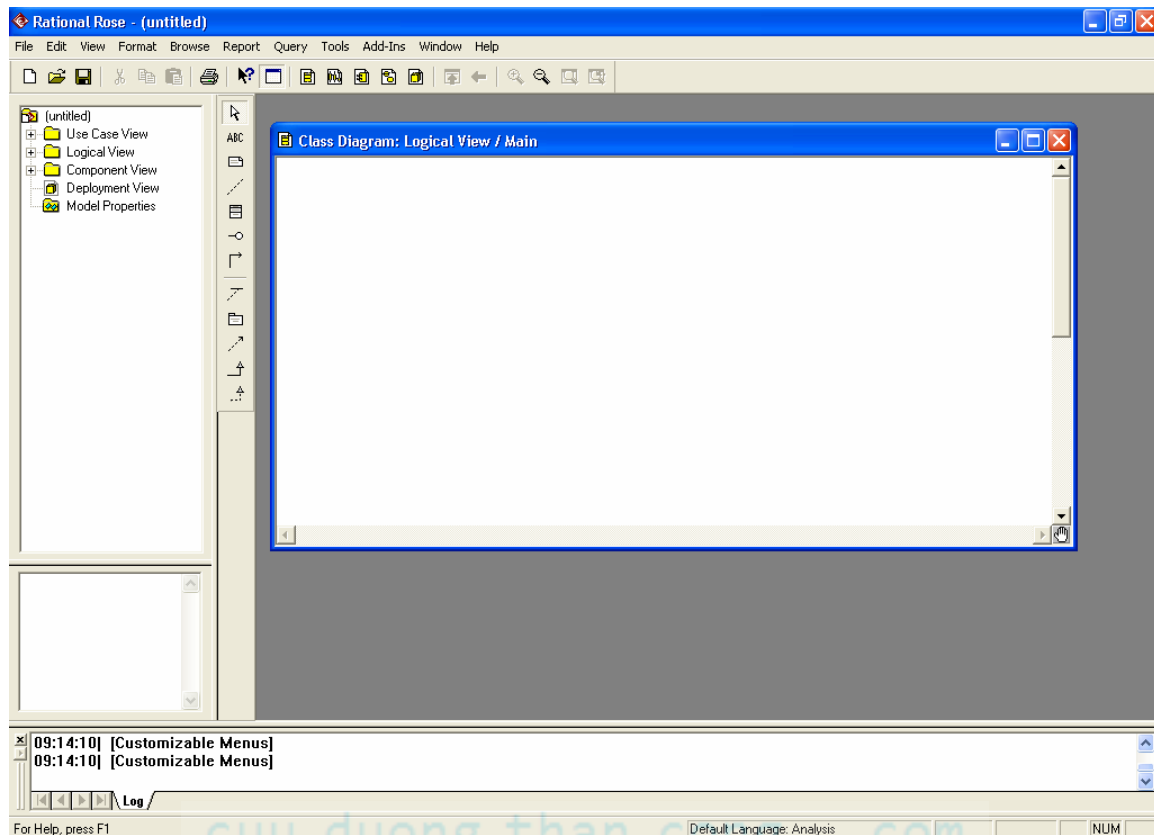


Hình 2.19: Màn hình khởi động Rational Rose

Trong giao diện của Rational Rose (Hình 2.19), cửa sổ phía bên trái là cửa sổ Browser chứa các View (hướng nhìn, quan điểm), trong mỗi View là các mô hình tương ứng của UML. Có thể xem mỗi View là một cách nhìn theo một khía cạnh nào đó của hệ thống.

- *Use Case View*: xem xét khía cạnh chức năng của hệ thống nhìn từ phía các tác nhân bên ngoài
- *Logical View*: xem xét quá trình phân tích và thiết kế logic của hệ thống để thực hiện các chức năng trong Use Case View.
- *Component View*: xem xét khía cạnh tổ chức hệ thống theo các thành phần và mối liên hệ giữa các thành phần đó.
- *Deployment View*: xem xét khía cạnh triển khai hệ thống theo các kiến trúc vật lý.

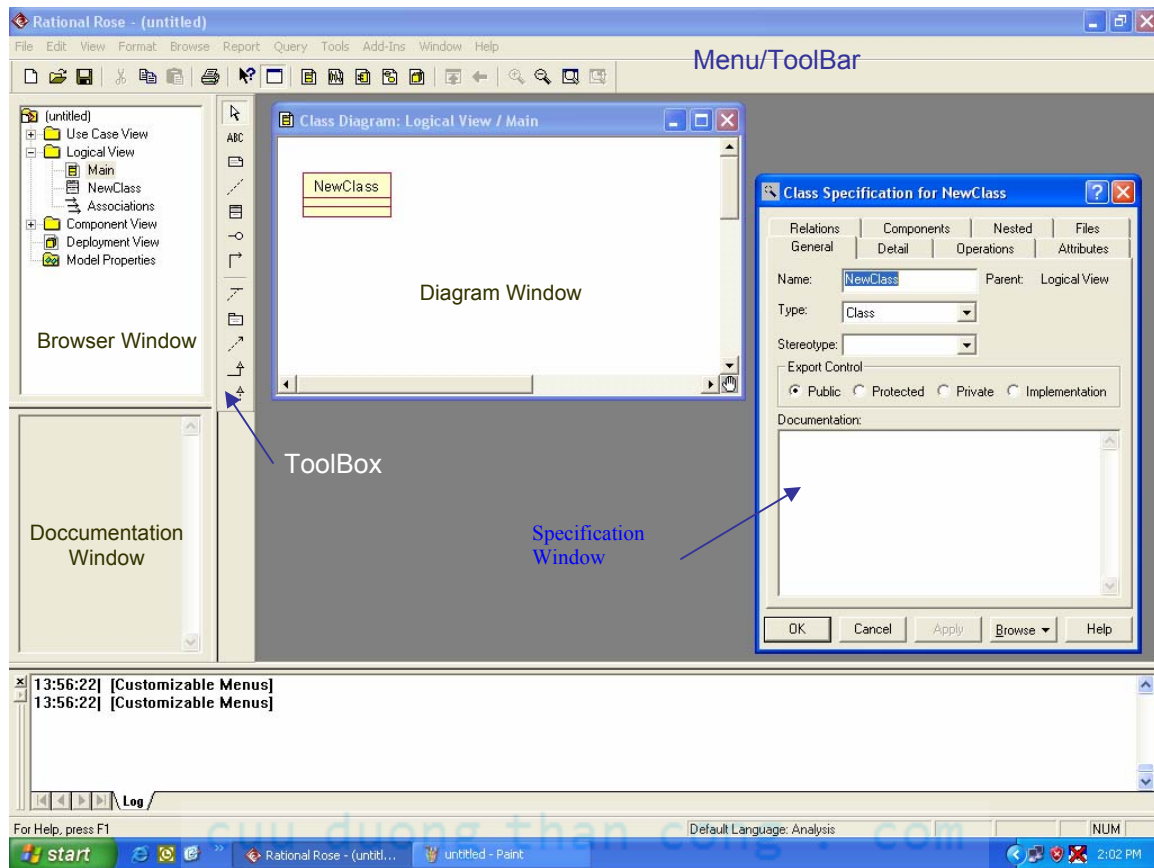
Cửa sổ phía bên phải của màn hình Rational Rose là cửa sổ biểu đồ (Diagram Windows) được sử dụng để vẽ các biểu đồ sử dụng các công cụ vẽ tương ứng trong Toolbox. Hầu hết các ký hiệu sử dụng để vẽ biểu đồ trong Rational Rose đều thống nhất với chuẩn UML.



Hình 2.20: Giao diện chính của Rational Rose

Giao diện chính của Rational Rose trong các biểu đồ đều được chia thành các phần như trong Hình 2.21. Ý nghĩa chính của các thành phần này như sau:

- MenuBar và ToolBar chứa các menu và công cụ tương tự như các ứng dụng Windows khác.
- Phần Browser Window cho phép người sử dụng chuyển tiếp nhanh giữa các biểu đồ trong các View.
- Phần Documentation Window dùng để viết các thông tin liên quan đến các phần tử mô hình tương ứng trong biểu đồ. Các thông tin này có thể là các ràng buộc, mục đích, các từ khóa ... liên quan đến phần tử mô hình đó.
- Phần Toolbox chứa các công cụ dùng để vẽ biểu đồ. Ứng với mỗi dạng biểu đồ thì sẽ có một dạng toolbox tương ứng.



Hình 2.21: Các thành phần trong giao diện Rational Rose

- Phần Diagram Window là không gian để vẽ và hiệu chỉnh các biểu đồ trong mô hình tương ứng.
- Cửa sổ Specification Window là đặc tả chi tiết của mỗi phần tử mô hình theo các trường thông tin tương ứng với dạng biểu đồ đó.

Vấn đề sử dụng Rational Rose cho các bước cụ thể trong phân tích thiết kế hệ thống sẽ được trình bày chi tiết trong Chương 3 và 4 của tài liệu này.

TỔNG KẾT CHƯƠNG 2

Chương 2 đã giới thiệu ngôn ngữ mô hình hoá thống nhất UML và công cụ Rational Rose cho phát triển phần mềm hướng đối tượng. Các nội dung chính cần ghi nhớ:

- UML ra đời từ sự kết hợp các phương pháp luận phát triển phần mềm hướng đối tượng khác nhau đã có trước đó. UML hiện nay đã được coi là ngôn ngữ mô hình hoá chuẩn cho phát triển các phần mềm hướng đối tượng.

- UML được chia thành nhiều hướng nhìn, mỗi hướng nhìn quan tâm đến hệ thống phần mềm từ một khía cạnh cụ thể.
- Nếu xét theo tính chất mô hình thì UML có hai dạng mô hình chính là mô hình tĩnh và mô hình động. Mỗi mô hình lại bao gồm một nhóm các biểu đồ khác nhau.
- Mỗi biểu đồ UML có một tập ký hiệu riêng để biểu diễn các thành phần của biểu đồ đó. Quá trình biểu diễn các biểu đồ cũng phải tuân theo các quy tắc được định nghĩa trong UML.
- Hiện nay có rất nhiều công cụ hỗ trợ phân tích thiết kế hệ thống hướng đối tượng sử dụng UML trong đó bộ công cụ Rational Rose là bộ công cụ được sử dụng rất rộng rãi với nhiều tính năng ưu việt. Các ví dụ trong tài liệu này đều được xây dựng và biểu diễn trên Rational Rose.

CÂU HỎI – BÀI TẬP

1. UML ra đời từ các ngôn ngữ và phương pháp mô hình hóa nào?
2. Hướng nhìn là gì? UML bao gồm các hướng nhìn nào?
3. Liệt kê các biểu đồ của UML và tập ký hiệu UML cho từng biểu đồ đó.
4. Liệt kê các bước phát triển phần mềm hướng đối tượng sử dụng UML
5. Phân biệt mô hình tĩnh và mô hình động trong UML?
6. Phân biệt các dạng quan hệ trong biểu đồ lớp như: quan hệ khái quát hóa, quan hệ kết hợp, quan hệ cộng hợp, quan hệ gộp.
7. Phân biệt biểu đồ tuần tự và biểu đồ cộng tác. Các chú ý khi biểu diễn hai biểu đồ này.

CHƯƠNG 3

PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

Chương này trình bày các bước phân tích hướng đối tượng, các khái niệm và quy tắc liên quan đến quá trình phân tích hệ thống. Nội dung cụ thể gồm:

- Tổng quan các bước của pha phân tích hướng đối tượng
- Bước xây dựng mô hình use case và kịch bản
- Bước xây dựng mô hình lớp
- Bước xây dựng mô hình động dựa trên biểu đồ trạng thái

3.1 TỔNG QUAN VỀ PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

3.1.1 Vai trò của pha phân tích

Trong các bước của vòng đời phát triển phần mềm nói chung, pha phân tích (hay đặc tả) có các nhiệm vụ sau:

- Thiết lập một cách nhìn tổng quan rõ ràng về hệ thống và các mục đích chính của hệ thống cần xây dựng.
- Liệt kê các nhiệm vụ mà hệ thống cần thực hiện.
- Phát triển một bộ từ vựng để mô tả bài toán cũng như những vấn đề liên quan trong miền quan tâm của bài toán.
- Đưa ra hướng giải quyết bài toán.

Như vậy, pha phân tích chỉ dừng lại ở mức xác định các đặc trưng mà hệ thống cần phải xây dựng là gì, chỉ ra các khái niệm liên quan và tìm ra hướng giải quyết bài toán chứ chưa quan tâm đến cách thức thực hiện xây dựng hệ thống như thế nào. Như cách nói trong ngôn ngữ tiếng Anh, pha phân tích nhằm trả lời cho câu hỏi “what”, còn câu hỏi “how” sẽ được trả lời trong pha thiết kế.

3.1.2 Các bước phân tích hướng đối tượng

Phân tích hướng đối tượng được chia làm ba bước tương ứng với ba dạng mô hình UML là:

- **Mô hình use case:** bước này nhằm xây dựng mô hình chức năng của sản phẩm phần mềm. Các chức năng này được nhìn từ quan điểm của những người sử dụng hệ thống. Kết quả của bước này là một biểu đồ use case được phân cấp cùng các scenario tương ứng của từng use case, trong đó biểu diễn đầy đủ các chức năng của hệ thống và được khách hàng chấp nhận.
- **Mô hình lớp:** biểu diễn các lớp, các thuộc tính và mối quan hệ giữa các lớp. Từ tập các use case và scenario, nhóm phát triển hệ thống sẽ phải chỉ ra các lớp, xác định các thuộc tính, các phương thức và các mối quan hệ giữa các lớp.
- **Mô hình động:** biểu diễn các hoạt động liên quan đến một lớp hay lớp con. Các hoạt động này được biểu diễn dưới dạng tương tự như sơ đồ máy trạng thái hữu hạn và được gọi là biểu đồ trạng thái. Ngoài biểu đồ trạng thái, trong mô hình động còn có các biểu đồ khác là: biểu đồ tương tác (gồm cả biểu đồ tuần tự, biểu đồ cộng tác) và biểu đồ động. Tuy nhiên, trong pha phân tích, người phát triển hệ thống chỉ quan tâm đến biểu đồ trạng thái cho mỗi lớp đã xác định được trong mô hình lớp.

3.1.3 Ví dụ

Để minh họa cho các bước phân tích cũng như trong pha thiết kế ở Chương 4, chúng ta hãy xét một *hệ quản lý thư viện* đơn giản. Giới hạn của hệ thống này được thể hiện qua các yêu cầu sau:

- Tài liệu trong thư viện bao gồm: sách, báo, tạp chí ... được mô tả chung gồm các thuộc tính: tên tài liệu, tác giả, nhà xuất bản, năm xuất bản, số lượng hiện có.
- Đối với các bạn đọc: thực hiện các thao tác tìm tài liệu, mượn, trả tài liệu và xem xét các thông tin về tài liệu mà mình đang mượn. Việc tìm kiếm tài liệu được thực hiện trực tiếp qua mạng. Tuy nhiên, giao dịch mượn và trả sách phải thực hiện trực tiếp tại thư viện.

- Quá trình mượn và trả tài liệu thông qua một *thẻ mượn* ghi đầy đủ nội dung liên quan đến bạn đọc và tài liệu được mượn; thời gian bắt đầu mượn và thời hạn phải trả.
- Đối với người quản lý thư viện (thủ thư): được phép cập nhật các thông tin liên quan đến tài liệu và bạn đọc.

Bài toán này sẽ được sử dụng làm ví dụ trong quá trình thực hiện các bước phân tích và thiết kế hệ thống (Chương 3, 4). Tài liệu phân tích thiết kế hệ thống sẽ được trình bày đầy đủ trong phần Phụ lục.

3.2 MÔ HÌNH USE CASE VÀ KỊCH BẢN

3.2.1 Vai trò của mô hình use case

Khi bắt đầu xây dựng một sản phẩm phần mềm, nhóm phát triển phải xác định các chức năng mà hệ thống cần phải thực hiện là gì. Biểu đồ use case được sử dụng để xác định các chức năng cũng như các tác nhân (người sử dụng hay hệ thống khác) liên quan đến hệ thống đó.

Có thể coi một use case là tập hợp của một loạt các kịch bản (scenario) liên quan đến việc sử dụng hệ thống theo một cách thức nào đó. Mỗi kịch bản (scenario) mô tả một chuỗi các sự kiện mà một người hay một hệ thống khác kích hoạt vào hệ thống đang phát triển theo tuần tự thời gian. Những thực thể tạo nên các chuỗi sự kiện như thế được gọi là các *tác nhân* (Actor). Một hệ thống sẽ bao gồm nhiều use case, liên kết với nhau bởi các mối quan hệ nào đó. Biểu đồ use case được phân rã thành các mức tương ứng với các chức năng ở các cấp độ khác nhau, nhìn từ quan điểm người sử dụng hệ thống. Sự cần thiết phải xây dựng biểu đồ use case thể hiện qua một số điểm sau:

- Use case là một công cụ tốt để người dùng tiếp cận và mô tả các chức năng của hệ thống theo quan điểm của mình. Biểu đồ use case được biểu diễn trực quan, do đó khách hàng và những người dùng tiềm năng của hệ thống có thể dễ dàng mô tả được những ý định thực sự của mình.
- Biểu đồ use case sẽ làm cho khách hàng và người dùng tiềm năng tham gia cùng nhóm phát triển trong bước khởi đầu của quá trình phân tích thiết kế hệ thống. Điều này sẽ giúp cho nhóm phát triển và khách hàng có được sự thống nhất chung về các chức năng thực sự cần thiết của hệ thống.

- Biểu đồ use case là cơ sở cho những bước tiếp theo của quá trình phân tích thiết kế hệ thống phần mềm. Dựa trên biểu đồ use case và các scenario, người phát triển hệ thống sẽ chỉ ra các lớp cần thiết cũng như các thuộc tính của các lớp đó.

Các mục tiêu chính cần đạt được của các use case là:

- Cần chỉ ra và mô tả được các yêu cầu mang tính chức năng của hệ thống, đây là kết quả rút ra từ sự thỏa thuận giữa khách hàng (và/hoặc người sử dụng cuối) và nhóm phát triển phần mềm.
- Đưa ra một mô tả rõ ràng và nhất quán về việc hệ thống cần phải làm gì, làm sao để mô hình có thể được sử dụng nhất quán trong suốt toàn bộ quá trình phát triển và tạo thành nền tảng cho việc thiết kế các chức năng sau này.
- Tạo nên một nền tảng cho các bước kiểm thử hệ thống, đảm bảo hệ thống thỏa mãn đúng những yêu cầu do người sử dụng đưa ra. Trong thực tế thường là để trả lời câu hỏi: Liệu hệ thống cuối cùng có thực hiện những chức năng mà khởi đầu khách hàng đã đề nghị hay không?
- Cung cấp khả năng theo dõi quá trình chuyển các yêu cầu về mặt chức năng thành các lớp cụ thể cũng như các phương thức cụ thể trong hệ thống.
- Đơn giản hóa việc thay đổi và mở rộng hệ thống qua việc thay đổi và mở rộng mô hình Use Case. Khi hệ thống cần thay đổi (thêm bớt các chức năng nào đó), người phát triển hệ thống chỉ cần bổ sung trong biểu đồ use case cho phù hợp, sau đó chỉ theo dõi riêng những use case đã bị thay đổi cùng những ảnh hưởng của chúng trong thiết kế hệ thống và xây dựng hệ thống.

Những công việc cụ thể cần thiết để tạo nên một mô hình Use Case bao gồm:

- 1. Xác định các tác nhân và các Use Case**
- 2. Xác định các mối quan hệ và phân rã biểu đồ use case**
- 3. Biểu diễn các use case thông qua các kịch bản**
- 4. Kiểm tra và hiệu chỉnh mô hình**

Nội dung cụ thể thực hiện trong mỗi bước này sẽ được trình bày cụ thể trong phần sau của tài liệu.

3.2.2 Xây dựng biểu đồ use case

Phần này sẽ trình bày quá trình xây dựng biểu đồ use case theo UML và áp dụng trong bộ công cụ Rational Rose.

Bước 1: Tìm các tác nhân và các use case

Để tìm các tác nhân, người phát triển hệ thống cần trả lời các câu hỏi sau:

- Ai (hay hệ thống nào) sẽ là người sử dụng những chức năng chính của hệ thống? (trả lời câu hỏi này ta sẽ tìm được các tác nhân chính).
- Ai cần sự hỗ trợ của hệ thống để thực hiện những công việc hàng ngày của họ?
- Ai sẽ cần bảo trì, quản trị và đảm bảo cho hệ thống hoạt động (tác nhân phụ)?
- Hệ thống sẽ phải xử lý và làm việc với những trang thiết bị phần cứng nào?
- Hệ thống cần phải tương tác với các hệ thống nào khác? Cần phân biệt hệ thống mà chúng cần phải xây dựng với các hệ thống sẽ tương tác với nó. Nghĩa là, cần xác định rõ biên giới giữa hệ thống yêu cầu xây dựng với hệ thống khác có thể bao gồm các hệ thống máy tính cũng như các ứng dụng khác trong chính chiếc máy tính mà hệ thống này sẽ hoạt động trong tương lai.
- Ai hay cái gì quan tâm đến kết quả mà hệ thống sẽ sản sinh ra?

Xem xét bài toán quản lý thư viện, các chức năng chính của hệ thống quản lý thư viện được thực hiện bởi thủ thư và bạn đọc của thư viện đó. Như vậy, chúng ta có hai tác nhân là *thủ thư* và *bạn đọc*, trong đó *bạn đọc* không phân biệt là sinh viên hay giáo viên.

Từ các tác nhân đã tìm được ở trên, người phát triển hệ thống sẽ tìm ra các use case qua việc xem xét các câu hỏi sau trên mỗi tác nhân:

- Tác nhân đó cần chức năng nào từ hệ thống. Hành động chính của tác nhân này là gì?
- Tác nhân cần phải xem, cập nhật hay lưu trữ thông tin gì trong hệ thống?
- Tác nhân có cần thông báo cho hệ thống những sự kiện nào đó hay không? Những sự kiện như thế đại diện cho những chức năng nào?
- Hệ thống có cần thông báo cho tác nhân khi có thay đổi trong hệ thống hay không?
- Hệ thống cần có những chức năng gì để đơn giản hóa các công việc của tác nhân?

Trong bài toán quản lý thư viện mà chúng ta đang xét, tác nhân bạn đọc, anh ta cần các chức năng liên quan đến tìm kiếm tài liệu, xem thông tin cá nhân, đăng ký mượn và trả sách. Còn tác nhân thủ thư sẽ thực hiện cập nhật các thông tin liên quan đến bạn đọc và các thông tin về tài liệu, thực hiện các giao dịch mượn và trả sách. Dựa vào đó, ta đã xác định được một số use case như: *tìm kiếm tài liệu, cập nhật, cập nhật bạn đọc, cập nhật tài liệu, quản lý mượn sách, quản lý trả sách, xem thông tin cá nhân*.

Ngoài ra, use case còn được xác định thông qua các câu hỏi khác như sau:

- Ngoài các tác nhân, các chức năng của hệ thống còn có thể được sinh ra bởi sự kiện nào khác (như sự kiện thời gian, tác động của chức năng khác, ...).
- Hệ thống cần những thông tin đầu vào đầu ra nào?

Trong bài toán quản lý thư viện, để cập nhật được thông tin, thủ thư phải thông qua việc đăng nhập hệ thống. Hay nói cách khác, sự kiện đăng nhập hệ thống sẽ là điều kiện cho use case cập nhật. Vậy ta sẽ cần thêm use case *cập nhật*.

Bước 2: Xác định mối quan hệ và phân rã biểu đồ use case

Trong sơ đồ use case, các dạng quan hệ sẽ được sử dụng trong các trường hợp tương ứng như sau:

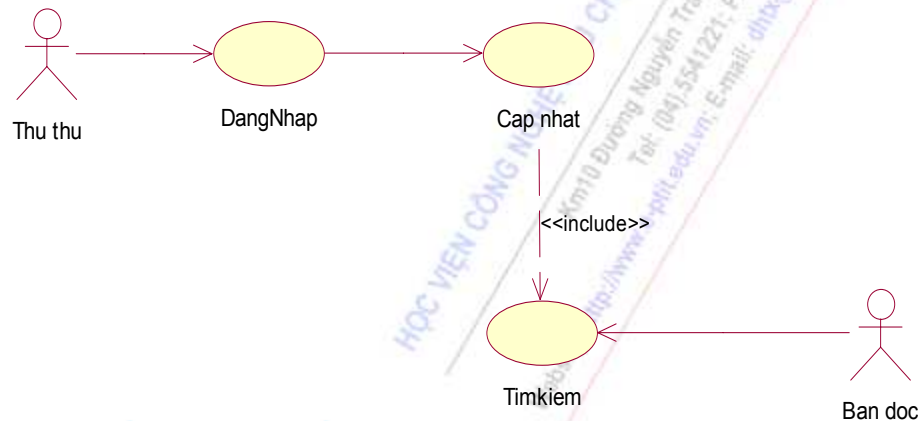
- *Quan hệ <<include>>*: sử dụng để chỉ ra rằng một use case được sử dụng bởi một use case khác.
- *Quan hệ mở rộng <<extend>>*: sử dụng để chỉ ra rằng một use case được mở rộng từ một use case khác bằng cách thêm vào một chức năng cụ thể.
- *Quan hệ generalization*: biểu thị use case này là tổng quát còn use case kia là cụ thể hóa của use case đó.
- *Quan hệ kết hợp*: thường dùng để biểu diễn mối liên hệ giữa actor và các use case (một actor kích hoạt một use case).

Dựa trên các mối quan hệ trên, biểu đồ use case được biểu diễn lại thành dạng phân cấp gọi là phân rã biểu đồ use case. Nguyên tắc phân rã biểu đồ use case như sau:

- *Xác định sơ đồ use case mức tổng quát*: từ tập tác nhân và use case đã được xác định ở bước trước, người phát triển cần tìm ra các chức năng chính của hệ thống. Các chức năng này phải có tính tổng quát, dễ dàng nhìn thấy được trên quan điểm của các tác nhân. Các dạng quan hệ thường dùng

trong sơ đồ use case mức tổng quát là quan hệ kết hợp, quan hệ tổng quát hóa và quan hệ include.

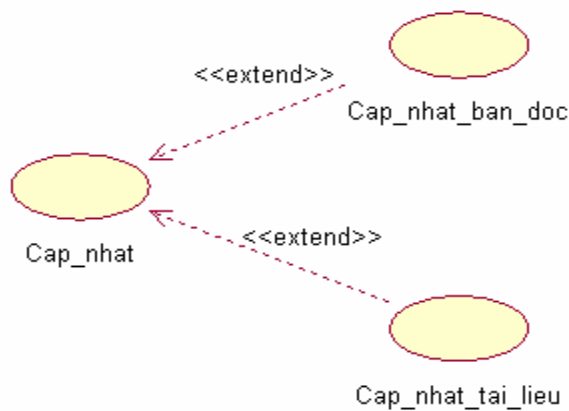
Ví dụ trong bài toán quản lý thư viện, xét trên quan điểm của các tác nhân *bạn đọc*, *thủ thư*, nếu tạm thời chưa xét đến các chức năng mượn và trả sách thì các chức năng tổng quát của hệ thống là: *đăng nhập*, *cập nhật* và *tìm kiếm*. Trong các use case này, use case cập nhật “include” chức năng của use case tìm kiếm (Hình 3.1).



Hình 3.1: Biểu đồ use case mức tổng quát trong bài toán quản lý thư viện

- **Phân rã các use case mức cao:** người phát triển tiến hành phân rã các use case tổng quát thành các use case cụ thể hơn sử dụng quan hệ “extend”. Các use case con (mức thấp) được lựa chọn bằng cách thêm vào use case cha một chức năng cụ thể nào đó và thường được mở rộng dựa trên cơ sở sự chuyển tiếp và phân rã các chức năng của hệ thống.

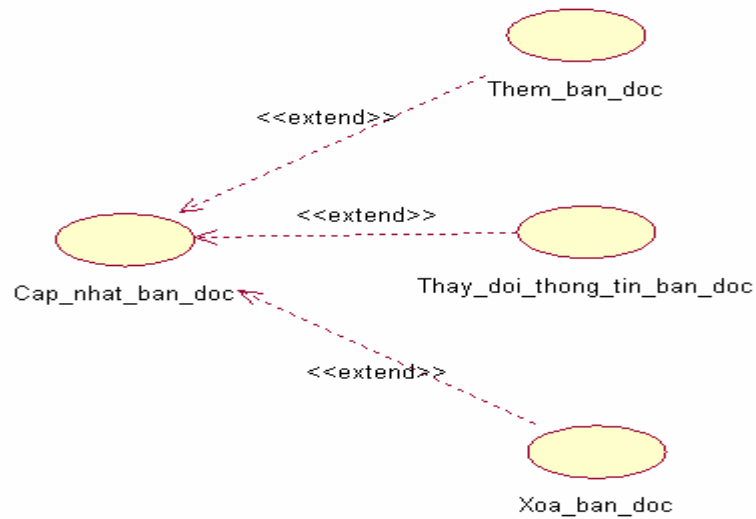
Ví dụ, trong bài toán quản lý thư viện, use case *cập nhật* có thể được phân rã thành cập nhật bạn đọc và cập nhật tài liệu (Hình 3.2)



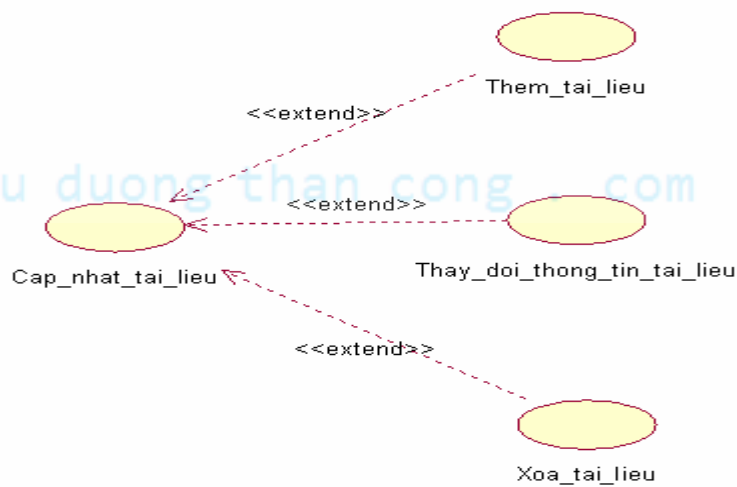
Hình 3.2: Phân rã use case cập nhật

- **Tiếp tục phân rã sơ đồ use case cho đến khi gặp use case ở nút lá.** Các use case ở nút lá thường gắn với một chức năng cụ thể trong đó hệ thống thực sự tương tác với các tác nhân (gửi kết quả đến các tác nhân hoặc yêu cầu tác nhân nhập thông tin ...). Trong các sơ đồ use case mức 2, nếu còn có use case nào chưa phải là nút lá thì cần tiếp tục được phân rã.

Trong ví dụ về bài toán quản lý thư viện, các use case *cập nhật bạn đọc* và *cập nhật tài liệu* đều có thể tiếp tục phân rã thành các use case con là *thêm bạn đọc*, *thay đổi thông tin bạn đọc* và *xóa bạn đọc* hay *thêm tài liệu*, *thay đổi thông tin tài liệu* và *xóa tài liệu*. Các use case này đã là nút lá vì nó biểu diễn một chức năng cụ thể của hệ thống trong đó có tương tác giữa tác nhân *thủ thư* và hệ thống (Hình 3.3 và Hình 3.4).



Hình 3.3: Phân rã use case Cập nhật bạn đọc



Hình 3.4: Phân rã use case cập nhật tài liệu

- **Hoàn thiện biểu đồ use case:** người phát triển tiến hành xem xét lại xem tất cả các use case đã được biểu diễn trong biểu đồ use case (ở tất cả các mức) hay chưa. Nếu còn có use case chưa có trong biểu đồ nào, người phát triển phải xem xét xem chức năng mà use case đó đại diện đã được thực hiện bởi các use case khác chưa để bổ sung thêm hoặc loại bỏ use case đó ra khỏi biểu đồ.

Bước 3: Biểu diễn các use case bởi kịch bản (scenario)

Sau khi hoàn thành phân rã biểu đồ use case, công việc tiếp theo của người phát triển hệ thống là biểu diễn các scenario tương ứng với các use case đó. Các scenario được biểu diễn theo mẫu chung như trong Bảng 3.1.

	Ý nghĩa
Tên Use case:	Tên use case
Tác nhân chính:	Tác nhân chính của use case
Mức:	Mức của use case trong biểu đồ phân rã
Người chịu trách nhiệm:	Người chịu trách nhiệm chính trong hoạt động của use case
Tiền điều kiện:	Tiền điều kiện: khi nào use case được kích hoạt.
Đảm bảo tối thiểu:	Đảm bảo tối thiểu: đảm bảo trong trường hợp use case thất bại.
Đảm bảo thành công:	Đảm bảo thành công: kết quả trong trường hợp use case hoàn thành.
Kích hoạt:	Sự kiện tác động kích hoạt use case.
Chuỗi sự kiện chính: 1. 2. 3.	Scenario chuẩn (trong trường hợp thành công)
Ngoại lệ: 1.a Ngoại lệ xảy ra ở bước 1 1.a.1 1.a.2 3.a Ngoại lệ xảy ra ở bước 3 3.a.1 3.a.2	Các scenario ngoại lệ tương ứng với các bước trong scenario chuẩn.

Bảng 3.1: Mẫu chung cho scenario

Bảng 3.2 biểu diễn scenario cho use case *Thêm sách* trong bài toán quản lý thư viện.

Tên use case	Thêm sách
Tác nhân chính	Thủ thư
Mức	3
Người chịu trách nhiệm	Người quản lý thư viện
Tiền điều kiện	Thủ thư đã đăng nhập vào hệ thống.
Đảm bảo tối thiểu	Hệ thống loại bỏ các thông tin đã thêm và quay lui lại bước trước.
Đảm bảo thành công	Thông tin về sách mới được bổ sung vào CSDL
Kích hoạt	Thủ thư chọn chức năng cập nhật sách trong menu.
<p>Chuỗi sự kiện chính:</p> <ol style="list-style-type: none"> 1. Hệ thống hiển thị form thêm sách và yêu cầu thủ thư đưa vào thông tin sách. 2. Thủ thư nhập thông tin về sách mới và nhấn Submit. 3. Hệ thống kiểm tra thông tin sách và xác nhận thông tin sách hợp lệ 4. Hệ thống nhập thông tin sách mới vào CSDL 5. Hệ thống thông báo đã nhập thành công. 6. Thủ thư thoát khỏi chức năng thêm sách. 	
<p>Ngoại lệ:</p> <ol style="list-style-type: none"> 3.a Hệ thống thông báo sách đã có trong CSDL. <ol style="list-style-type: none"> 3.a.1 Hệ thống hỏi thủ thư có thêm số lượng sách hay không. 3.a.2 Thủ thư thêm số lượng sách 3.a.3 Hệ thống thêm số lượng cho sách đã có 3.a.4 Hệ thống thông báo nhập thành công. 3.b Hệ thống thông báo thông tin sách không hợp lệ <ol style="list-style-type: none"> 3.b.1 Hệ thống yêu cầu thủ thư nhập lại thông tin. 3.b.2 Thủ thư nhập lại thông tin sách. 	

Bảng 3.2: Scenario cho use case Thêm sách

Bước 4: Hiệu chỉnh mô hình

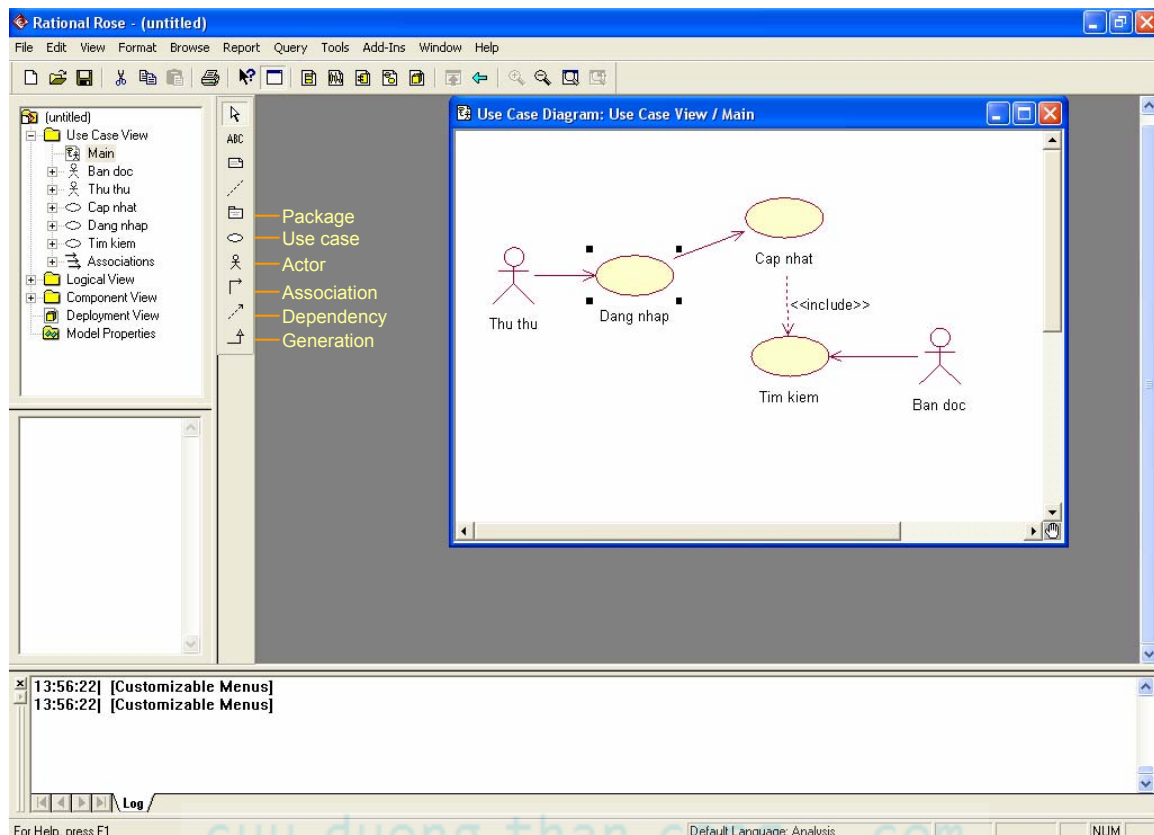
Bước này thực hiện kiểm tra lại toàn bộ biểu đồ use case, bổ sung hoặc thay đổi các thông tin nếu cần thiết. Trong bước này, toàn bộ biểu đồ use case cùng các scenario và các tài liệu khác liên quan sẽ được chuyển cho khách hàng xem xét. Nếu khách hàng có điều gì chưa nhất trí, nhóm phát triển sẽ phải sửa đổi lại biểu đồ use case cho phù hợp. Bước này chỉ kết thúc khi khách hàng và nhóm phát triển hệ thống có được sự thống nhất.

3.2.3 Xây dựng biểu đồ use case trong Rational Rose

Biểu đồ use case được xây dựng trong Use Case View của Rational Rose (Hình 3.5). Các công cụ thông thường sử dụng trong biểu đồ use case gồm use case, actor, các quan hệ association và dependency đều xuất hiện trong ToolBox tương ứng của biểu đồ use case.

Các bước xây dựng biểu đồ use case trong Rational Rose là:

1. *Biểu diễn các tác nhân*
2. *Biểu diễn và đặc tả các use case mức tổng quát*
3. *Biểu diễn các mối quan hệ*
4. *Phân rã biểu đồ use case và đặc tả các use case mức thấp*



Hình 3.5: Giao diện của biểu đồ use case

Bước 1: Biểu diễn các tác nhân. Để thêm vào biểu đồ một tác nhân, ta thực hiện các bước sau:

- B1. Chọn công cụ actor trên hộp công cụ
- B2. Đưa con trỏ vào vùng màn hình diagram và đặt vào vị trí thích hợp
- B3. Mở cửa sổ đặc tả actor và viết tên của tác nhân

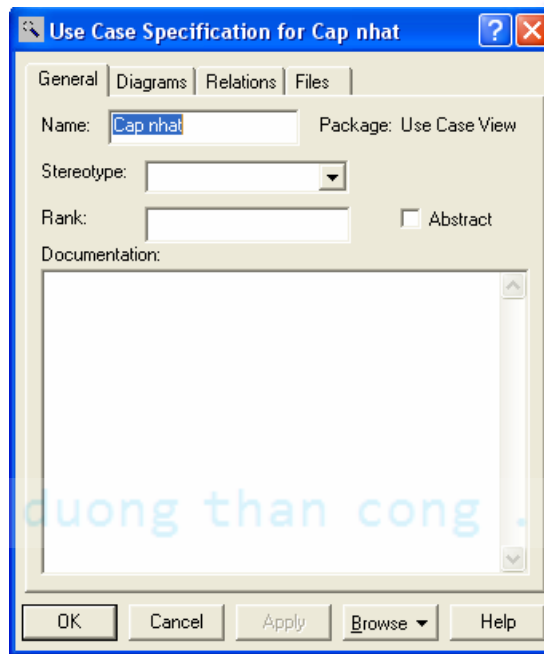
Bước 2: Biểu diễn các use case mức cao

- B1. Chọn công cụ use case trên hộp công cụ
- B2. Đưa con trỏ vào màn hình diagram và đặt use case cần tạo vào vị trí thích hợp
- B3. Mở cửa sổ đặc tả use case, đặt tên cho use case và mô tả các thông tin khác.

Cửa sổ Specification của một use case được biểu diễn như trong Hình 3.6. Trong cửa sổ này có các thanh Tab:

- Tab General đưa ra các thông tin chung về use case như tên, kiểu...

- Tab Diagram cho biết các biểu đồ đi kèm của use case đó (khi mở rộng một use case thì biểu đồ mức dưới sẽ xuất hiện ở đây).
- Tab Relations liệt kê các mối quan hệ của use case đó với các use case và actor khác.
- Tab Files là các file kèm theo use case (có thể là các scenario hoặc các dạng file khác).

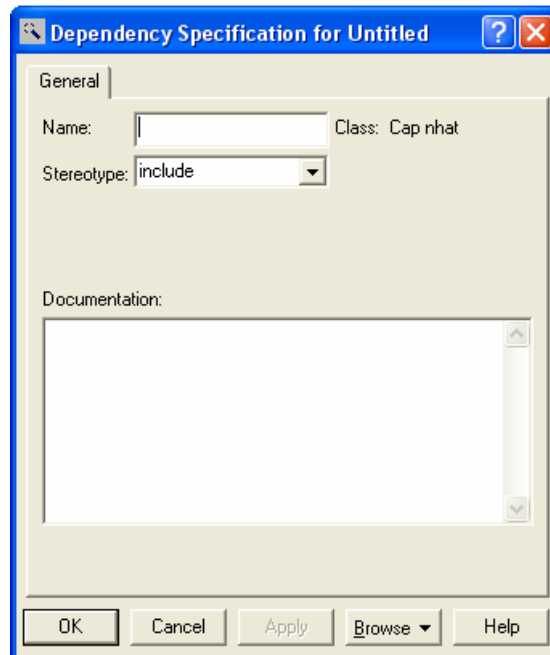


Hình 3.6: Cửa sổ đặc tả một use case

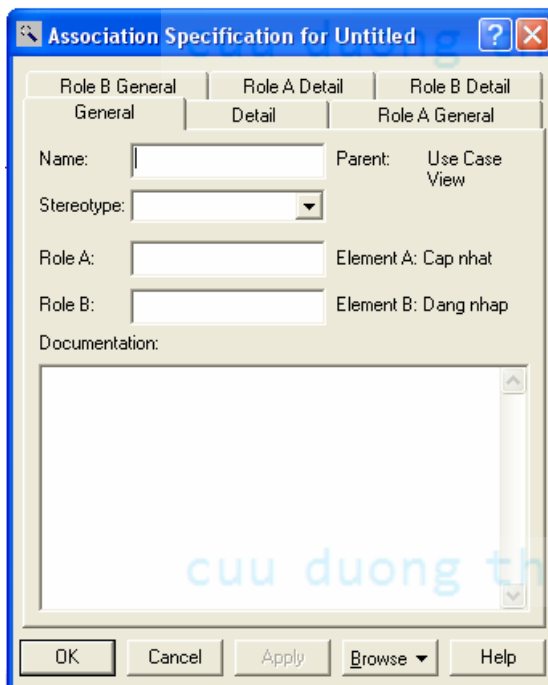
Bước 3: Biểu diễn và đặc tả các quan hệ

- B1. Chọn kiểu quan hệ tương ứng trong hộp công cụ: (quan hệ association, dependency).
- B2. Đặt con trỏ vào đối tượng khởi đầu quan hệ (actor hoặc use case) và kéo đến đối tượng cuối.
- B3. Mở cửa sổ đặc tả quan hệ để chọn kiểu quan hệ và đặt tên quan hệ cùng một số thông tin khác.

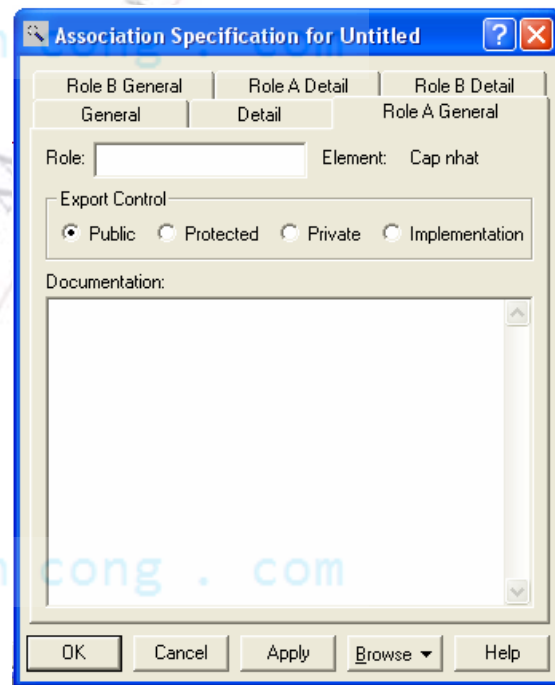
Tương tự với các use case, quan hệ giữa các use case cũng có một cửa sổ đặc tả tương ứng. Một trong những điểm quan trọng nhất trong đặc tả một quan hệ giữa các use case là chỉ ra stereotype của quan hệ đó. Hình 3.7 là cửa sổ đặc tả quan hệ kiểu phụ thuộc (Dependency). Hình 3.8.a và 3.8.b là hai Tab khác nhau của cửa sổ đặc tả quan hệ dạng kết hợp (association).



Hình 3.7: Cửa sổ đặc tả một quan hệ dạng Dependency



Hình 3.8.a: Đặc tả quan hệ association – Tab General



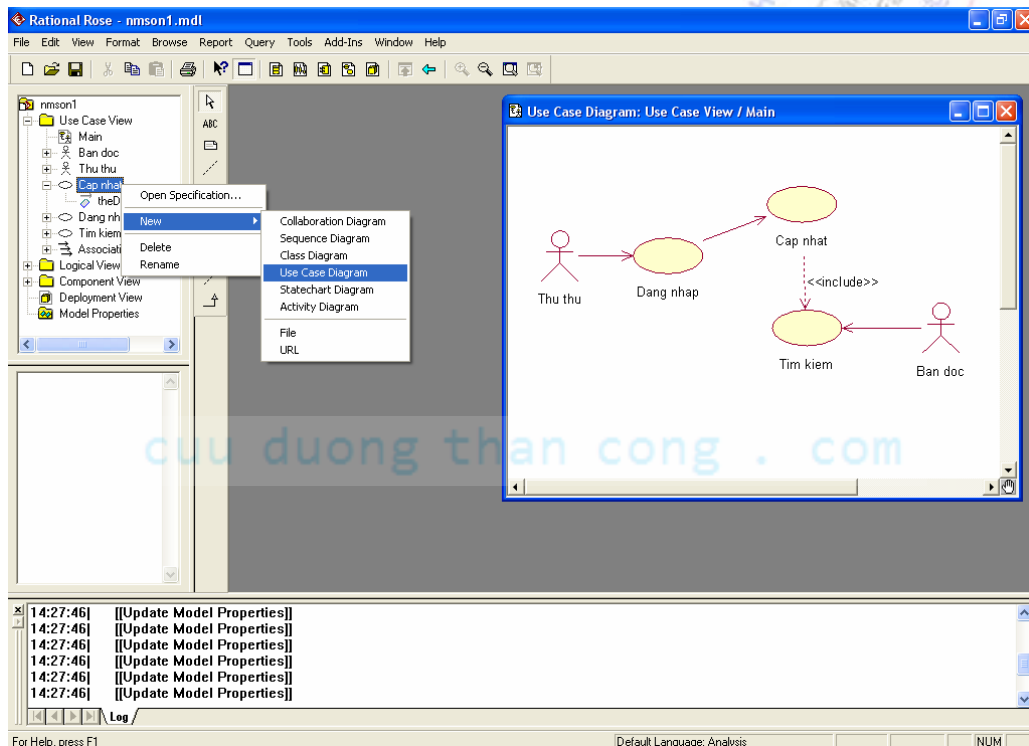
Hình 3.8.b: Đặc tả quan hệ association – Tab Role A General

Bước 4: Phân rã biểu đồ use case.

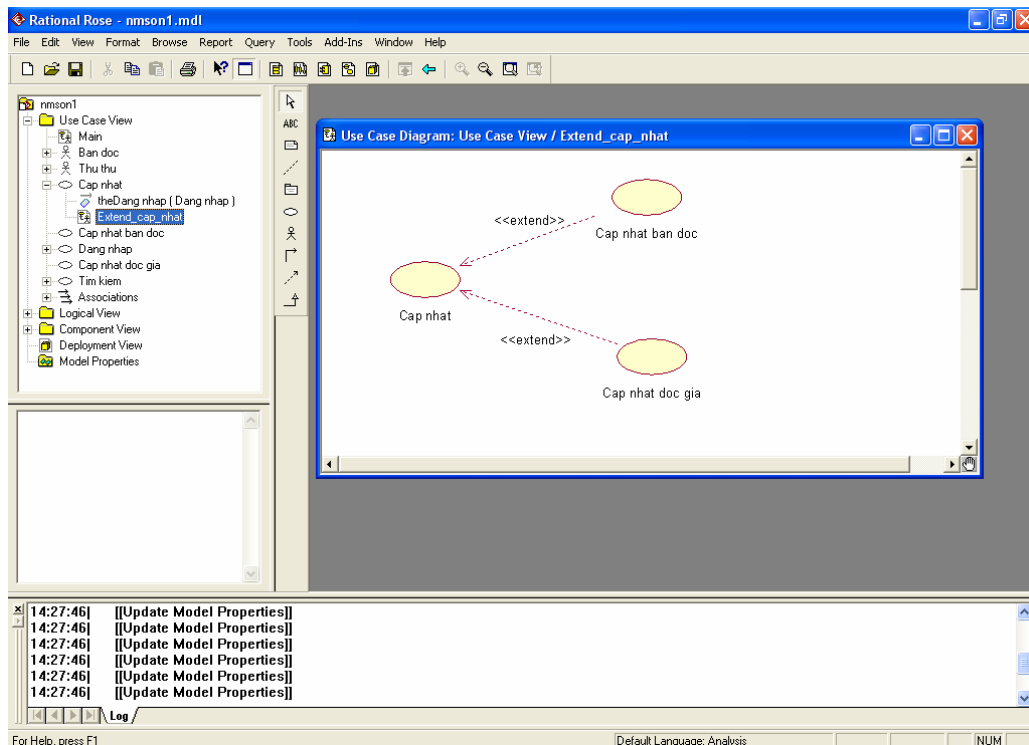
Một trong những nhiệm vụ của bước xây dựng biểu đồ use case là phải phân rã biểu đồ use case. Để thực hiện công việc này, chúng ta làm theo hai bước sau:

- B1. Nhấn chuột phải vào use case tương ứng cần phân rã trong Browser Window và chọn chức năng xây dựng Use Case Diagram mới (Hình 3.9).
- B2. Vẽ biểu đồ use case mức thấp tương tự như biểu đồ use case mức cao.

Khi tạo xong biểu đồ use case mức thấp, biểu đồ này sẽ xuất hiện phía dưới use case tương ứng trong Browser Window (Hình 3.10).

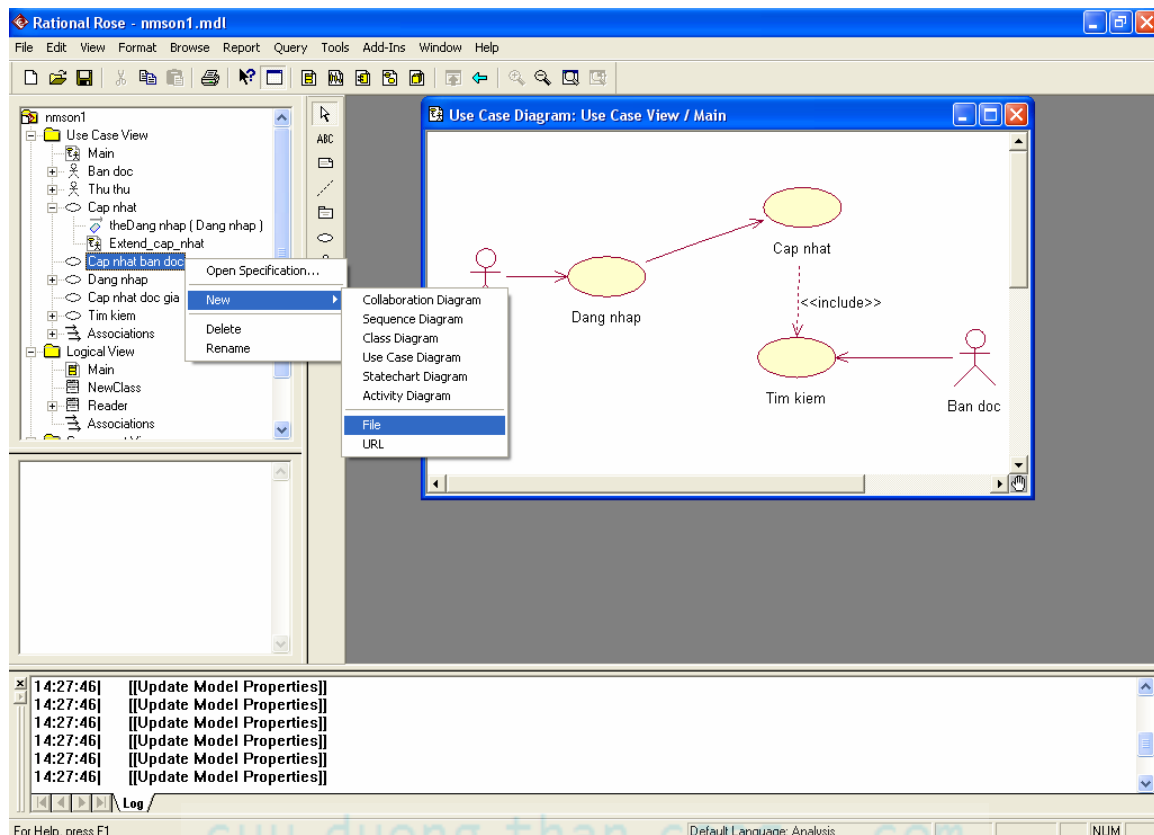


Hình 3.9: Phân rã use case



Hình 3.10: Một sơ đồ use case mức 2

Rational Rose cũng cho phép gắn kèm các file vào trong biểu đồ use case. Chúng ta có thể lợi dụng chức năng này để gắn các file biểu diễn scenario vào trong use case tương ứng (Hình 3.11).



Hình 3.11: Gắn file vào một use case

3.3 MÔ HÌNH LỚP

3.3.1 Vấn đề xác định lớp

Khái niệm cơ bản nhất trong phương pháp hướng đối tượng là khái niệm đối tượng. *Một đối tượng được hiểu là một thực thể có thực hoặc là một thực thể khái niệm. Mỗi đối tượng được mô tả bởi các trạng thái và hành vi cho biết đối tượng đó sẽ hành động như thế nào khi nhận được thông điệp từ các đối tượng khác.*

Hoạt động của hệ thống được thể hiện qua trạng thái của các đối tượng và sự tương tác giữa các đối tượng đó.

Một nhóm đối tượng có chung thuộc tính và phương thức tạo thành một lớp. Vấn đề xác định lớp trở thành một trong những nhiệm vụ cơ bản của phân tích, thiết kế hệ thống hướng đối tượng.

Mối tương tác giữa các đối tượng trong hệ thống sẽ được biểu diễn thông qua mối quan hệ giữa các lớp. Các lớp (bao gồm cả các thuộc tính và phương thức) cùng với các mối quan hệ sẽ tạo thành biểu đồ lớp.

Biểu đồ lớp là một biểu đồ dạng mô hình tĩnh. Một biểu đồ lớp miêu tả hướng nhìn tĩnh của một hệ thống bằng các khái niệm lớp và mối quan hệ giữa chúng với nhau.

Một trong các mục đích của biểu đồ lớp là tạo nền tảng cho các biểu đồ khác, thể hiện các khía cạnh khác của hệ thống (ví dụ như trạng thái của đối tượng hay cộng tác động giữa các đối tượng, được chỉ ra trong các biểu đồ động). Một lớp trong một biểu đồ lớp có thể được thực thi trực tiếp trong một ngôn ngữ hướng đối tượng có hỗ trợ trực tiếp khái niệm lớp. Một biểu đồ lớp chỉ chỉ ra các lớp, nhưng bên cạnh đó còn có một biến tấu hơi khác đi một chút chỉ ra các đối tượng thật sự là các thực thể của các lớp này (biểu đồ đối tượng).

Xác định lớp là một trong những bước khó nhất trong phát triển phần mềm hướng đối tượng. Không có một quy tắc chung nào cho việc xác định lớp trong mọi hệ thống. Kết quả của bước xác định lớp phụ thuộc nhiều vào kinh nghiệm của các nhóm phát triển phần mềm khác nhau. Các phương pháp xác định lớp được đưa ra chỉ mang tính định hướng cho nhóm phát triển chứ không giúp nhóm phát triển tìm ra cụ thể lớp nào là cần thiết hay không cần thiết, đúng hay sai.

Có nhiều phương pháp xác định lớp khác nhau. Ba phương pháp xác định lớp sau đây được xem là phổ biến và nhiều nhóm phát triển đã áp dụng:

- **Phương pháp trích danh từ:** theo phương pháp này, đầu tiên người phát triển hệ thống cần định nghĩa sản phẩm phần mềm bằng một câu, sau đó kết hợp các ràng buộc để phát triển thành một đoạn. Dựa trên đoạn văn mô tả này, người phát triển sẽ lấy ra các danh từ, chia thành các nhóm và đề cử ra các lớp cũng như thuộc tính và phương thức của các lớp đó
- **Phương pháp dùng thẻ ghi CRC** (class responsibility collaboration): dựa trên một số lớp đã phương pháp này sử dụng một thẻ ghi cho mỗi lớp trong đó biểu diễn các thông tin liên quan đến trách nhiệm (responsibility) của lớp đó và các lớp phối hợp với nó (collaboration). Từ thẻ ghi này, người phát triển sẽ tìm ra các lớp khác cần thiết và quan trọng hơn là xác định đầy đủ các thuộc tính, phương thức của từng lớp và mối quan hệ giữa các lớp.

- **Phương pháp xác định lớp từ use case và scenario:** người phát triển nghiên cứu cẩn thận các use case và scenario (cả chuẩn và ngoại lệ) để tìm ra các thành phần đóng vai trò nào đó trong các use case. Các thành phần này sẽ được tập hợp lại và đề cử ra các lớp. Các danh từ xuất hiện trong scenario biểu diễn thông tin cho một thành phần như vậy có thể trở thành các thuộc tính còn các động từ xuất hiện trong mỗi quan hệ giữa các thành phần đó có thể trở thành các phương thức tương ứng trong lớp đó.

Phương pháp xác định lớp từ use case và scenario sẽ được trình bày cụ thể trong các phần tiếp theo của tài liệu.

3.3.2 Xây dựng biểu đồ lớp trong pha phân tích

Biểu đồ lớp là một trong những biểu đồ quan trọng nhất, có tính quyết định trong tiến trình phát triển phần mềm hướng đối tượng. Trong pha phân tích, biểu đồ lớp chưa được xây dựng hoàn chỉnh mà chỉ có các nhiệm vụ chính là:

- Xác định các lớp
- Xác định các thuộc tính và một số phương thức cơ bản (chưa chi tiết các phương thức).
- Bước đầu chỉ ra một số mối quan hệ trong sơ đồ lớp.

Bước 1: Xác định các lớp từ các use case và scenario

Bước này được thực hiện theo nguyên tắc chung như sau:

- Nghiên cứu kỹ tất cả các use case và scenario để tìm ra các danh từ có vai trò nào đó trong các scenario (khởi đầu một tương tác, bắt đầu hay nhận một hành động trong scenario, ...). Các danh từ này sẽ trở thành các lớp ứng cử viên.
- Loại bỏ các lớp ứng cử viên không thích hợp. Các danh từ không thích hợp thuộc vào một trong các trường hợp sau:
 - Lớp dư thừa: do có hai hay nhiều danh từ cùng chỉ một thực thể nên ta chỉ cần giữ lại một từ duy nhất và loại bỏ các từ khác.
 - Danh từ không thích hợp: đó là các danh từ không liên quan đến phạm vi của bài toán.

- Danh từ mô tả những lớp không rõ ràng: đó là các danh từ hoặc không biểu diễn một thực thể cụ thể hoặc các khái niệm không rõ nghĩa.
- Các danh từ chỉ là một vai trò (role) trong mối quan hệ với một lớp khác.
- Các danh từ biểu diễn các công cụ xây dựng phần mềm hoặc các thuật ngữ trong lập trình hay thuật toán (ví dụ stack, list, array, ...).

Xem xét bài toán quản lý thư viện, từ các use case và scenario, ta có thể liệt kê các danh từ như sau: *bạn đọc, tên bạn đọc, địa chỉ bạn đọc, thủ thư, username, password, thẻ mượn, sách, ngày mượn sách, ngày trả sách, số lượng sách ...* Dựa vào tập danh từ này, bước đầu ta có thể xác định một số lớp như: *bạn đọc, thủ thư, thẻ mượn, sách.*

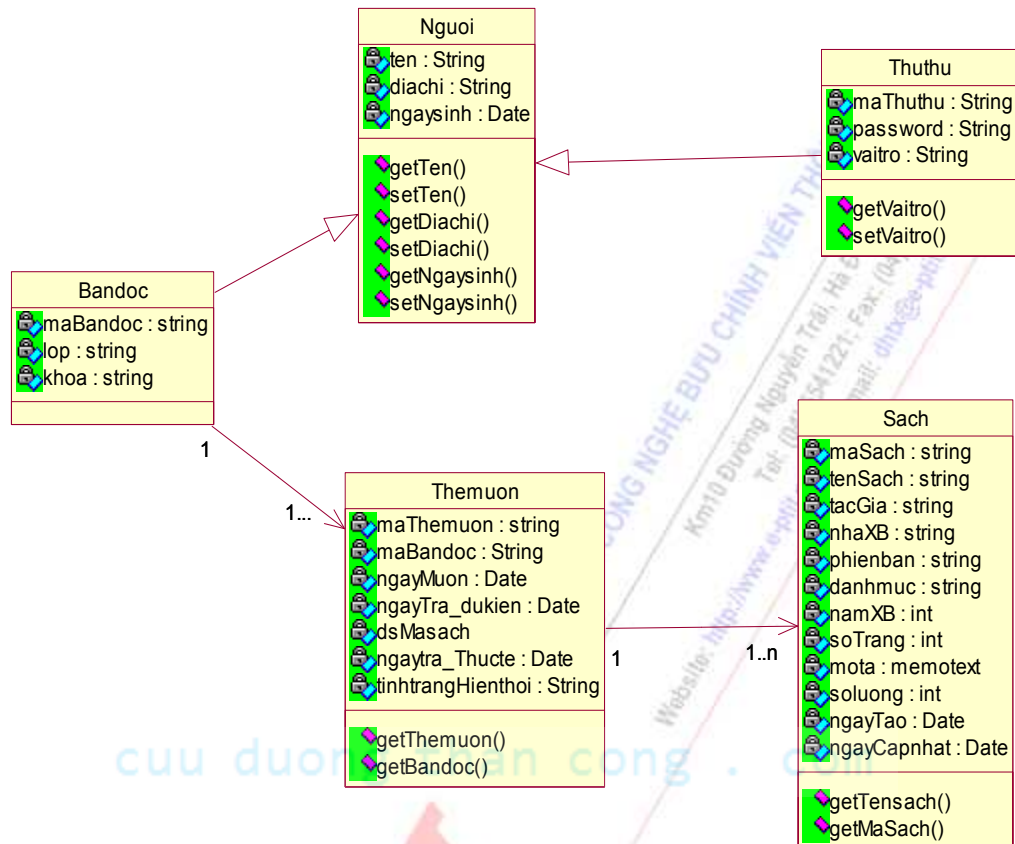
Bước 2: Xác định các thuộc tính và một số phương thức cơ bản

Dựa trên tập các lớp đã được xác định, người phát triển hệ thống tiếp tục nghiên cứu kỹ các use case và scenario và trả lời các câu hỏi sau:

- Với mỗi lớp, những danh từ nào mô tả thông tin của lớp đó. Trả lời câu hỏi này sẽ giúp ta tìm ra các thuộc tính.
- Những thông tin nào của lớp thực sự liên quan đến lĩnh vực quan tâm của hệ thống. Trả lời câu hỏi này giúp ta loại các thuộc tính không cần thiết.
- Những thông tin nào là thông tin riêng của lớp (các thuộc tính private), những thông tin nào có thể chia sẻ trong mối quan hệ với lớp khác (các thuộc tính protected hoặc public).

Tiếp theo, người phát triển hệ thống xem xét các động từ đi kèm với các danh từ biểu diễn lớp trong scenario và xem xét xem các động từ ấy có trở thành các phương thức được hay không. Tuy nhiên, trong pha phân tích, chúng ta chỉ có thể xác định một số phương thức để nhận thấy và cũng chưa cần xác định chi tiết giá trị trả về cũng như các tham số. Các thông tin này sẽ được cụ thể hóa trong pha thiết kế.

Biểu đồ lớp bước đầu của hệ quản lý thư viện được biểu diễn như trong Hình 3.12. Các lớp *Bạn đọc* và *Thủ thư* được kế thừa từ một lớp chung tên là *Người*. Tại một thời điểm, một *bạn đọc* có tương ứng một *Thẻ mượn*. Một *thẻ mượn* có thể cho mượn cùng một lúc một hoặc nhiều cuốn sách.



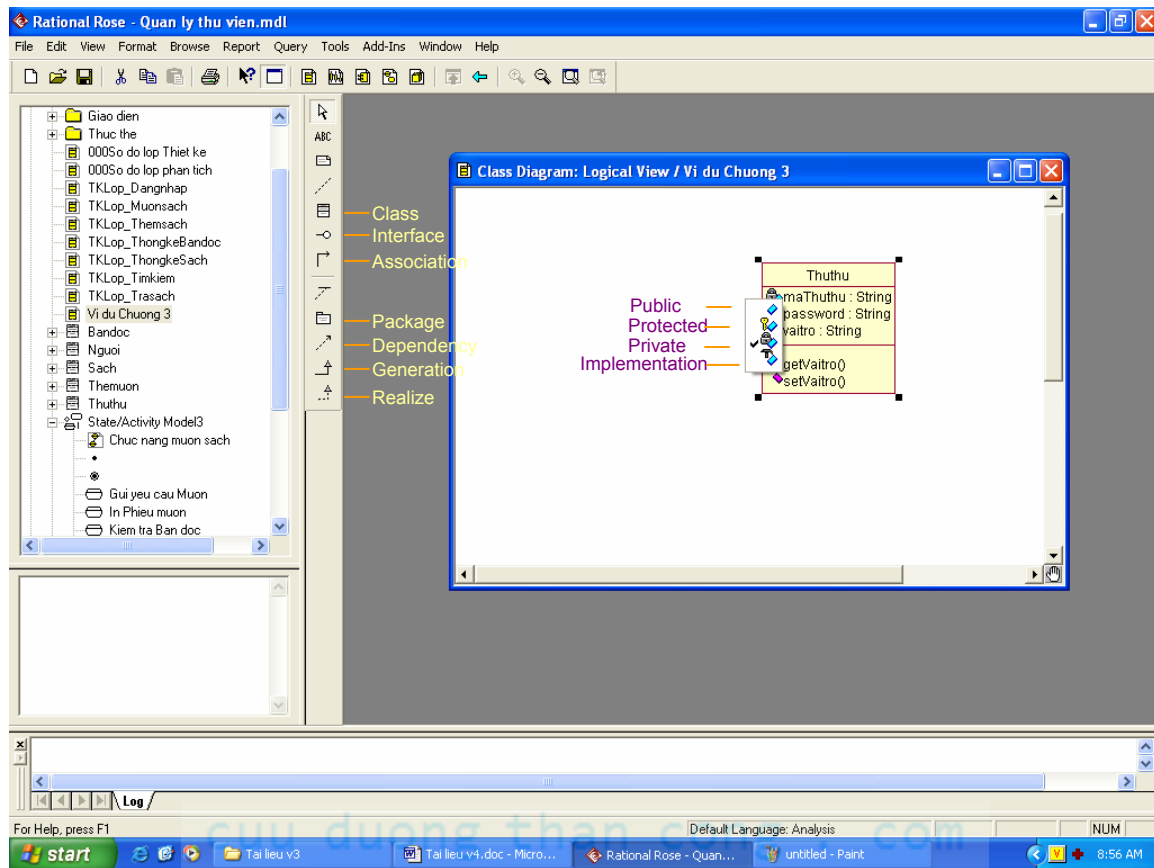
Hình 3.12: Sơ đồ lớp phân tích của hệ thống quản lý thư viện

3.3.3 Biểu diễn biểu đồ lớp trong Rational Rose

Biểu đồ lớp được xây dựng trong Logical View. Các công cụ sử dụng để xây dựng biểu đồ này được thể hiện như trong Hình 3.13. Mỗi lớp trong Rational Rose cũng được chia thành 3 phần: tên lớp, các thuộc tính và các phương thức.

Các bước biểu diễn biểu đồ lớp trong Rational Rose gồm:

1. Biểu diễn các lớp
2. Đặc tả các thuộc tính và phương thức của lớp
3. Đặc tả chi tiết các lớp
4. Biểu diễn các quan hệ



Hình 3.13: Giao diện xây dựng biểu đồ lớp

Bước 1: Biểu diễn các lớp

Đề biểu diễn từng lớp trong biểu đồ lớp, ta thực hiện các bước sau:

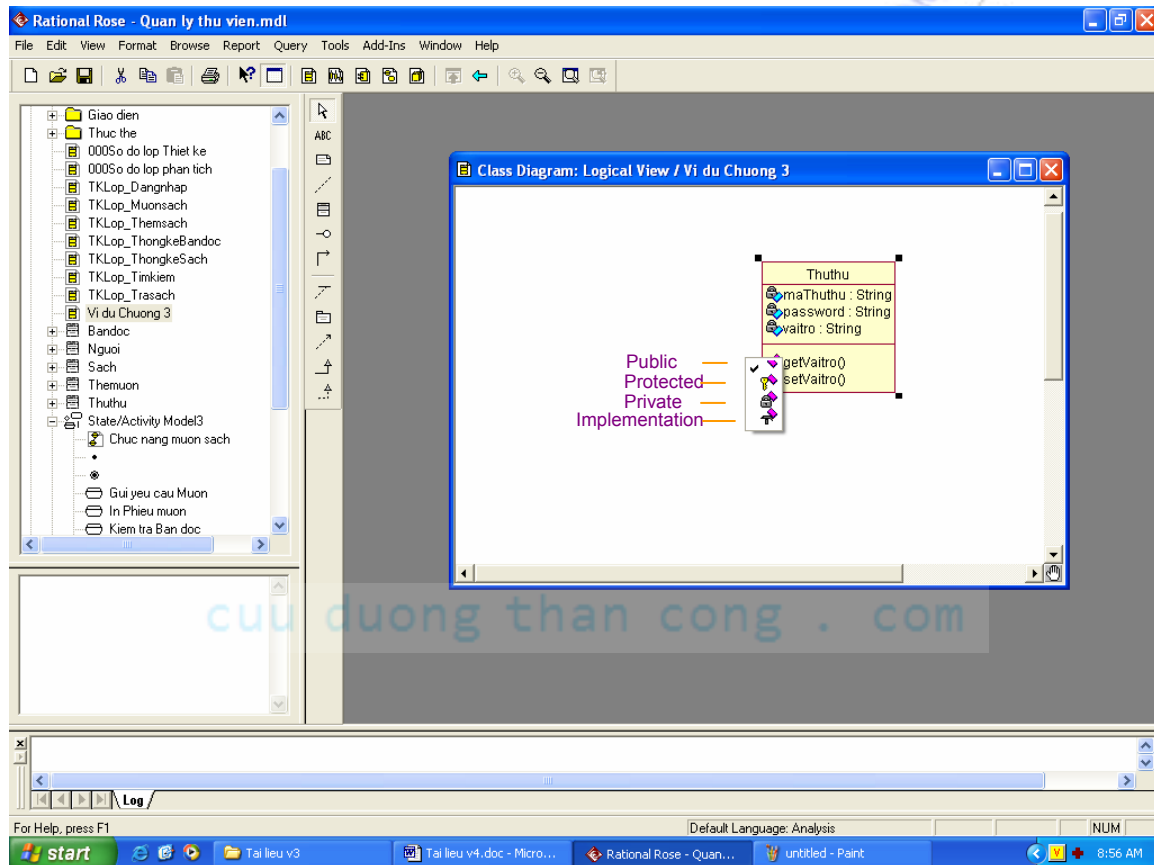
- B1. Chọn công cụ class trong hộp công cụ
- B2. Đưa vào màn hình Class Diagram và đưa vào vị trí thích hợp
- B3. Đặt tên cho lớp.
- B4. Click vào vùng thứ hai trong 3 vùng của biểu diễn lớp và thêm vào tên các thuộc tính.
- B5. Click vào vùng thứ 3 thêm vào các tên các phương thức cho lớp.

Bước 2: Biểu diễn các thuộc tính và phương thức

- B1. Nhấn chuột vào từng thuộc tính và phương thức cần đặc tả
- B2. Chọn phạm vi truy nhập của thuộc tính (và phương thức) (xem hình 3.20)
- B3. Đặc tả kiểu cho thuộc tính

- B4. Đặc tả giá trị trả về và các tham số cho phương thức

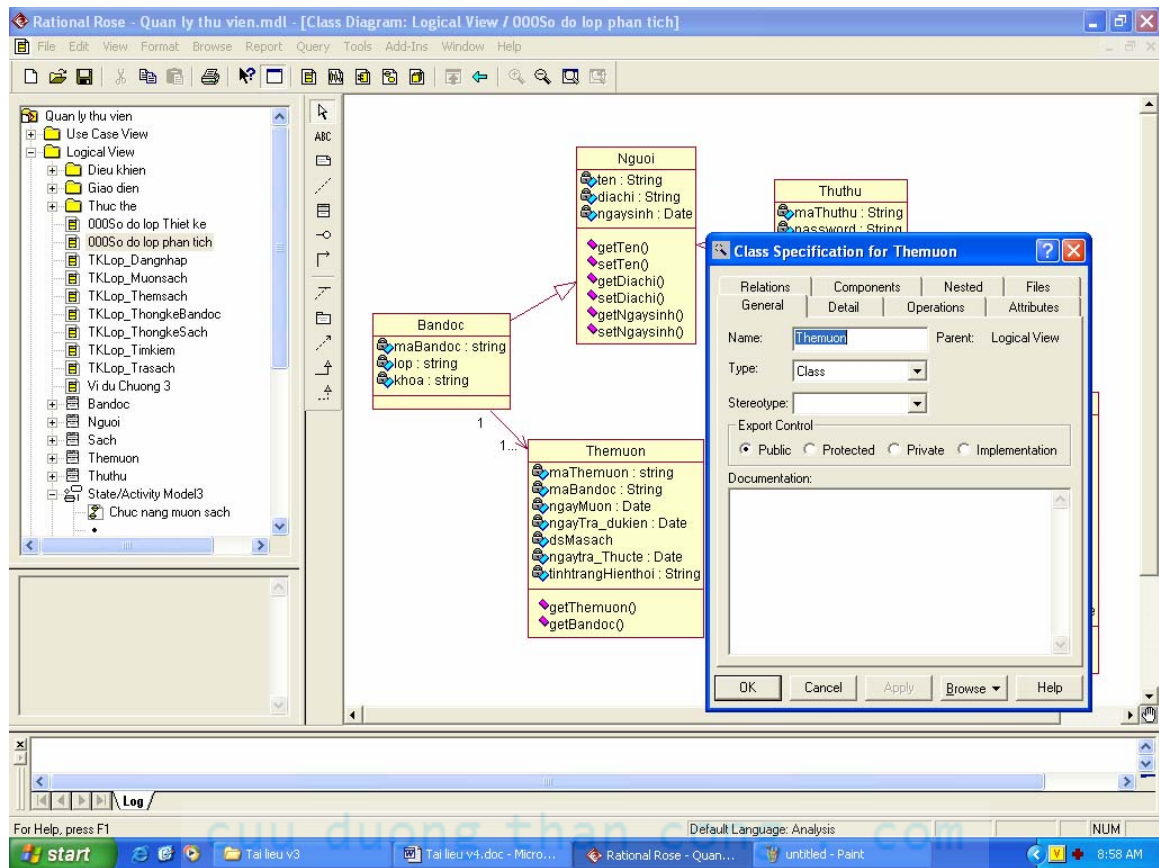
Hình 3.13 đã chỉ ra các phạm vi của một thuộc tính. Phạm vi mặc định của thuộc tính là private. Tương tự, các phạm vi của phương thức được biểu diễn như trong Hình 3.14. Phạm vi mặc định của một phương thức là public.



Hình 3.14: Các phạm vi khác nhau của phương thức

Bước 3: Đặc tả chi tiết một lớp

Cửa sổ đặc tả một lớp được biểu diễn như trong Hình 3.15 với rất nhiều Tab khác nhau. Tab General cung cấp các thông tin chung về lớp (tên, kiểu ...). Các Tab Operations và Attributes cho biết các phương thức và thuộc tính tương ứng của lớp. Tab Relations biểu diễn các mối quan hệ của lớp đó với các lớp khác. Tab Components biểu diễn các thành phần (nếu có) của lớp. Tab Files cho biết các file đính kèm với lớp đó.

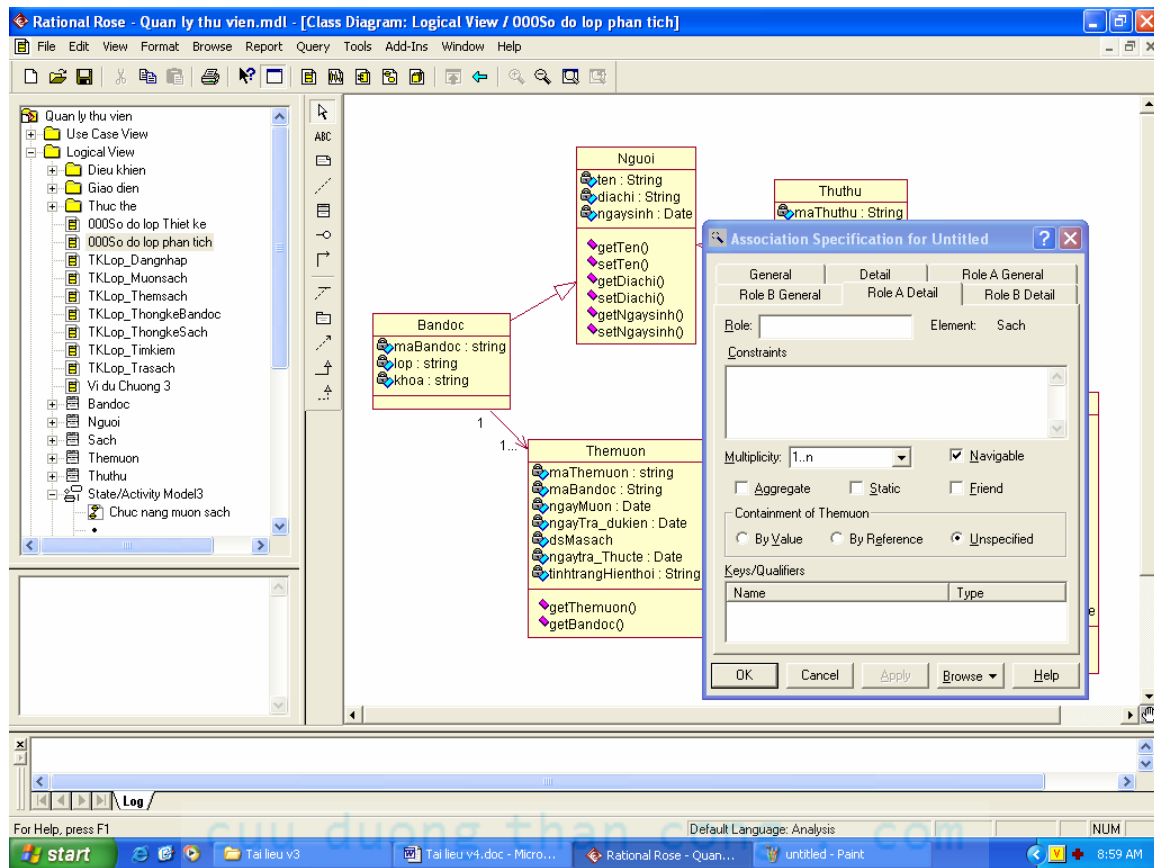


Hình 3.15: Cửa sổ đặc tả của một lớp

Bước 4: Biểu diễn quan hệ giữa các lớp

- B1. Chọn loại quan hệ phù hợp
- B2. Thực hiện kéo thả giữa hai lớp cần xác định quan hệ
- B3. Đặc tả quan hệ.

Cửa sổ đặc tả một mối quan hệ giữa các lớp (kiểu kết hợp: association) được mô tả như trong Hình 3.16. Ngoài các thông tin tương tự như mối quan hệ kết hợp giữa các use case, trong cửa sổ đặc tả mối quan hệ giữa các lớp còn cần chỉ rõ *tính nhiều* (multiplicity) của quan hệ đó. Tính nhiều sẽ được xác định cho cả hai lớp tham gia trong quan hệ. Một lớp sẽ đóng vai trò là Role A, còn lớp kia sẽ đóng vai trò là Role B.



Hình 3.16: Cửa sổ đặc tả một mối quan hệ giữa các lớp

3.4 MÔ HÌNH ĐỘNG DỰA TRÊN BIỂU ĐỒ TRẠNG THÁI

3.4.1 Khái quát về mô hình động

Mô hình động mô tả khía cạnh động trong phần mềm hướng đối tượng. Các tương tác và hành vi *động* trong mô hình UML được chia thành ba dạng:

- Tương tác giữa các đối tượng trong thời gian chạy. Tương tác này được biểu diễn thông qua mô hình tuần tự và/hoặc mô hình cộng tác
- Các hành động tổng quát biểu diễn các tiến trình kinh doanh hoặc tương tác với người dùng. Tương tác này được biểu diễn qua biểu đồ động.
- Các chuyển đổi trạng thái theo thời gian, được biểu diễn qua biểu đồ trạng thái.

Ta sẽ lần lượt xem xét từng dạng mô hình này.

Biểu đồ tuần tự

Mục đích: biểu diễn tương tác giữa những người dùng và những đối tượng bên trong hệ thống. Biểu đồ này cho biết các thông điệp được truyền tuần tự như thế nào theo thời gian. Thứ tự các sự kiện trong biểu đồ tuần tự hoàn toàn tương tự như trong scenario mô tả use case tương ứng.

Biểu diễn: Biểu đồ tuần tự được biểu diễn bởi các đối tượng và message truyền đi giữa các đối tượng đó.

Biểu đồ cộng tác

Mục đích: tương tự như biểu đồ tuần tự, biểu đồ cộng tác biểu diễn tương tác giữa những người dùng và các đối tượng bên trong hệ thống và giữa những đối tượng này với nhau. Biểu đồ cộng tác nhấn mạnh vào mối quan hệ về mặt không gian giữa các đối tượng.

Biểu diễn: Các message trong biểu đồ cộng tác được đánh số theo thứ tự thời gian nhưng khác với biểu đồ tuần tự, biểu đồ cộng tác nhấn mạnh mối quan hệ về mặt không gian giữa các đối tượng trong hệ thống.

Biểu đồ hoạt động

Mục đích: Biểu đồ động được sử dụng để biểu diễn các hoạt động như các luồng công việc hoặc các tiến trình khác nhau trong hệ thống được xây dựng. Biểu đồ động sẽ được biểu diễn thông qua các hoạt động và các chuyển tiếp xảy ra khi chuyển tiếp các hoạt động khi có các điều kiện phù hợp hoặc khi các hoạt động trước được hoàn thành.

Biểu diễn: Biểu đồ động được biểu diễn thông qua các hoạt động, các đồng bộ hay rẽ nhánh và các chuyển tiếp giữa các hoạt động đó. Chi tiết sẽ được trình bày trong pha thiết kế hướng đối tượng.

Biểu đồ trạng thái:

Mục đích: Biểu đồ trạng thái được sử dụng để biểu diễn các trạng thái và sự chuyển tiếp giữa các trạng thái của các đối tượng trong một lớp xác định. Thông thường, mỗi lớp sẽ có một biểu đồ trạng thái (trừ lớp trừu tượng là lớp không có đối tượng).

Biểu diễn: Tương tự như biểu đồ động, biểu đồ trạng thái cũng được biểu diễn dưới dạng máy trạng thái hữu hạn với các trạng thái và sự chuyển tiếp giữa các trạng thái đó. Tuy nhiên, trong biểu đồ trạng thái không có các quá trình đồng bộ và rẽ nhánh như trong biểu đồ động.

Mô hình động trong pha phân tích

Trong pha phân tích, người phát triển chỉ tập trung vào xây dựng biểu đồ trạng thái cho các lớp tìm được trong bước trước. Các biểu đồ tương tác và biểu đồ động chủ yếu được xây dựng trong pha thiết kế, trong đó làm rõ mối quan hệ giữa các đối tượng cũng như các hoạt động của hệ thống để xây dựng biểu đồ lớp chi tiết.

Dựa trên các lớp đã tìm ra trong mô hình lớp, **biểu đồ trạng thái** sẽ được xây dựng cho mỗi lớp. Biểu đồ này sẽ cho biết các trạng thái có thể có của các đối tượng lớp đó và các điều kiện chuyển đổi giữa các trạng thái. Dựa trên các biểu đồ tương tác, người phân tích hệ thống sẽ xem xét lại sơ đồ lớp để bổ sung các thuộc tính cho các lớp cũng như bước đầu xác định được các mối quan hệ giữa các lớp.

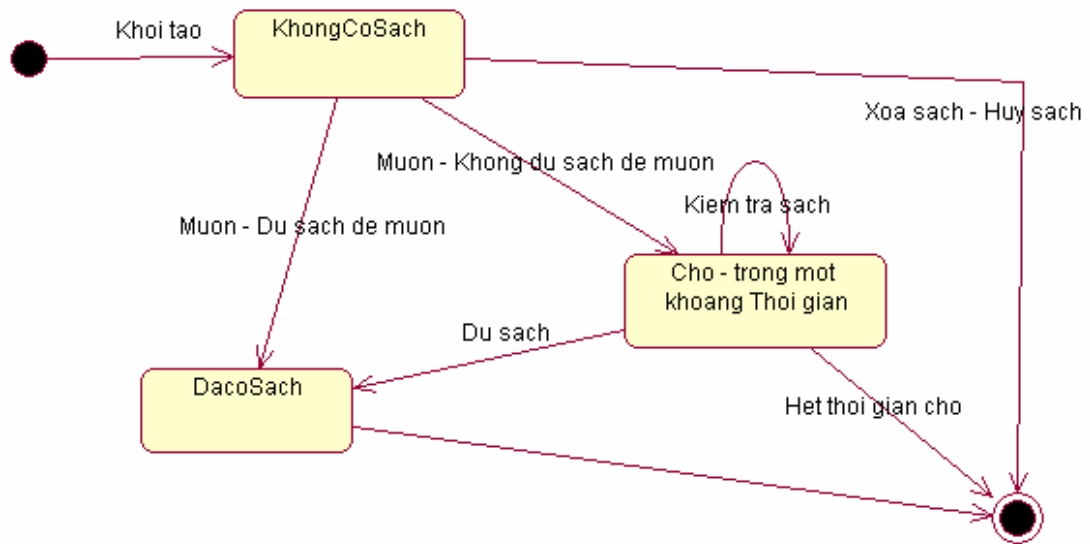
Có hai dạng biểu đồ trạng thái:

- Biểu đồ trạng thái cho một use case: mô tả các trạng thái và chuyển tiếp trạng thái của một đối tượng thuộc một lớp nào đó trong hoạt động của một use case cụ thể.
- Biểu đồ trạng thái hệ thống mô tả tất cả các trạng thái của một đối tượng trong toàn bộ hoạt động của cả hệ thống.

Biểu đồ trạng thái cho mỗi use case dễ xác định hơn vì chúng ta có thể dựa trên biểu đồ use case và các scenario đã có để xác định trạng thái. Còn biểu đồ trạng thái hệ thống chỉ có được khi ta xem xét tất cả các use case.

Tuy nhiên, biểu đồ trạng thái hệ thống sẽ hỗ trợ tốt hơn cho người phân tích trong việc bổ sung các thuộc tính còn thiếu cho biểu đồ lớp và các phương thức của bản thân lớp đó.

Hình 3.17 biểu diễn một biểu đồ trạng thái của lớp **thẻ mượn** trong bài toán quản lý thư viện. Ở đây chỉ xét riêng các trạng thái của lớp này trong chức năng quản lý mượn sách. Trong biểu đồ này ta có các trạng thái chưa có sách, chờ đợi và đã có sách.



Hình 3.17: Biểu đồ trạng thái lớp Thẻ mượn

3.4.3 Xây dựng biểu đồ trạng thái

Biểu đồ trạng thái cho mỗi lớp được xây dựng theo các bước sau:

- Bước 1: Nhận biết các trạng thái và sự kiện
- Bước 2: Xây dựng biểu đồ
- Bước 3: Hiệu chỉnh biểu đồ

Bước 1: Nhận biết các trạng thái và sự kiện

Quá trình phát hiện sự kiện và trạng thái của một đối tượng được thực hiện bằng việc trả lời các câu hỏi sau:

- Một đối tượng có thể có những trạng thái nào?: Hãy liệt kê ra tất cả những trạng thái mà một đối tượng có thể có trong vòng đời của nó.
- Những sự kiện nào có thể xảy ra?: Vì sự kiện gây ra việc thay đổi trạng thái nên nhận ra các sự kiện là một bước quan trọng để nhận diện trạng thái.
- Trạng thái mới sẽ là gì?: Sau khi nhận biết sự kiện, người thiết kế cần xem xét sau khi sự kiện này xảy ra thì trạng thái mới sinh ra sẽ là gì.
- Có những thủ tục ảnh hưởng đến trạng thái của một đối tượng?.

- Những sự kiện và sự chuyển tiếp nào là không thể xảy ra?
- Cái gì khiến cho một đối tượng được tạo ra?: Đối tượng thường được tạo ra do một sự kiện nào đó. Câu hỏi này giúp xác định chuyển tiếp đầu tiên trong biểu đồ trạng thái.
- Cái gì khiến cho một đối tượng bị hủy?: Đối tượng sẽ bị hủy đi khi chúng không còn vai trò gì nữa. Trả lời câu hỏi này sẽ giúp tìm ra các chuyển tiếp cuối cùng trong biểu đồ.

Bước 2: Xây dựng biểu đồ

Sau khi đã trả lời câu hỏi trong bước 1, người phát triển sẽ phải sắp xếp các trạng thái và sự kiện tìm được vào trong một biểu đồ. Xuất phát từ trạng thái khởi đầu, người thiết kế sẽ xác định các trạng thái tiếp theo và biểu diễn các chuyển tiếp giữa các trạng thái đó. Gắn với mỗi chuyển tiếp là một sự kiện. Các sự kiện sẽ được biểu diễn theo cấu trúc chung như sau:

Sự kiện [điều kiện] hoạt động

Trong đó: tên sự kiện được đặt lên đầu, tiếp theo đó là điều kiện (đặt trong 2 dấu ngoặc vuông) của sự kiện đó và cuối cùng là hành động đáp ứng của sự kiện.

Mỗi biểu đồ trạng thái có thể có một hoặc nhiều trạng thái kết thúc. Dựa trên quá trình chuyển tiếp trạng thái, người phát triển sẽ phải xác định chuyển tiếp nào có thể dẫn tới trạng thái kết thúc trong vòng đời đối tượng.

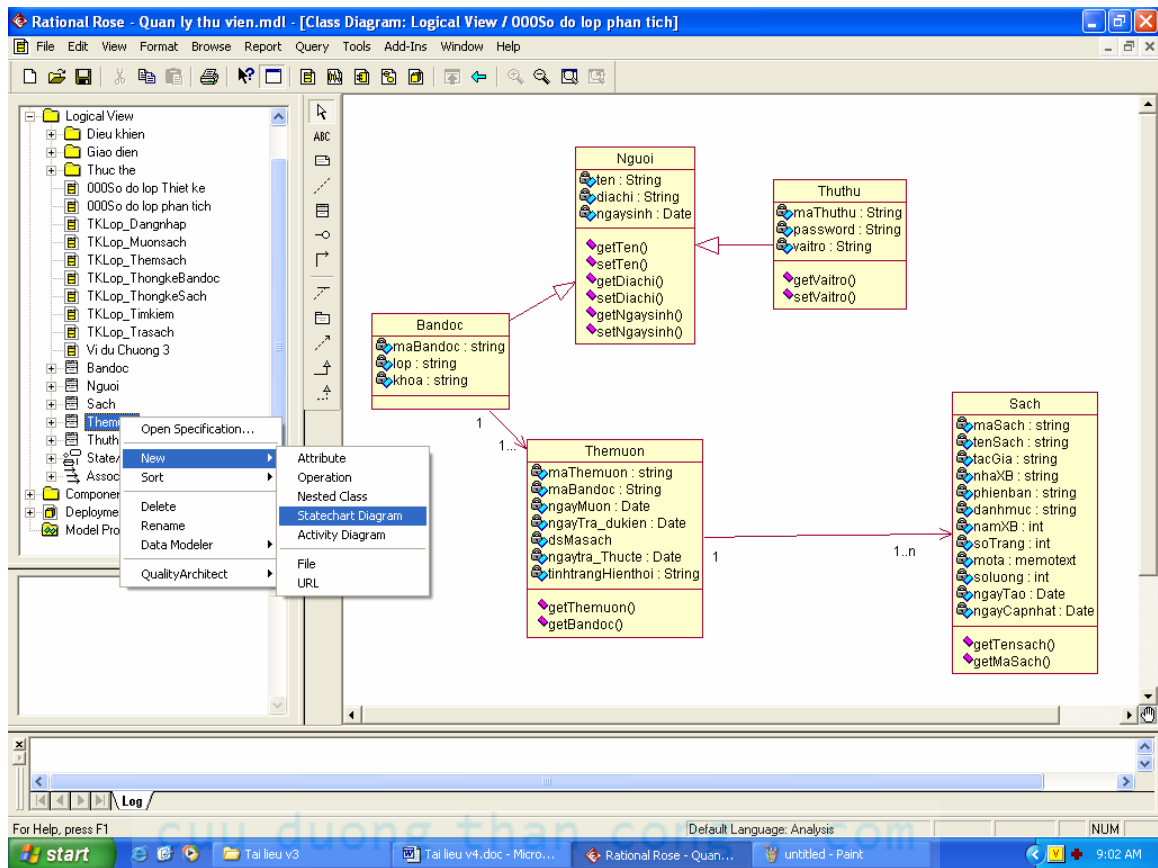
Bước 3: Hiệu chỉnh biểu đồ trạng thái

Người phát triển tiến hành xem xét lại toàn bộ các biểu đồ trạng thái cho từng lớp và sửa đổi lại biểu đồ trạng thái nếu cần thiết. Các biểu đồ trạng thái sẽ được sử dụng để xác định đầy đủ các thuộc tính cho biểu đồ lớp. Vì vậy, bước hiệu chỉnh biểu đồ trạng thái có thể tiếp tục cho đến pha thiết kế.

3.4.3 Biểu diễn biểu đồ trạng thái trong Rational Rose

Biểu đồ trạng thái được xây dựng cho mỗi lớp. Các bước thực hiện thêm biểu đồ trạng thái như sau:

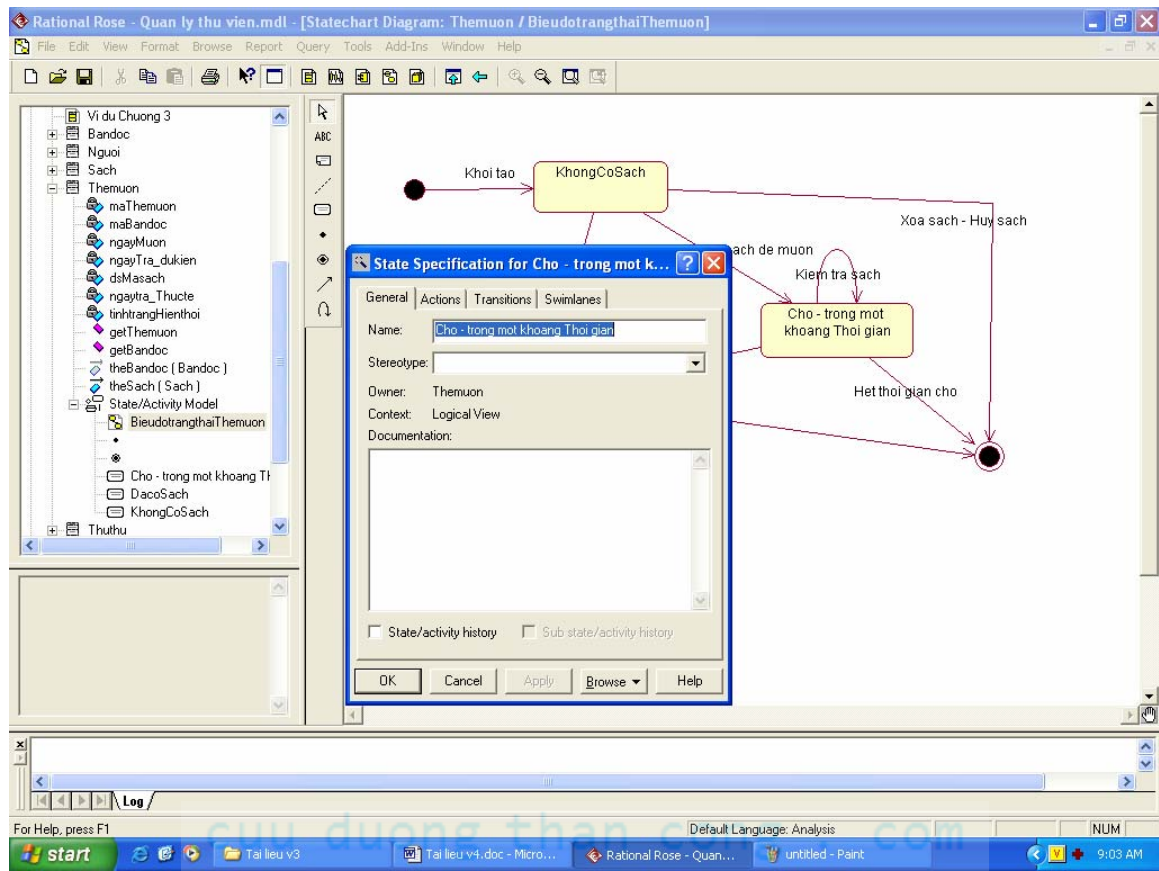
- **Bước 1.** Trong Browser Window, từ một lớp tương ứng, ta nhấn chuột phải và chọn New – Statechart Diagram. Ví dụ trong Hình 3.18 ta lựa chọn xây dựng biểu đồ trạng thái cho lớp Borrow-Card.



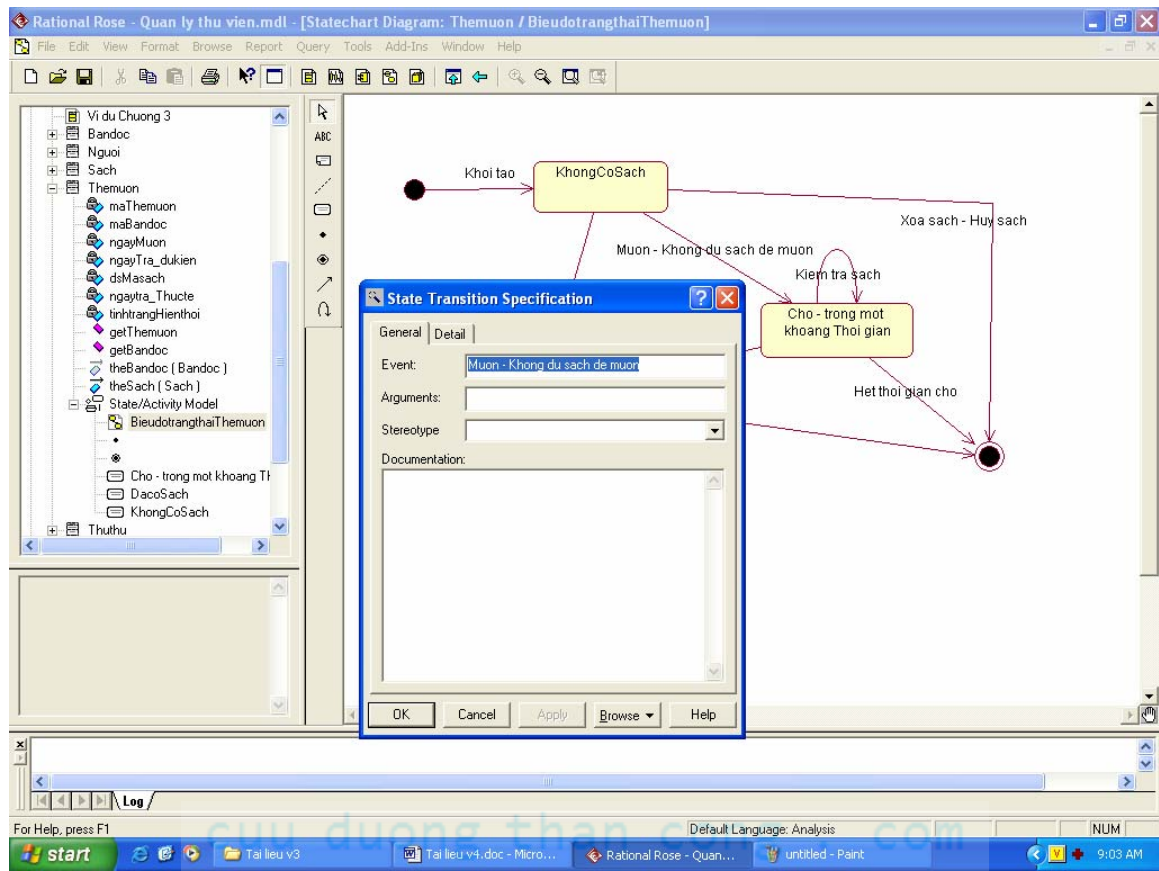
Hình 3.18: Lựa chọn xây dựng biểu đồ trạng thái cho mỗi lớp

- **Bước 2.** Trong cửa sổ xây dựng biểu đồ trạng thái, chọn công cụ state để thêm các trạng thái vào biểu đồ
- **Bước 3.** Đặc tả trạng thái sử dụng của số đặc tả. Hình 3.19 và 3.20 là các cửa sổ đặc tả trạng thái và chuyển tiếp trạng thái. Các thông tin đặc tả này hoàn toàn thống nhất với chuẩn UML đã trình bày trong chương 2 của tài liệu này.
- **Bước 4.** Biểu diễn các quan hệ (chuyển tiếp) trong biểu đồ trạng thái.

CHƯƠNG 3: PHA PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG



Hình 3.19: Đặc tả trạng thái



Hình 3.20: Đặc tả chuyển tiếp trạng thái

TỔNG KẾT CHƯƠNG 3

Chương 3 đã trình bày các bước trong pha Phân tích hướng đối tượng. Một số nội dung sau cần ghi nhớ:

- Pha phân tích hướng đối tượng gồm 3 bước chính được gắn với ba dạng mô hình UML là: mô hình use case, mô hình lớp và mô hình động
- Bước xây dựng mô hình use case gồm 2 việc chính là: xây dựng và phân ra biểu đồ use case và biểu diễn các use case theo dạng kịch bản.
- Bước xây dựng mô hình lớp tiến hành xây dựng biểu đồ lớp. Biểu đồ lớp trong pha phân tích chủ yếu là phát hiện các lớp (dạng lớp thực thể), xác định các thuộc tính và các mối quan hệ đơn giản giữa các lớp đó.
- Bước xây dựng mô hình động trong pha phân tích tập trung vào xây dựng biểu đồ trạng thái mô tả các trạng thái và chuyển tiếp trạng thái của các đối tượng.

tượng của các lớp. Dựa trên biểu đồ trạng thái, người phân tích sẽ có thể hiệu chỉnh lại được biểu đồ lớp, bổ sung các thuộc tính còn thiếu.

- Tài liệu cũng đã đưa ra những hướng dẫn và gợi ý thực hiện cho mỗi bước nhỏ trong pha phân tích.

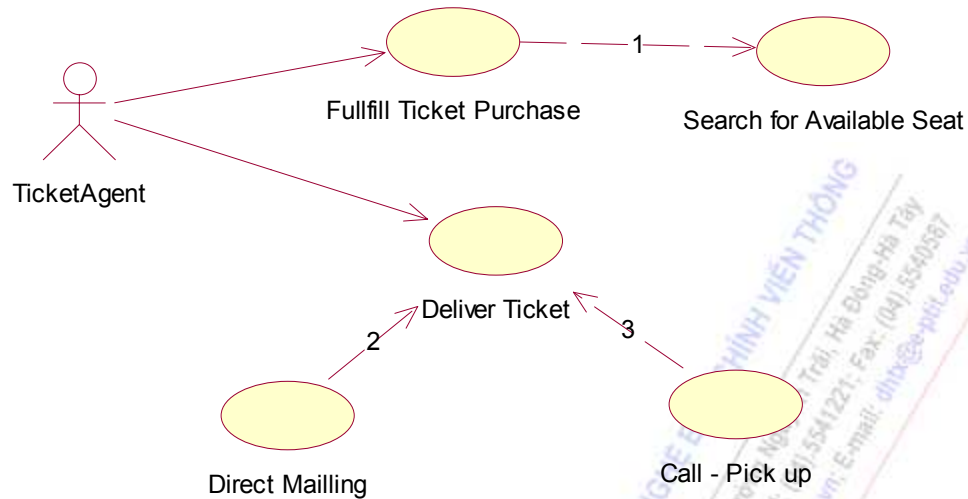
CÂU HỎI – BÀI TẬP

A. CÂU HỎI

1. Biểu đồ use case là gì? Vai trò của biểu đồ use case trong xác định yêu cầu khách hàng?
2. Phân biệt các quan hệ <<include>> và <<extend>> trong biểu đồ use case.
3. Khái niệm kế thừa trong lập trình hướng đối tượng có tương đương với quan hệ khái quát hoá (generalization) giữa các lớp trong UML không. Tại sao
4. Mỗi quan hệ kết hợp 2 chiều là gì? Biểu diễn quan hệ này như thế nào?
5. Phân biệt mỗi quan hệ cộng hợp và quan hệ gộp
6. Khi nào có thể sử dụng mỗi quan hệ thực thi (realization) trong biểu đồ lớp.
7. Biểu đồ trạng thái dùng để làm gì
8. Phân biệt sự khác nhau giữa biểu đồ trạng thái cho một use case và biểu đồ trạng thái hệ thống

B. BÀI TẬP

1. Xem hình vẽ sau:

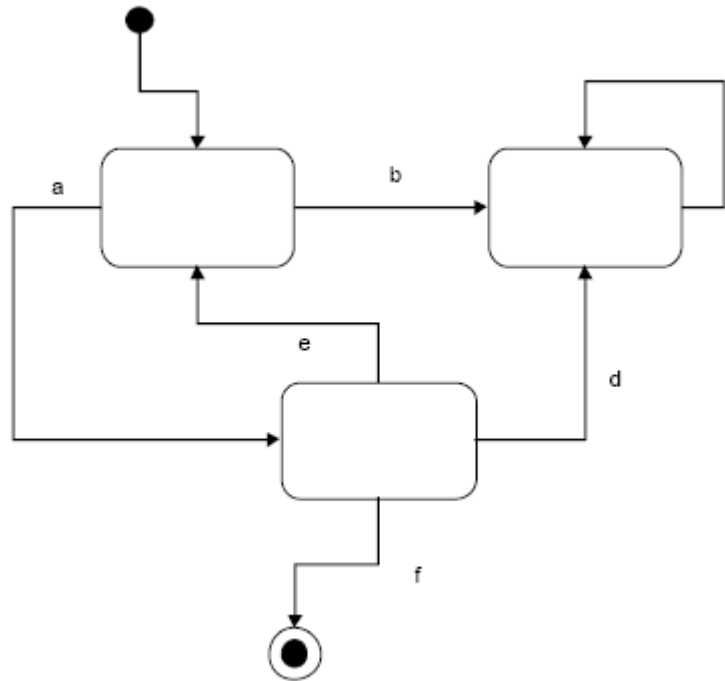


Các nhãn nào sau đây phù hợp với các quan hệ tương ứng với các đường 1, 2, 3?

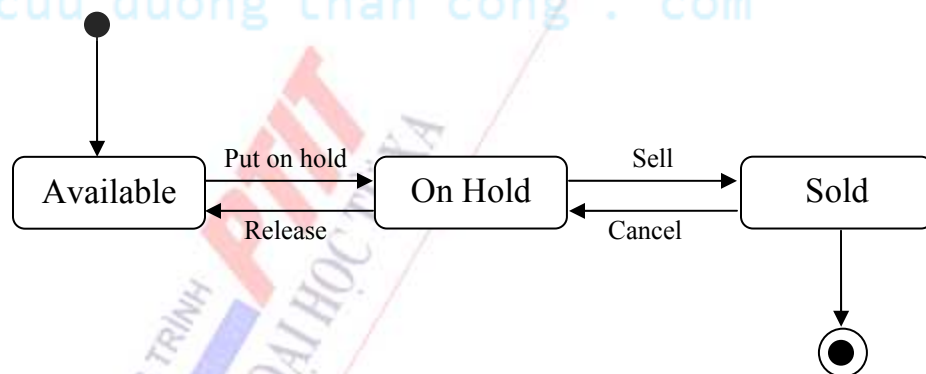
- includes, includes, includes.
 - includes, extends, extends.
 - extends, includes, extends.
 - extends, extends, includes.
 - includes, includes, extends.
2. Xem xét biểu đồ lớp phân tích trong hệ thống thông tin nhân sự. Trong hệ thống này , xét 2 lớp:
- Lớp Employee (Nhân viên) chứa thông tin về mã nhân viên, tên, địa chỉ và ngày sinh nhân viên.
 - Lớp Address (Địa chỉ) chứa thông tin về số nhà, phố, thành phố.

Hãy xác định mối quan hệ giữa lớp Employee và lớp Address

- Trong hệ thống thông tin khách hàng, lớp Bill (Hoá đơn) sinh ra hoá đơn thanh toán cho mỗi khách hàng sử dụng giá trị trả về của hàm tính tổng số tiền calculateAmt() trong lớp Purchase. Xác định mối quan hệ giữa lớp Bill và lớp Purchase.
- Cho một biểu đồ trạng thái (hình vẽ). Đưa ra 3 chuỗi sự kiện (bắt đầu từ trạng thái khởi đầu) làm cho biểu đồ trạng thái bị dẫn tới tình trạng bế tắc. Giải thích.



5. Chuyển tiếp nào trong biểu đồ trạng thái sau là không hợp lệ



- A. Put on hold
 B. Release
 C. Sell
 D. Cancel
6. Ký hiệu *hiển hiện* (*visibility*) nào sau đây chỉ ra một thuộc tính hay phương thức là hiện hữu với các lớp trong cùng một gói.
- A. +
 B. -

- C. #
- D. ~
7. Mỗi quan hệ nào biểu diễn mối quan hệ giữa hai lớp mà sự thay đổi trong phương thức và thuộc tính của lớp này ảnh hưởng đến các thuộc tính và phương thức của lớp kia.
- A. Quan hệ phụ thuộc
 - B. Quan hệ nhân bản (multiplicity)
 - C. Quan hệ thực thi
 - D. Quan hệ kết hợp

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Km10 Đường Nguyễn Trãi, Hà Đông-Hà Tây
Tel: (04) 5541221; Fax: (04) 5540567
Website: <http://www.e-ptit.edu.vn>; E-mail: dhtr@e-ptit.edu.vn

cuuduongthancong.com

CHƯƠNG TRÌNH
PTIT
ĐÀO TẠO ĐẠI HỌC TỪ XA
cuuduongthancong.com

CHƯƠNG 4

PHA THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Chương này trình bày các bước và quá trình thực hiện các bước thiết kế hướng đối tượng. Nội dung cụ thể gồm:

- Tổng quan về thiết kế hướng đối tượng
- Bước xây dựng các biểu đồ tương tác
- Bước xây dựng biểu đồ lớp chi tiết
- Bước xây dựng biểu đồ thành phần và biểu đồ triển khai

4.1 TỔNG QUAN VỀ THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

4.1.1 Vai trò của pha thiết kế

Trong tiến trình phát triển phần mềm nói chung, bước thiết kế hướng đối tượng có vai trò như sau:

- Trả lời câu hỏi “how” thay vì câu hỏi “what” như trong pha phân tích. Mục tiêu của pha thiết kế là phải xác định hệ thống sẽ được xây dựng như thế nào dựa trên kết quả của pha phân tích.
- Đưa ra các phần tử hỗ trợ giúp cấu thành nên một hệ thống hoạt động thực sự.
- Định nghĩa một chiến lược cài đặt cho hệ thống.

Các đặc trưng của pha thiết kế hướng đối tượng bao gồm:

- Mô hình hóa chi tiết hệ thống dựa trên các lớp, các đối tượng trong miền ứng dụng của hệ thống đó.
- Thiết kế dựa trên chiến lược trừu tượng hoá phân cấp dữ liệu (hierachical data abstraction) trong đó các thành phần sẽ được thiết kế từ các lớp, đối tượng, các module và các tiến trình.
- Các phương thức thường được thiết kế trong mối quan hệ với các đối tượng xác định hoặc một lớp các đối tượng đó.

4.1.2 Các bước thiết kế hướng đối tượng

Dựa trên các kết quả của pha phân tích, pha thiết kế hướng đối tượng được chia thành các bước như sau:

- Xây dựng các biểu đồ tương tác, bao gồm biểu đồ tuần tự và biểu đồ cộng tác.
- Xây dựng biểu đồ lớp chi tiết: thực hiện hoàn chỉnh sơ đồ lớp, xác định và biểu diễn đầy đủ các phương thức cho từng lớp, xác định mối quan hệ giữa các lớp.
- Thiết kế chi tiết: xây dựng các biểu đồ động cho các phương thức phức tạp trong các lớp và xây dựng bảng thiết kế chi tiết cũng như kế hoạch cài đặt và tích hợp.
- Xây dựng biểu đồ thành phần và biểu đồ triển khai hệ thống
- Phát sinh mã, chuẩn bị cho cài đặt hệ thống

Các bước này sẽ được trình bày trong các phần sau của tài liệu này.

3.2 CÁC BIỂU ĐỒ TƯƠNG TÁC

Như đã trình bày trong phần 3.4, các biểu đồ tương tác biểu diễn các tương tác giữa các tác nhân bên ngoài và các đối tượng bên trong hệ thống cũng như tương tác giữa các đối tượng bên trong hệ thống đó. Biểu đồ tương tác có hai dạng là:

- *Biểu đồ tuần tự (sequence diagram)* nhấn mạnh thứ tự thực hiện các tương tác
- *Biểu đồ cộng tác (collaboration diagram)* nhấn mạnh đến mối quan hệ và sự bố trí giữa các đối tượng trong tương tác đó.

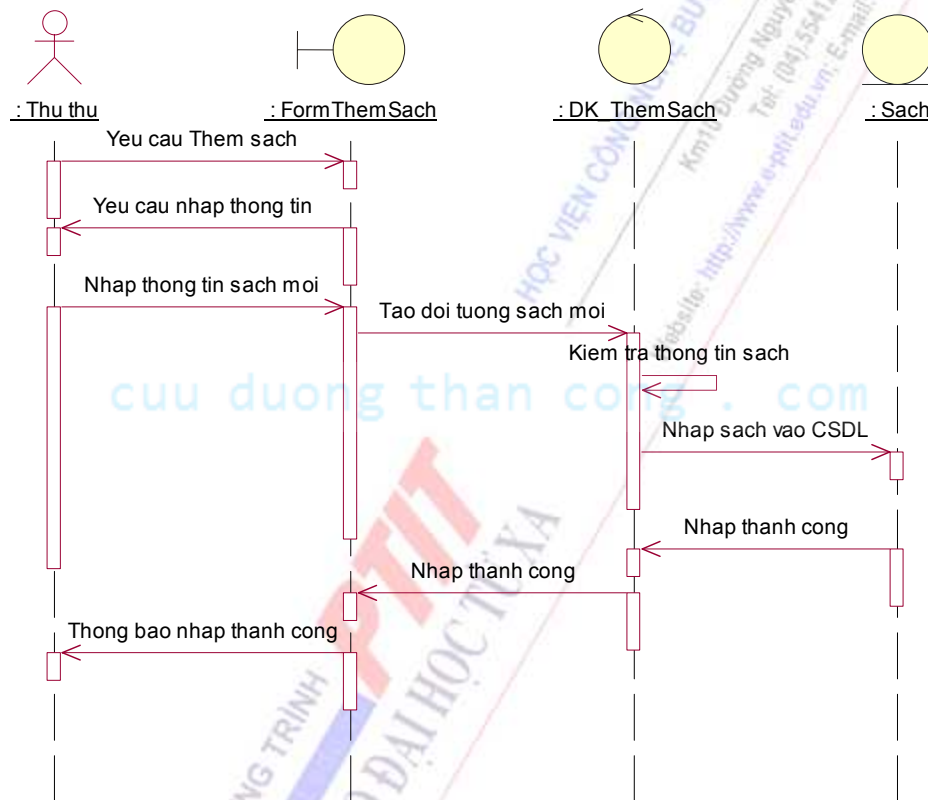
Tùy vào yêu cầu của hệ thống cụ thể, người phát triển hệ thống có thể lựa chọn một trong hai biểu đồ hoặc sử dụng cả hai biểu đồ. Trong phần này, tài liệu sẽ tập trung trình bày các phần tử mô hình UML sử dụng trong các biểu đồ tương tác và cách thức xây dựng các biểu đồ tương tác đó.

4.2.2 Xây dựng biểu đồ tuần tự

Thông thường, các biểu đồ tuần tự được gắn với các use case. Các message trong biểu đồ tuần tự sẽ biểu diễn lại thứ tự các sự kiện trong scenario của use case đó (cả chuẩn và ngoại lệ).

Hình 4.1 biểu diễn một ví dụ về biểu đồ tuần tự đơn giản mô tả chức năng thêm sách được xây dựng nên từ scenario đã trình bày trong chương trước. Trong

chức năng thêm sách, các đối tượng tham gia gồm: *Thủ thư*, *Form Thêm sách*, *đối tượng điều khiển Thêm sách* và *đối tượng Sách*. Thứ tự thực hiện message trong biểu đồ là theo chiều từ trên xuống dưới. Nhìn vào một biểu đồ tuần tự như vậy ta có thể thấy được ngay thứ tự thực hiện các hành động của một đối tượng trong chức năng (use case) đang xem xét. Biểu đồ tuần tự này mô tả lại kịch bản (scenario) của use case Thêm sách nhưng dựa trên các đối tượng của các lớp đã xác định trong pha phân tích. Với mỗi chức năng, thông thường chúng ta sẽ thêm một lớp giao diện (lớp Form) và một lớp điều khiển cho chức năng đó.



Hình 4.1: Biểu đồ tuần tự cho use case Thêm sách

Một số chú ý khi vẽ biểu đồ tuần tự:

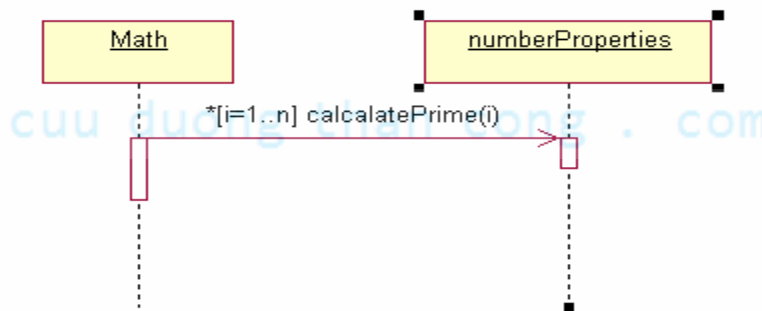
- Sự kiện được biểu diễn kèm theo các message nằm ngang.
- Đối tượng luôn gắn với các đường life line dọc theo biểu đồ. Điểm kết thúc của đường life line này đánh dấu thời điểm hủy đối tượng hoặc khi tương tác đã kết thúc.

- Trục thời gian được quy định từ trên xuống dưới. Các message ở trên sẽ xảy ra trước các message ở phía dưới.
- Trong biểu đồ tuần tự có thể xuất hiện các message từ một đối tượng đến chính bản thân nó.

Tiếp theo, ta xem xét một số vấn đề phức tạp hơn khi xây dựng biểu đồ tuần tự gồm: biểu diễn các message lặp, sử dụng các message tạo và huỷ và phân nhánh các đối tượng.

- **Biểu diễn các message lặp**

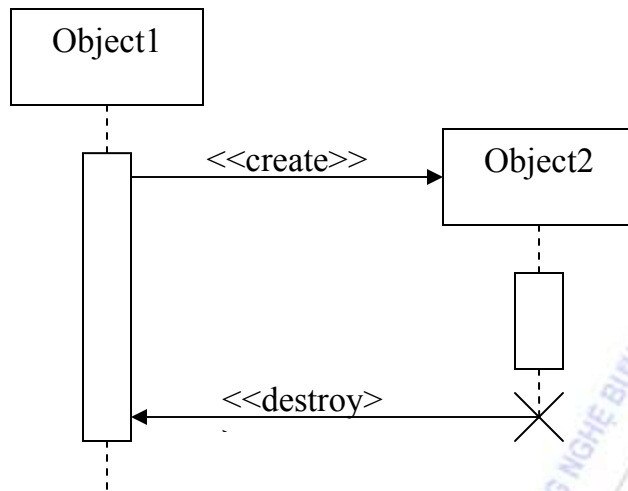
Trong biểu đồ tuần tự, có một số trường hợp ta cần biểu diễn các message được gửi theo vòng lặp (nhiều lần liên tiếp) giữa hai đối tượng. Khi đó, ta bổ sung thêm cấu trúc: $*[i=1..n]$ vào trước message; với i là biến điều khiển lặp, n là số lần lặp. Xem xét ví dụ trong Hình 4.2.



Hình 4.2: Biểu diễn message lặp

- **Sử dụng các message tạo và huỷ**

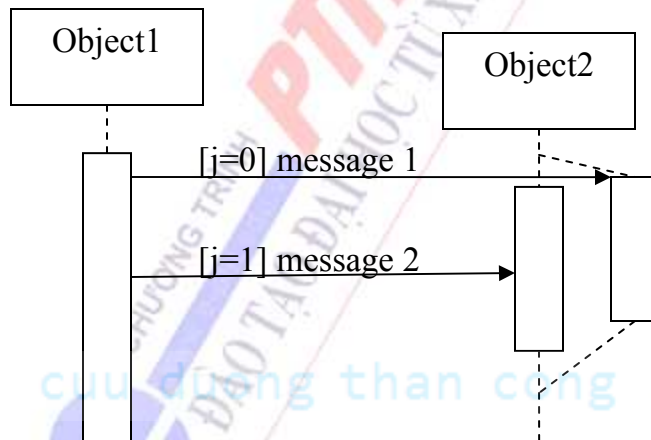
Thông thường, các message trong biểu đồ tuần tự được gửi và nhận từ các đối tượng đã tồn tại. Tuy nhiên, trong trường hợp các đối tượng tham gia tương tác thuộc về các lớp có quan hệ phụ thuộc thì ta phải sử dụng các message tạo và huỷ. Các message tạo và huỷ được biểu diễn trong ví dụ Hình 4.3.



Hình 4.3: Sử dụng message tạo và hủy

- **Biểu diễn phân nhánh các đối tượng**

Trong trường hợp ứng với các giá trị khác nhau của tham số, đối tượng hoạt động khác nhau thì chúng ta dùng cách biểu diễn phân nhánh đối tượng. Xem ví dụ Hình 4.4.



Hình 4.4: Phân nhánh các đối tượng

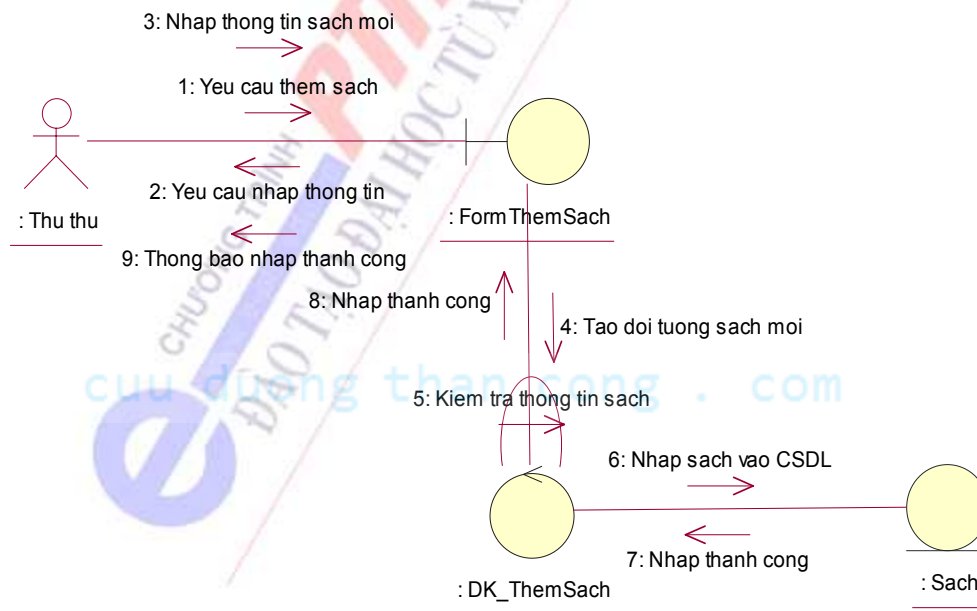
4.2.3 Xây dựng biểu đồ cộng tác

Biểu đồ cộng tác cũng có các message với nội dung tương tự như trong biểu đồ tuần tự. Tuy nhiên, các đối tượng được đặt một cách tự do trong không gian của biểu đồ và không có đường life line cho mỗi đối tượng. Các message được đánh số thể hiện thứ tự thời gian.

Một số chú ý khi xây dựng biểu đồ cộng tác:

- Giữa hai đối tượng có thể có nhiều message. Các message này sẽ cùng được biểu diễn trong không gian giữa hai đối tượng, kèm theo số thứ tự của nó.
- Trong biểu đồ cộng tác cũng có thể có các message từ một đối tượng đến bản thân nó. Message này sẽ biểu diễn bởi một đường vô hướng xuất phát và kết thúc trên đối tượng đó.

Hình 4.5 mô tả một biểu đồ tương tác kiểu cộng tác (collaboration diagram). Nội dung biểu đồ này hoàn toàn tương tự như trong biểu đồ tuần tự trong Hình 4.1. Nếu như biểu đồ tuần tự nhấn mạnh đến thứ tự các message thì biểu đồ cộng tác lại nhấn mạnh đến quan hệ giữa các đối tượng. Do đó, trong biểu đồ cộng tác không có các đường life line. Các đối tượng sẽ được bố trí tự do trong biểu đồ theo hình dung của người thiết kế.



Hình 4.5: Biểu đồ cộng tác cho use case Thêm sách

Từ Hình 4.5, ta thấy các thành phần cơ bản của một biểu đồ cộng tác là:

- *Các đối tượng (object)*: trong biểu đồ cộng tác, các đối tượng vẫn được biểu diễn với dạng hoàn toàn tương tự như trong biểu đồ tuần tự nhưng không có đường life line phía dưới.
- *Các message có đánh số thứ tự*: giữa các đối tượng có tương tác trong biểu đồ cộng tác, người ta vẽ các đường liên kết vô hướng. Các message sẽ được biểu diễn phía trên đường liên kết đó và mỗi message sẽ được đánh số thứ tự tương ứng với thứ tự xuất hiện về mặt thời gian của message đó.

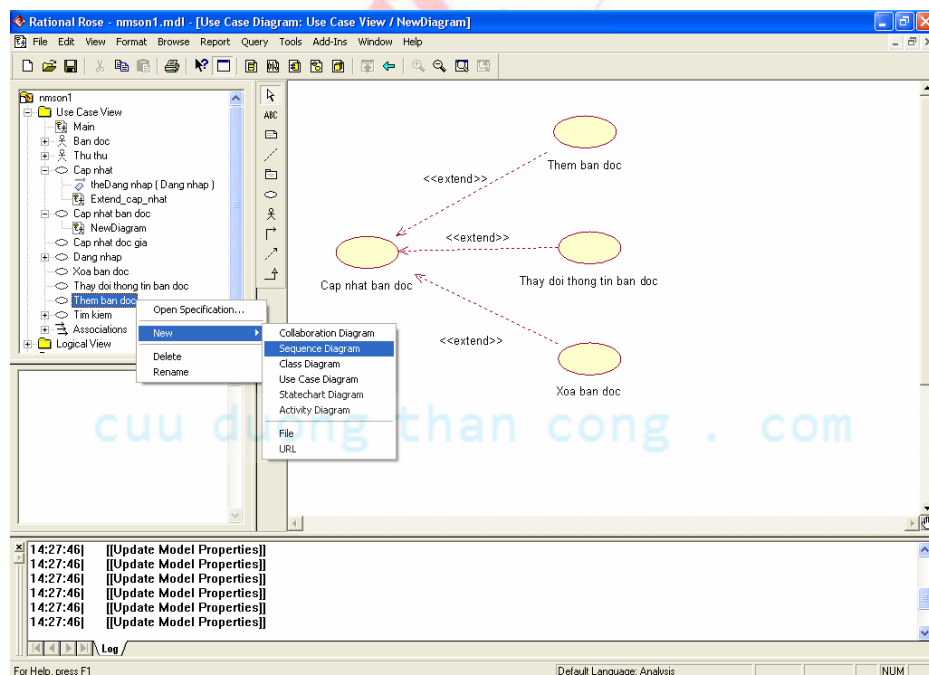
4.2.4 Biểu diễn các biểu đồ tương tác trong Rational Rose

Các biểu đồ tương tác có thể được xây dựng từ các use case tương ứng. Trong trường hợp đó, người sử dụng có thể nhấn chuột phải vào use case tương ứng và lựa chọn New rồi đến tên loại biểu đồ cần xây dựng.

a) Xây dựng biểu đồ tuần tự

- **Bước 1:** Chọn use case cần xây dựng biểu đồ tuần tự. Click chuột phải, chọn New – Sequence Diagram.

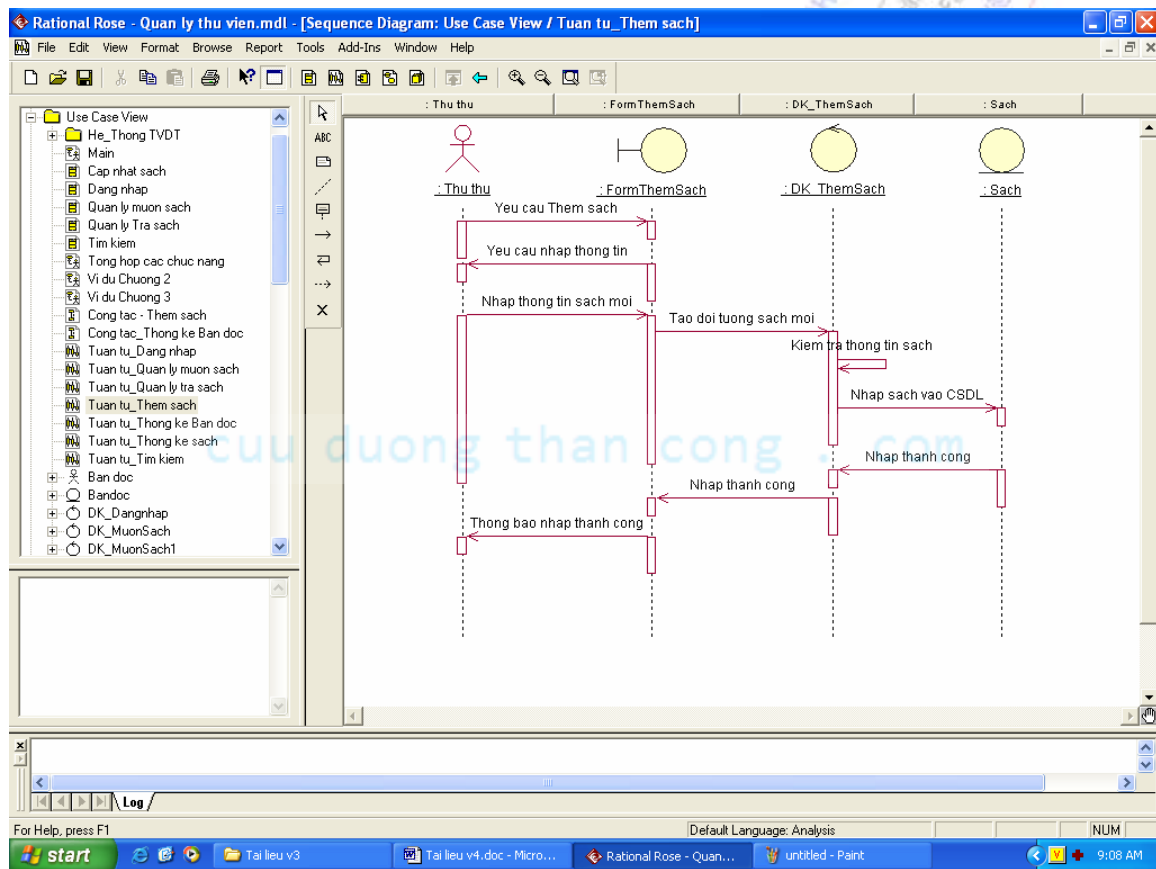
Hình 4.6 là lựa chọn xây dựng biểu đồ tuần tự, Hình 4.7 là lựa chọn xây dựng biểu đồ cộng tác.



Hình 4.6: Lựa chọn xây dựng biểu đồ tuần tự cho mỗi use case

- **Bước 2:** Thêm các đối tượng vào biểu đồ tuần tự. Chọn ký hiệu đối tượng trong hộp công cụ và kéo vào cửa sổ biểu đồ.
- **Bước 3:** Thêm các message.
- **Bước 4:** Đặc tả các message: đặt tên hoặc mô tả dưới dạng hàm.

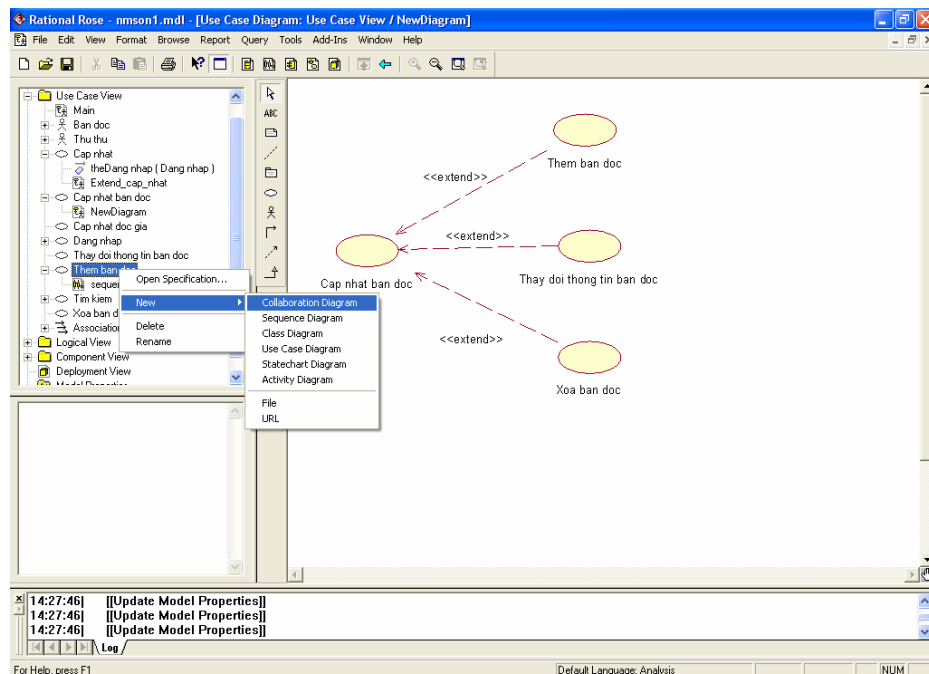
Một biểu đồ tuần tự có dạng như trong Hình 4.7. Hộp công cụ trong các biểu đồ tuần tự và cộng tác là các đối tượng và các dạng mũi tên biểu diễn các dạng message khác nhau.



Hình 4.7: Biểu diễn một biểu đồ tuần tự

b) Xây dựng biểu đồ cộng tác

- **Bước 1:** Lựa chọn use case cần xây dựng biểu đồ cộng tác. Xem hình 4.8.



Hình 4.8: Lựa chọn xây dựng biểu đồ cộng tác cho mỗi use case

- **Bước 2:** Biểu diễn các đối tượng trong không gian biểu đồ.
- **Bước 3:** Biểu diễn các message trong biểu đồ cộng tác. Mỗi message kèm theo số thứ tự của message đó.

4.3 BIỂU ĐỒ LỚP CHI TIẾT

4.3.1 Xác định các phương thức cho mỗi lớp

Bước đầu tiên trong xây dựng biểu đồ lớp chi tiết là xác định phương thức cho các lớp. Dựa trên các biểu đồ tương tác vừa xây dựng, quá trình xác định các phương thức được thực hiện theo các nguyên tắc sau:

- Xem xét các message trong các biểu đồ tương tác để xác định hành động tương ứng với message đó thuộc trách nhiệm của lớp nào.
- Các phương thức nào cần thiết để chuyển đổi các trạng thái trong biểu đồ trạng thái của một lớp.
- Xác định xem với mỗi lớp, lớp đó cần các hàm tạo và hàm hủy hay không.

Sau khi đã xác định đầy đủ các phương thức, công việc tiếp theo là phải xác định chi tiết giá trị trả về và các tham số liên quan với mỗi phương thức.

4.3.2 Xác định mối quan hệ giữa các lớp

Trong bước này, người phát triển hệ thống phải xác định đầy đủ mối quan hệ giữa các lớp và các vấn đề liên quan đến các mối quan hệ đó. Những công việc cụ thể phải thực hiện trong bước này là:

- Xác định cụ thể dạng của quan hệ giữa các lớp
- Xác định số lượng trong mỗi mối quan hệ

Bước 1: Xác định cụ thể dạng của quan hệ giữa các lớp

Như đã trình bày trong chương 2, có bốn dạng quan hệ cơ bản trong sơ đồ lớp là: quan hệ kế thừa, quan hệ kiểu kết hợp, quan hệ gộp và quan hệ phụ thuộc. Trong bước này, người phát triển phải tìm ra các quan hệ giữa các lớp và xác định cụ thể quan hệ đó thuộc dạng nào. Nếu như các danh từ giúp chúng ta tìm ra lớp thì các động từ trong các scenario sẽ giúp chúng ta tìm ra các quan hệ. Các quan hệ sẽ được phân loại dựa trên nguyên tắc sau:

- Hai lớp có mối quan hệ kiểu kết hợp với nhau nếu các động từ trong tương tác giữa các lớp biểu hiện một sự thay thế, đại diện, sự bao hàm, sự giao tiếp, sự sở hữu hay yêu cầu thỏa mãn điều kiện nào đó.
- Quan hệ gộp thường được biểu diễn qua các động từ như: được tạo thành từ, bao gồm...
- Hai lớp có quan hệ kế thừa nếu một lớp này là khái quát hoá (trừu tượng hoá) của lớp kia.
- Hai lớp có quan hệ phụ thuộc nếu hoạt động của lớp này quyết định lớp kia.

Trong hệ thống quản lý thư viện, mối quan hệ giữa các lớp Reader và Librarian với lớp Person là mối quan hệ kế thừa. Quan hệ giữa các lớp Reader và Book với lớp Borrow_Card là quan hệ kết hợp vì ta có thể biểu diễn các mối quan hệ này thông qua các câu như: *để mượn sách, bạn đọc cần dùng thẻ mượn, mỗi thẻ mượn có thể mượn được một hay nhiều cuốn sách.*

Bước 2: Xác định số lượng (multiplicities) trong mỗi mối quan hệ

Mỗi mối quan hệ trong sơ đồ lớp có thể có số lượng tương ứng ở đầu mỗi lớp. Số lượng này xác định số thể hiện có thể có của lớp đó trong mỗi quan hệ với lớp kia. Các kiểu biểu diễn số lượng (multiplicities) được biểu diễn như trong Bảng 4.2.

Multiplicities	Ý nghĩa
0..1	Không có hoặc có 1 thể hiện. Tương tự $n..m$ sẽ thể hiện có từ n đến m thể hiện.
0..* hoặc *	Không giới hạn số thể hiện của lớp (gồm cả giá trị 0).
1	Có chính xác 1 thể hiện
1..n	Có ít nhất một thể hiện

Bảng 4.2: Các kiểu biểu diễn số lượng trong biểu đồ lớp

Xem xét sơ đồ lớp phân tích đã trình bày trong chương 3, trong quan hệ giữa lớp Reader và lớp Borrow_card, mỗi bạn đọc có thể có một hoặc nhiều thẻ mượn hoặc cũng có thể không có thẻ mượn nào. Tuy nhiên, một thẻ mượn phải tương ứng với một bạn đọc nào đó. Như vậy, số lượng trong quan hệ này sẽ là: 1 ở phía Reader và 1..n ở phía Borrow_card.

4.3.4 Hoàn chỉnh biểu đồ lớp chi tiết

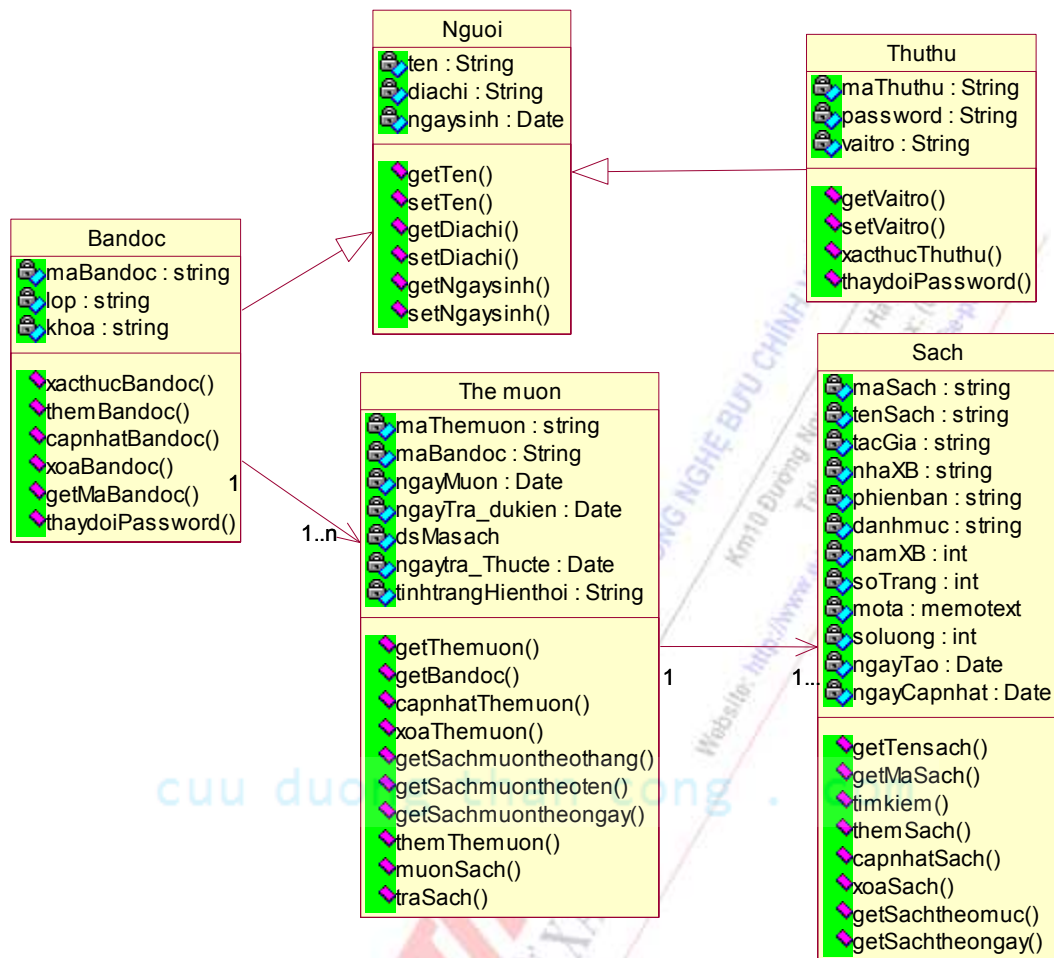
Đây là bước cuối cùng của sơ đồ lớp. Trong bước này, người thiết kế phải thực hiện các công việc sau:

- **Bổ sung các lớp còn thiếu.** Trong pha phân tích, chúng ta đã xác định được các lớp thực thể. Ở pha thiết kế, chúng ta cần tiếp tục xác định các lớp còn thiếu để hoàn chỉnh sơ đồ lớp. Các lớp còn thiếu này thường thuộc một trong các dạng sau:
 - *Các lớp biên:* là các lớp liên quan đến giao diện hệ thống, thực hiện nhận thông tin từ người dùng và gửi đến các đối tượng bên trong hệ thống. Gọi là các lớp biên vì các lớp này phân tách phần bên trong và bên ngoài hệ thống. Thông thường, mỗi form nhận thông tin sẽ trở thành một lớp nhưng cũng có trường hợp nhiều form tương tự nhau sẽ được mô tả trong một lớp.
 - *Các lớp trung gian:* giữa hai lớp có quan hệ $m...n$ (nhiều thể hiện của lớp này tương ứng với nhiều thể hiện của lớp kia), người ta thường sinh ra thêm một lớp trung gian để chuyển quan hệ đó thành 2 quan hệ dạng $1..n$. Các lớp này cũng có thể đại diện cho một thực thể xác định trong hệ thống nhưng cũng có thể không đại diện cho một thực thể xác

định nào. Trong trường hợp không đại diện cho thực thể xác định nào, lớp trung gian sinh ra chỉ có tác dụng hỗ trợ cho quá trình lập trình và sẽ được đặt tên theo một quy định chung nào đó mà nhóm phát triển đưa ra.

- *Các lớp trừu tượng*: trong một số trường hợp, một số lớp có thể có các thuộc tính chung hoặc phương thức chung. Khi đó, để tiện cho cài đặt, người thiết kế có thể bổ sung thêm các lớp trừu tượng, tức là các lớp không có đối tượng. Các lớp trừu tượng không đại diện cho một thực thể tham gia trong hoạt động của hệ thống, do vậy, các lớp này có thể có dạng đặc biệt: hoặc chỉ có thuộc tính mà không có phương thức, hoặc ngược lại, chỉ có phương thức mà không có thuộc tính.
- *Các lớp điều khiển*: Là các lớp chỉ làm nhiệm vụ điều khiển hoạt động của hệ thống ứng với một chức năng nhất định. Thông thường, mỗi use case phức tạp đều phải có một lớp điều khiển tương ứng. Lớp điều khiển nhận thông tin từ các lớp biên (lớp giao diện), gửi yêu cầu đến các lớp thực thể để thực thi chức năng mà nó đảm nhiệm rồi lại trả về kết quả cho các lớp biên.
- ***Hiệu chỉnh mô tả thuộc tính và phương thức*** theo đúng chuẩn của ngôn ngữ sẽ sử dụng trong pha cài đặt hệ thống.
- ***Kiểm thử tính đúng đắn của biểu đồ lớp***. Người thiết kế có thể sử dụng một số công cụ để kiểm tra tính đúng đắn của biểu đồ lớp, hoặc tiến hành thử sinh khung mã theo ngôn ngữ đã chọn để kiểm tra và xác định lỗi trong biểu đồ lớp. Tuy nhiên, những cách này chỉ giúp tìm ra các lỗi cú pháp. Muốn tìm được các lỗi về ngữ nghĩa, người thiết kế phải xem xét lại tất cả các tài liệu của biểu đồ use case, scenario, biểu đồ trạng thái, biểu đồ tương tác và biểu đồ động.

Hình 4.10 mô tả một biểu đồ lớp chi tiết cho hệ thống quản lý thư viện. Trong biểu đồ này chưa xét đến các lớp giao diện (lớp biên) và chưa thể hiện các đặc tả đầy đủ của các phương thức và thuộc tính. So với sơ đồ lớp phân tích, sơ đồ lớp thiết kế cũng bổ sung thêm nhiều thuộc tính mới để chi tiết hoá các đối tượng.



Hình 4.9: Biểu đồ lớp chi tiết cho bài toán Quản lý thư viện

Sau khi có sơ đồ lớp như trên, người thiết kế cần tiếp tục mở rộng sơ đồ lớp, bổ sung các lớp biên và các lớp trung gian cho phù hợp với ngôn ngữ và môi trường lập trình.

4.3 THIẾT KẾ CHI TIẾT

Sau khi hoàn thành biểu đồ lớp, bước thiết kế chi tiết là bước rất gần gũi với lập trình cài đặt hệ thống. Nhiệm vụ của thiết kế chi tiết là:

- *Xây dựng biểu đồ hoạt động* để mô tả chi tiết các phương thức phức tạp trong biểu đồ lớp. Biểu đồ động này sẽ là cơ sở để người lập trình cài đặt chính xác phương thức.
- *Xây dựng các bảng thiết kế chi tiết.* Công việc này thường gắn liền với quá trình lập kế hoạch và phân công công việc trong quá trình cài đặt hệ thống.

Bảng thiết kế chi tiết của các lớp ngoài việc biểu diễn các thông tin về tên lớp, các thuộc tính và các phương thức, các tham số, kiểu và giá trị trả về, ... còn cần chỉ rõ người chịu trách nhiệm cài đặt lớp (hay modul) đó và các thông tin về thời gian yêu cầu.

Hai nhiệm vụ này sẽ được trình bày chi tiết trong hai phần tiếp theo của tài liệu.

4.3.1 Xây dựng biểu đồ hoạt động cho các phương thức

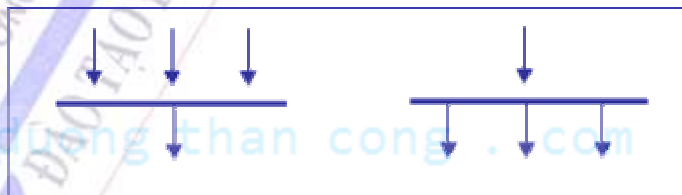
Biểu đồ hoạt động có thể được sử dụng cho nhiều mục đích khác nhau, ví dụ như:

- Để xác định các hành động phải thực hiện trong phạm vi một phương thức. Đây là vai trò thường gặp nhất và quan trọng nhất của biểu đồ hoạt động.
- Để xác định công việc cụ thể của một đối tượng.
- Để chỉ ra một nhóm hành động liên quan được thực hiện như thế nào và chúng sẽ ảnh hưởng đến những đối tượng nằm xung quanh.

Có thể xem biểu đồ hoạt động là một loại sơ đồ khối (Flow chart) miêu tả thuật toán. Điểm khác biệt là các sơ đồ khối bình thường chỉ được áp dụng đối với các quá trình tuần tự, còn biểu đồ hoạt động có thể xử lý cả các quá trình song song.

Các phần tử mô hình UML cho biểu đồ hoạt động bao gồm:

- *Hoạt động (Activity)*: là một quy trình được định nghĩa rõ ràng, có thể được thực hiện bởi một hàm hoặc một nhóm đối tượng. Hoạt động được thể hiện bằng hình chữ nhật bo tròn cạnh.
- *Thanh đồng bộ hóa (Synchronisation bar)*: chúng cho phép ta mở ra hoặc là đóng lại các nhánh chạy song song nội bộ trong tiến trình.

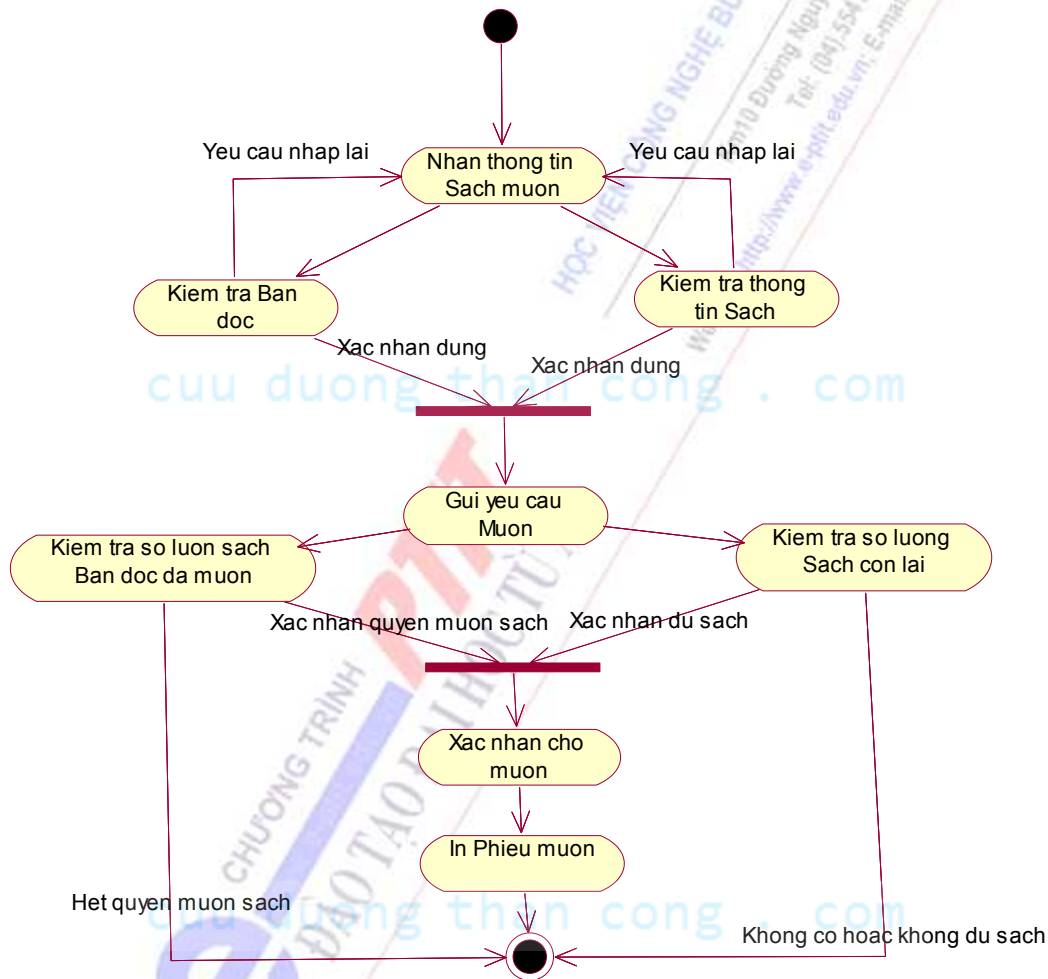


Hình 4.10: Thanh đồng bộ hoá trong biểu đồ động

- *Điều kiện (Guard Condition)*: các biểu thức logic có giá trị hoặc đúng hoặc sai. Điều kiện được thể hiện trong ngoặc vuông, ví dụ: [Customer existing].
- *Các luồng (swimlane)*: Mỗi biểu đồ động có thể biểu diễn sự phối hợp hoạt động trong nhiều lớp khác nhau. Khi đó mỗi lớp được phân tách bởi một

luồng (swimlane) riêng biệt. Các luồng này được biểu diễn đơn giản là các ô khác nhau trong biểu đồ.

Hình 4.11 mô tả một biểu đồ hoạt động cho phương thức mượn sách trong lớp Borrow_card. Trong biểu đồ này, ta có thể thấy có hai thanh đồng bộ hóa tương ứng với trạng thái chờ trong biểu đồ trạng thái. Ở đây, thanh đồng bộ thứ nhất sẽ chờ kết quả kiểm tra các thông tin người dùng nhập vào, thanh thứ hai sẽ chờ các kết quả kiểm tra liên quan đến điều kiện để có thể mượn sách.



Hình 4.11: Biểu đồ hoạt động cho phương thức Mượn sách lớp Thẻ mượn

Vấn đề quan trọng còn lại trong việc xây dựng biểu đồ hoạt động là xác định phương thức nào cần xây dựng biểu đồ hoạt động? Người thiết kế chỉ cần xây dựng biểu đồ hoạt động cho những phương thức phức tạp hoặc có vai trò quyết

định tới hoạt động của hệ thống. Việc đánh giá một phương thức có phức tạp hay không dựa theo các tiêu chí sau:

- Phương thức đó có cần xây dựng theo một thuật toán phức tạp hay không?
- Phương thức đó có tham chiếu tới nhiều phương thức của các lớp khác trong quá trình hoạt động hay không và ngược lại kết quả của phương thức đó có ảnh hưởng đến nhiều lớp khác hay không.
- Kết quả của phương thức đó có quyết định một chức năng (use case) cụ thể nào của hệ thống hay không.

4.3.2 Xây dựng bảng thiết kế chi tiết

Bảng thiết kế chi tiết được thiết kế riêng cho từng lớp. Mỗi nhóm phát triển có thể có một bảng thiết kế với cấu trúc riêng. Dưới đây là một mẫu bảng thiết kế tham khảo.

Tên lớp			
Người thiết kế			
Người cài đặt			
Thời gian			
Tên thuộc tính	Mô tả	Kiểu	Phạm vi
Thuoc_tinh_1			
Thuoc_tinh_2			
Tên phương thức	Mô tả	Giá trị trả về	Phạm vi
Phuong_thuc_1			
Phuong_thuc_2			
Đoạn khung mã cho lớp			

Bảng 4.3: Mẫu bảng thiết kế chi tiết lớp

4.4 BIỂU ĐỒ THÀNH PHẦN VÀ BIỂU ĐỒ TRIỂN KHAI

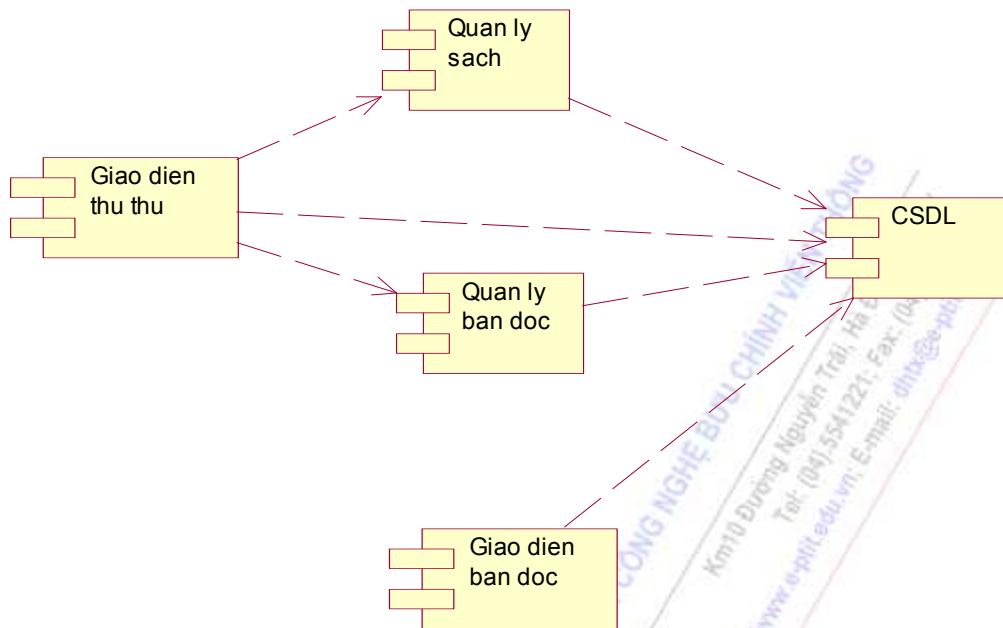
Sau khi đã hoàn thành biểu đồ lớp chi tiết và thiết kế chi tiết sử dụng biểu đồ hoạt động, trong bước này người thiết kế cần xác định rõ mô hình triển khai hệ thống và tiến hành phát sinh khung mã để chuyển sang pha cài đặt. Nội dung bước này gồm các hoạt động sau:

- Xây dựng biểu đồ thành phần
- Xây dựng biểu đồ triển khai
- Phát sinh mã cho hệ thống

4.4.1 Xây dựng biểu đồ thành phần

Mô hình thành phần được sử dụng để biểu diễn các thành phần phần mềm cấu thành nên hệ thống. Một hệ phần mềm có thể được xây dựng từ đầu sử dụng mô hình lớp như đã trình bày trong các phần trước của tài liệu, hoặc cũng có thể được tạo nên từ các thành phần sẵn có. Mỗi thành phần có thể coi như một phần mềm nhỏ hơn, cung cấp một khối dạng hộp đen trong quá trình xây dựng phần mềm lớn. Nói cách khác, các thành phần là các gói được xây dựng cho quá trình triển khai hệ thống. Các thành phần có thể là các gói ở mức cao như *JavaBean*, các gói thư viện liên kết động *dll*, hoặc các phần mềm nhỏ được tạo ra từ các thành phần nhỏ hơn như các lớp và các thư viện chức năng.

Hình 4.12 chỉ ra các thành phần có mặt trong hệ quản lý thư viện. Hệ thống cần quản lý các thông tin liên quan đến sách và bạn đọc do vậy sẽ có hai thành phần thực hiện các công việc này (*Quản lý sách* và *Quản lý bạn đọc*). Các thành phần quản lý này sẽ thao tác trên *CSDL* của hệ thống nên chúng ta có thành phần cài đặt *CSDL*. Ngoài ra hệ thống cũng cần một các thành phần giao tiếp với người dùng gồm *Giao diện bạn đọc* và *Giao diện thử thư* được cài đặt riêng trên các máy client. Thông thường, biểu đồ thành phần thường kết hợp với biểu đồ triển khai để trở thành một biểu đồ vật lý chung của cả hệ thống.



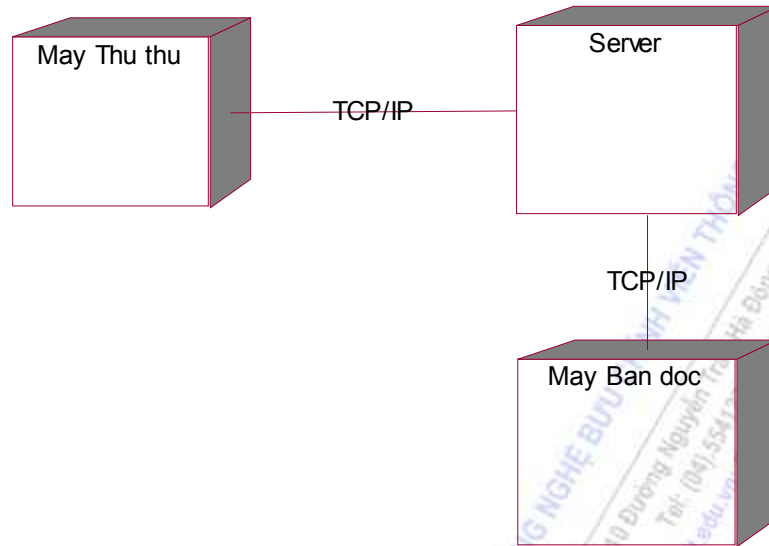
Hình 4.12: Các thành phần của hệ thống quản lý thư viện

4.4.2 Xây dựng biểu đồ triển khai

Biểu đồ triển khai biểu diễn các nodes và các mối quan hệ giữa chúng. Thông thường, các nodes được kết nối với nhau thông qua các liên kết truyền thông (communication association) như các kết nối mạng, liên kết TCP-IP, microwave...

Mối quan hệ giữa các node trong biểu đồ triển khai được biểu diễn thông qua các liên kết truyền thông và được đánh số theo thứ tự thời gian tương tự như trong biểu đồ cộng tác (collaboration diagram). Chú ý rằng các message truyền thông truyền đi giữa các node có thể là các luồng thông tin xác định hoặc cũng có thể là các đối tượng rời rạc, ví dụ như các file, các thông tin yêu cầu ...

Hình 4.14 biểu diễn biểu đồ triển khai cho hệ quản lý thư viện. Biểu đồ này cho biết hệ thống sẽ được cài đặt trên ba *dạng* máy tính khác nhau: các máy client dành cho thủ thư sẽ cài đặt thành phần *giao diện thủ thư*, *quản lý sách*, *quản lý bạn đọc*; các máy client dành cho bạn đọc chỉ cài *giao diện bạn đọc*; CSDL và thành phần điều khiển CSDL được cài trên một server chung gọi là Server.



Hình 4.13: Biểu đồ triển khai cho hệ quản lý thư viện

Các dạng liên kết truyền thông có thể có trong biểu đồ triển khai là:

- TCP/IP: sử dụng bộ giao thức TCP/IP để liên kết. Thông thường đây là các ứng dụng dựa trên Web.
- SNA: cũng là ứng dụng dựa trên Web nhưng sử dụng bộ giao thức SNA.
- Microwave: sử dụng liên kết bằng sóng vô tuyến tần số cao.
- Hồng ngoại: sử dụng liên kết hồng ngoại.
- Giao thức không dây: liên kết sử dụng các dạng giao thức không dây khác.

Trong ví dụ Hình 4.13, các liên kết đều được thực hiện trên nền giao thức TCP/IP thông qua kết nối mạng Internet hoặc kết nối mạng LAN nội bộ.

Ngoài các liên kết truyền thông thông thường, giữa các node còn có thể có mối quan hệ dạng phụ thuộc. Mối quan hệ phụ thuộc sẽ được biểu diễn bởi các mũi tên đứt nét với kiểu chính là dạng phụ thuộc giữa hai node (hoặc hai thành phần). Kết quả của biểu đồ triển khai kết hợp với biểu đồ thành phần là một mô hình triển khai hệ thống đầy đủ với các node, liên kết giữa các node và các thành phần bên trong các node đó. Mô hình này được gọi chung là mô hình vật lý (physical model) của hệ thống và sẽ là cơ sở để cài đặt, tích hợp hệ thống cũng như triển khai hệ thống tới người sử dụng.

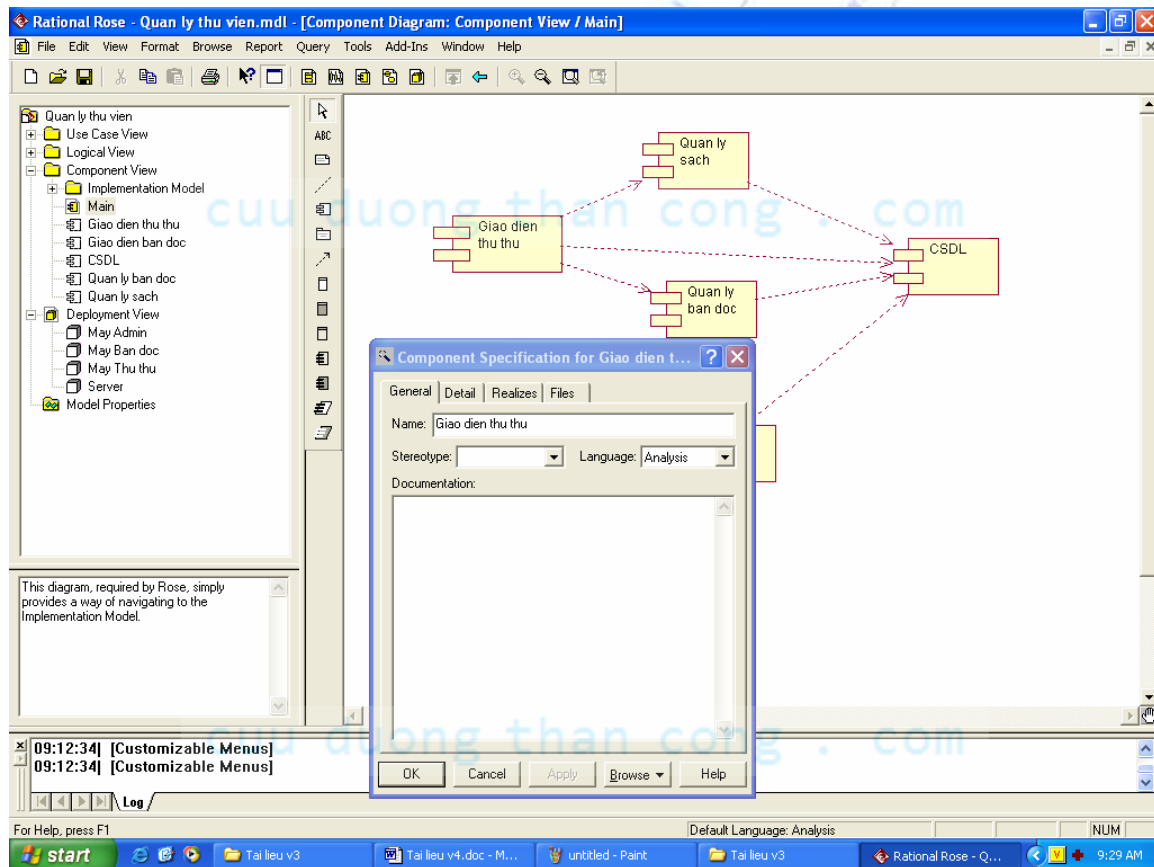
4.4.3 Biểu diễn biểu đồ thành phần và biểu đồ triển khai trong Rational Rose

Biểu đồ thành phần được xây dựng trong Component View để biểu diễn các thành phần trong hệ thống.

Các bước xây dựng biểu đồ thành phần trong Rational Rose:

- B1. Thêm các thành phần: lựa chọn công cụ thành phần trong hộp công cụ và kéo vào biểu đồ.
- B2. Đặc tả các thành phần (thông thường chỉ mô tả tên)
- B3. Biểu diễn các quan hệ giữa các thành phần (nếu có).
- B4. Bổ sung các thành phần con (nếu có)

Một biểu đồ thành phần ví dụ và cửa sổ đặc tả thành phần được biểu diễn như trong Hình 4.14.



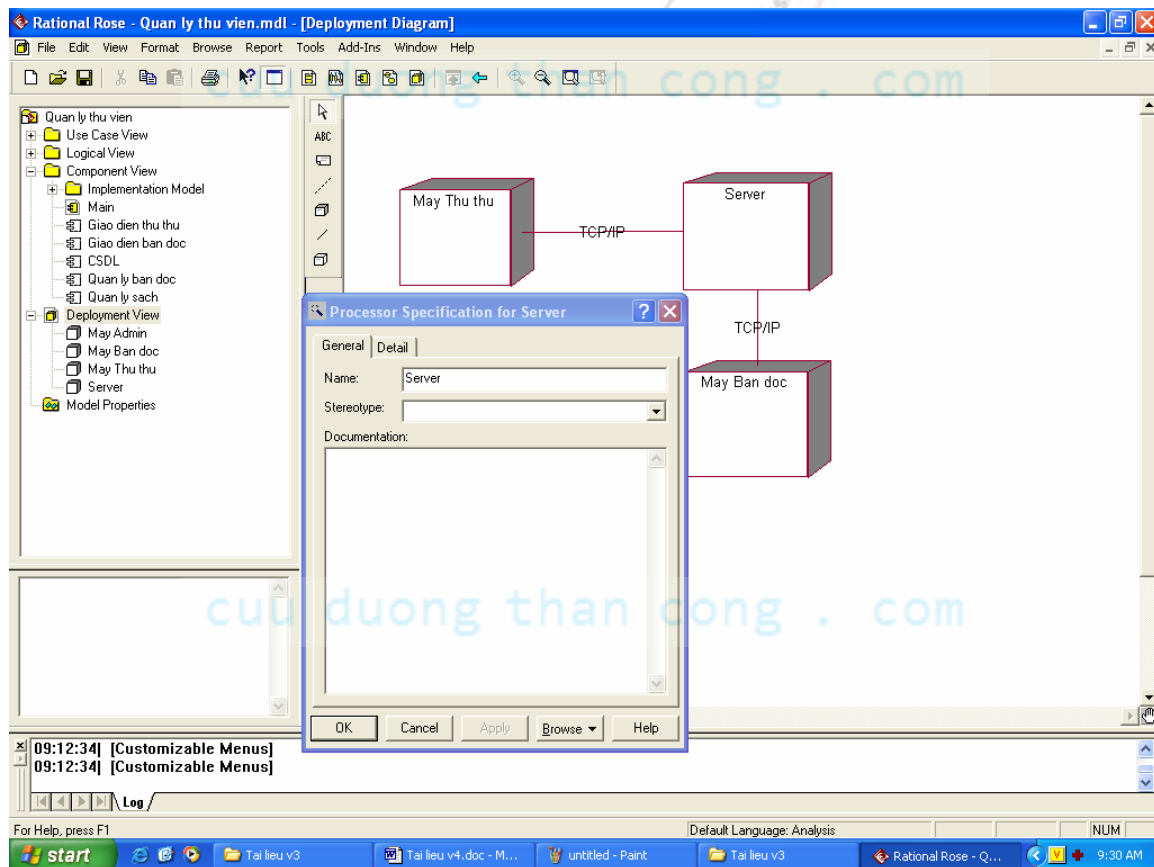
Hình 4.14: Xây dựng biểu đồ thành phần

Biểu đồ triển khai được xây dựng trong Deployment View. Các công cụ chính để xây dựng một biểu đồ triển khai trong Rational Rose là các Processor, Device và các Connection.

Các bước xây dựng biểu đồ triển khai trong Rational Rose:

- B1. Thêm các Processor: lựa chọn công cụ processor trong hộp công cụ và kéo vào biểu đồ.
- B2. Thêm các Device: lựa chọn công cụ Device trong hộp công cụ và kéo vào biểu đồ.
- B3. Biểu diễn các quan hệ: lựa chọn công cụ protocol và kéo giữa các processor hay device tương ứng.

Hình 4.15 biểu diễn một sơ đồ triển khai đơn giản trong đó có ba Processor đại diện cho các máy tính có cài đặt hệ dịch vụ thư viện. Các thành phần của server sẽ được cài đặt trong Library Server, phần giao diện với bạn đọc sẽ cài đặt trong Student PC còn giao diện với các thủ thư sẽ cài đặt trong Librarian PC.



Hình 4.15: Xây dựng biểu đồ triển khai

TỔNG KẾT CHƯƠNG 4

Chương 4 đã trình bày các bước trong pha thiết kế hướng đối tượng. Các nội dung cần nắm vững gồm:

- Pha thiết kế hướng đối tượng gồm 4 bước: xây dựng biểu đồ tương tác, xây dựng biểu đồ lớp chi tiết, thiết kế chi tiết và xây dựng biểu đồ thành phần và biểu đồ triển khai.
- Trong bước xây dựng biểu đồ tương tác, người thiết kế biểu diễn lại các use case ứng với các đối tượng của các lớp đã xác định trong pha phân tích. Có hai dạng biểu đồ tương tác là: biểu đồ tuần tự (nhấn mạnh đến thứ tự thời gian các message) và biểu đồ cộng tác (nhấn mạnh đến vai trò của các đối tượng trong tương tác).
- Bước xây dựng biểu đồ lớp chi tiết thực hiện bổ sung các lớp thiết kế (lớp biên, lớp trung gian, lớp điều khiển ...); xác định và mô tả chi tiết các phương thức; và biểu diễn các quan hệ giữa các lớp. Kết quả của bước này là một biểu đồ lớp thiết kế hoàn chỉnh.
- Bước thiết kế chi tiết tiến hành xây dựng biểu đồ hoạt động để biểu diễn các phương thức phức tạp hoặc các hoạt động phối hợp nhiều đối tượng thuộc nhiều lớp khác nhau. Tiếp theo, bước thiết kế chi tiết cũng xây dựng bảng thiết kế chi tiết để phân công trách nhiệm cho các thành viên trong nhóm phát triển.
- Bước xây dựng biểu đồ triển khai hệ thống tiến hành xác định các thành phần, các giao thức mạng; quan tâm đến ngôn ngữ lập trình và môi trường ứng dụng để xác định mô hình kiến trúc triển khai hệ thống.

Tài liệu cũng đã đưa ra những gợi ý, hướng dẫn và các chú ý cho từng bước trong thiết kế hướng đối tượng.

CÂU HỎI – BÀI TẬP

A. CÂU HỎI

1. Phân biệt sự khác nhau giữa biểu đồ trạng thái cho một use case và biểu đồ trạng thái hệ thống
2. Biểu đồ tương tác dùng để làm gì.

3. Phân biệt hai kiểu biểu đồ tương tác: biểu đồ tuần tự và biểu đồ cộng tác
4. Một liên kết trong biểu đồ cộng tác biểu diễn cái gì
5. Biểu đồ hoạt động dùng để làm gì
6. Một hoạt động trong biểu đồ hoạt động mô tả cái gì
7. Một chuyển tiếp trong biểu đồ hoạt động biểu diễn cái gì.
8. Phân biệt các kiểu lớp: lớp thực thể, lớp biên, lớp điều khiển, lớp trừu tượng.

B. BÀI TẬP

1. Các biểu đồ tương tác được xây dựng chủ yếu dựa trên nguồn nào sau đây:

- A. Biểu đồ trạng thái
- B. Các biểu đồ use case
- C. Biểu đồ lớp
- D. Biểu đồ hoạt động

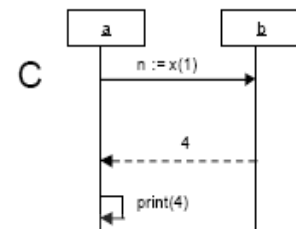
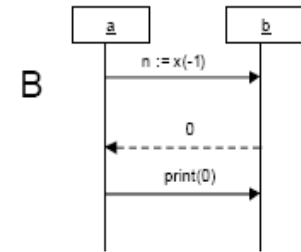
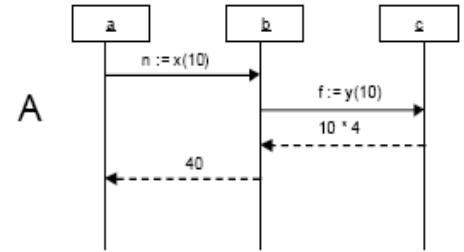
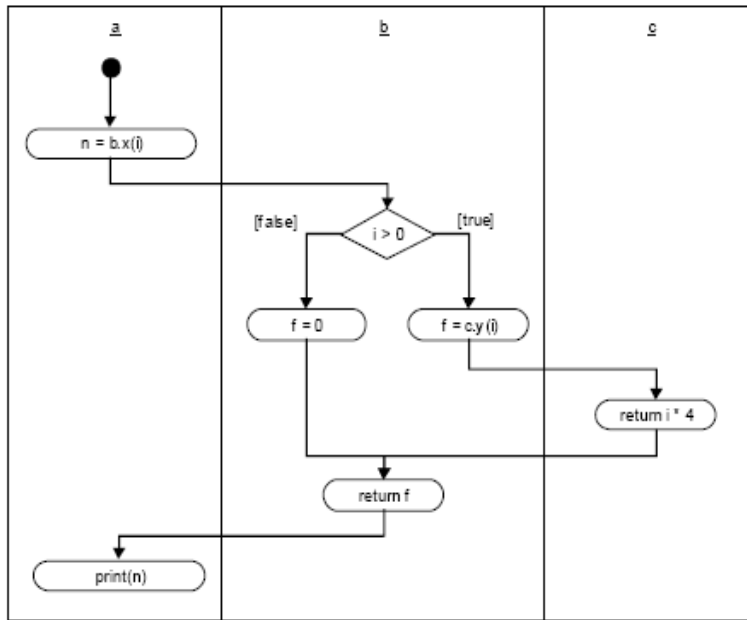
2. Để xem xét hoạt động của hệ thống có sự phối hợp của các đối tượng hoặc trong vòng đời của một đối tượng, ta có thể dùng các biểu đồ nào sau đây (Chọn 2)

- A. Biểu đồ lớp
- B. Biểu đồ use case
- C. Biểu đồ trạng thái
- D. Biểu đồ hoạt động

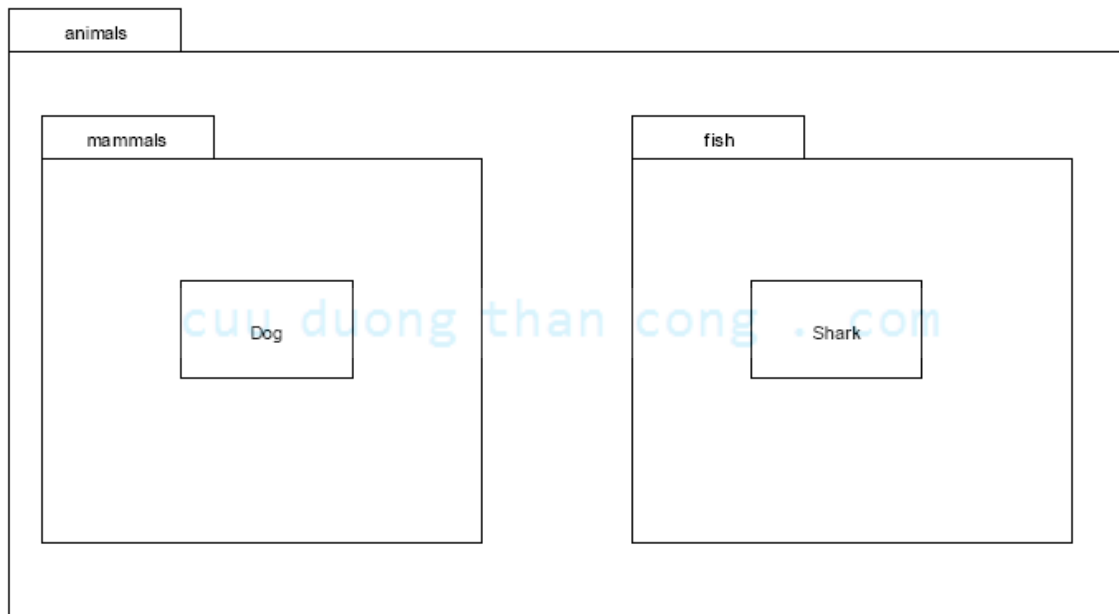
3. Để mô tả các thành phần (hoặc các đối tượng) của hệ thống được đặt ở đâu trong môi trường vật lý, chúng ta dùng biểu đồ nào sau đây:

- A. Biểu đồ hoạt động
- B. Biểu đồ trạng thái
- C. Biểu đồ thành phần
- D. Biểu đồ triển khai

4. Biểu đồ tuần tự (sequence diagram) nào sau đây là không phù hợp nếu nó nằm trong cùng mô hình với biểu đồ hoạt động đã cho. Giải thích.



5. Cho biểu đồ gỏi như hình vẽ. Các hàm trong lớp Dog muốn tham chiếu đến lớp Shark thì dùng ký pháp nào sau đây:



- A. animals::fish::Shark
- B. fish::Shark
- C. animals::Shark
- D. Không tham chiếu được đến lớp Shark

6.

- a) Sử dụng ngôn ngữ C++ để biểu diễn lớp Người có các thông tin sau: tên, tuổi, giới, chiều cao, cân nặng. Cài đặt phương thức tạo cho lớp Người và hai hàm gán Tên và gán Tuổi.
- b) Cài đặt tiếp lớp Nhân viên – có quan hệ kế thừa từ lớp Người, thêm thuộc tính lương. Hàm khởi tạo của lớp Nhân viên sử dụng lại hàm khởi tạo của lớp Người và gán lương mặc định bằng 0.

7. Giả sử đã có lớp Người như trong Bài 6. Hãy cài đặt liên kết 1-1 giữa người với Người là quan hệ Vợ-Chồng sử dụng ngôn ngữ C++ (bổ sung hai hàm getVoChong() và setVoChong).

8. Mở rộng lớp Người trong Bài 6 để cài đặt liên kết 1-1 và 1-nhiều giữa Người với Người xác định Cha, Mẹ hay Con của người đó (một người có một Cha, một Mẹ nhưng có thể có nhiều con) sử dụng ngôn ngữ C++.

9. Bổ sung thêm hệ thống trong Bài 6 hai lớp Tay và Chân. Hãy sử dụng ngôn ngữ C++ để cài đặt quan hệ gộp (composition) giữa lớp Người và hai lớp Tay, Chân.

PHỤ LỤC

PHÂN TÍCH THIẾT KẾ HỆ THỐNG THƯ VIỆN ĐIỆN TỬ

Hệ thống quản lý thư viện đã được giới thiệu và sử dụng làm các ví dụ trong chương 3 và 4 của tài liệu. Trong phần phụ lục này, tài liệu sẽ trình bày chi tiết các vấn đề liên quan đến hệ thống, xác định đầy đủ yêu cầu hệ thống và các biểu đồ UML trong quá trình phân tích thiết kế hệ thống.

Trong pha thiết kế, sau pha xây dựng biểu đồ lớp thiết kế, hệ thống sẽ được thiết kế theo từng chức năng (các use case) trong đó mỗi chức năng được thiết kế gồm các thành phần: giao diện của chức năng đó, lớp điều khiển và lớp thực thể.

1. GIỚI THIỆU HỆ THỐNG

1.1 Hoạt động nghiệp vụ thư viện

Theo nghiệp vụ quản lý thư viện thông thường, hoạt động thư viện của một trường đại học có thể được tóm tắt như sau:

- Thư viện làm các phích sách gồm các thông tin: mã số sách, tên tác giả, tên sách, nhà xuất bản, năm xuất bản, số trang, tóm tắt nội dung, số bản. Các phích sách có thể được phân theo chuyên ngành hoặc loại tài liệu.
- Mỗi sinh viên được cấp một thẻ thư viện gồm các thông tin: tên, tuổi, địa chỉ, lớp, chuyên ngành. Sinh viên muốn mượn sách thì tra cứu phích sách rồi ghi vào phiếu mượn.
- Ví dụ một phiếu mượn có thể có dạng như sau:

Số phiếu mượn:

Số thẻ TV:

Ngày mượn:

Số hiệu sách

Ngày trả:

Thuộc đơn vị:

Ngày hẹn trả:

Tên sách:

Tình trạng:

- Sau khi kiểm tra đầy đủ thông tin trên phiếu mượn, thủ thư kiểm tra điều kiện mượn của sinh viên và xác nhận cho phép mượn sách. Một số thông tin trong phiếu mượn được lưu lại để quản lý, phiếu mượn sẽ được gài vào chỗ sách được lấy đi, sách được giao cho sinh viên.
- Khi sinh viên trả sách: Từ thẻ sinh viên, xác định phiếu mượn, việc trả sách được ghi nhận vào dòng ngày trả và tình trạng. Phiếu mượn được lưu lại để quản lý và theo dõi.
- Sinh viên trả muộn hơn ngày hẹn trả sẽ bị phạt

1.2 Yêu cầu hệ thống

Hệ thống quản lý thư viện được xây dựng nhằm mục đích giải quyết các yêu cầu sau:

1. Giúp sinh viên tra cứu sách theo chuyên ngành, theo chủ đề, theo tên sách, theo tên tác giả, ... trên các máy tính trạm.
2. Cung cấp cho thủ thư các thông tin về các đầu sách một sinh viên đang mượn và hạn phải trả; và các cuốn sách còn đang được mượn.
3. Thống kê hàng tháng số sách cho mượn theo các chủ đề, tác giả ... Thống kê các đầu sách không có người mượn trên 1 năm, 2 năm, 3 năm.
4. Hỗ trợ thủ thư cập nhật thông tin sách, xác nhận cho mượn sách và nhận lại sách khi sinh viên trả sách.
5. Hỗ trợ quản lý các thông tin về sinh viên dựa trên thẻ thư viện, thông tin thẻ mượn.
6. Hỗ trợ chức năng quản trị chung hệ thống (admin) trong đó người quản trị chung có thể thay đổi thông tin hoặc thêm bớt các thủ thư.

Các yêu cầu phi chức năng:

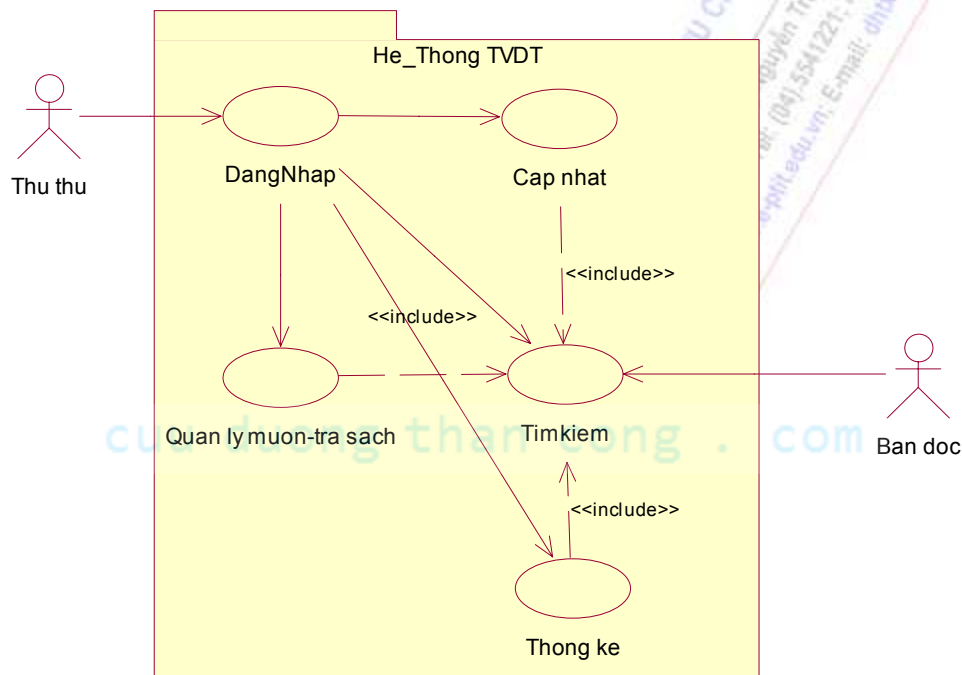
1. Hệ thống triển khai theo dạng Website trên hệ thống mạng nội bộ của trường
2. Sinh viên có thể tra cứu thông tin sách trên môi trường mạng. Tuy nhiên việc mượn và trả sách phải thực hiện trực tiếp trên Thư viện. Thủ thư sử dụng hệ thống để cập nhật và quản lý quá trình mượn trả sách.
3. Thông tin thống kê phải đảm bảo tính chính xác, khách quan. Các hình thức phạt với các sinh viên quá hạn sẽ được lưu lại và thông báo cho sinh viên biết.

2 PHA PHÂN TÍCH

2.1 Xây dựng biểu đồ use case

a) Biểu đồ use case tổng quát

Dựa trên các yêu cầu như trên, biểu đồ use case tổng quát của hệ thống sẽ mở rộng từ biểu đồ use case đã trình bày trong Chương 3. Biểu đồ này được biểu diễn trong Hình P.1.

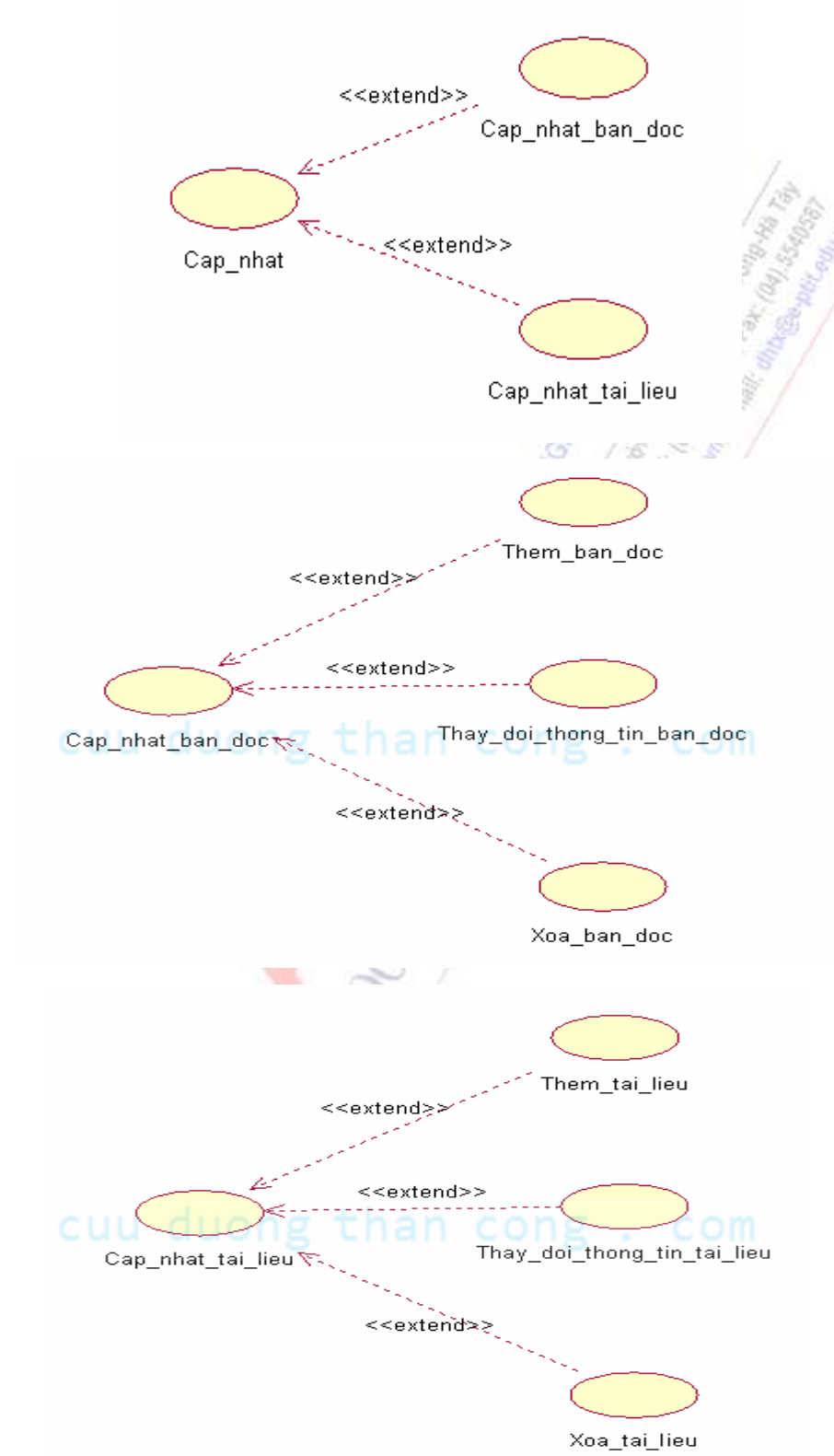


Hình P.1: Biểu đồ use case tổng quát của hệ thống

b) Phân rã biểu đồ use case

- **Phân rã use case Cập nhật**

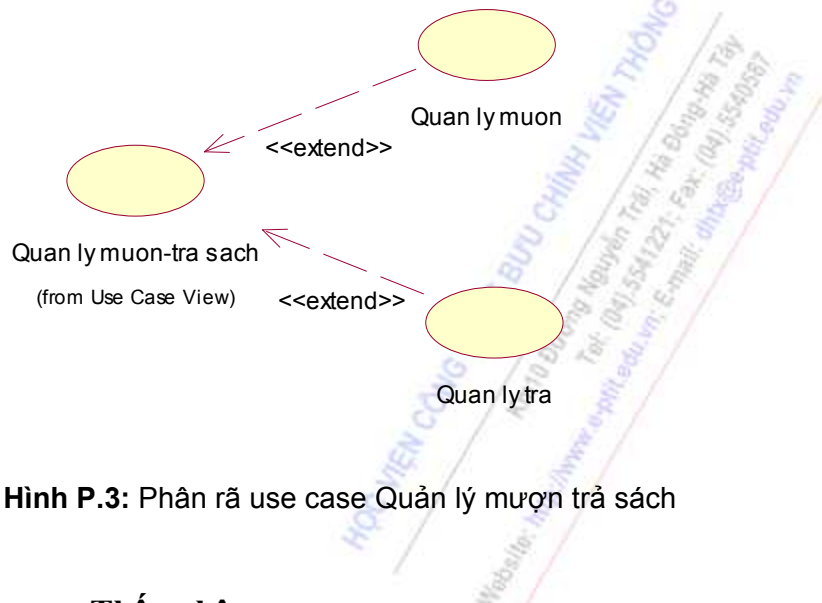
Quá trình phân rã use case Cập nhật hoàn toàn tương tự như đã trình bày trong chương 3 của tài liệu.



Hình P.2: Phân rã use case cập nhật

- **Phân rã use case Quản lý mượn trả sách**

Use case quản lý mượn – trả sách được thực hiện bởi thủ thư và có thể được phân rã thành hai use case nhỏ là Quản lý mượn và Quản lý trả (Hình P.3).

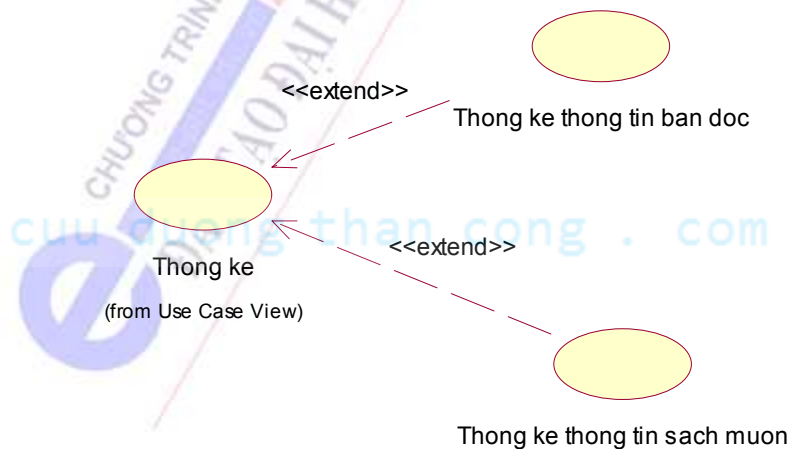


Hình P.3: Phân rã use case Quản lý mượn trả sách

- **Phân rã use case Thống kê**

Use case thống kê có thể được phân rã thành hai use case nhỏ hơn là:

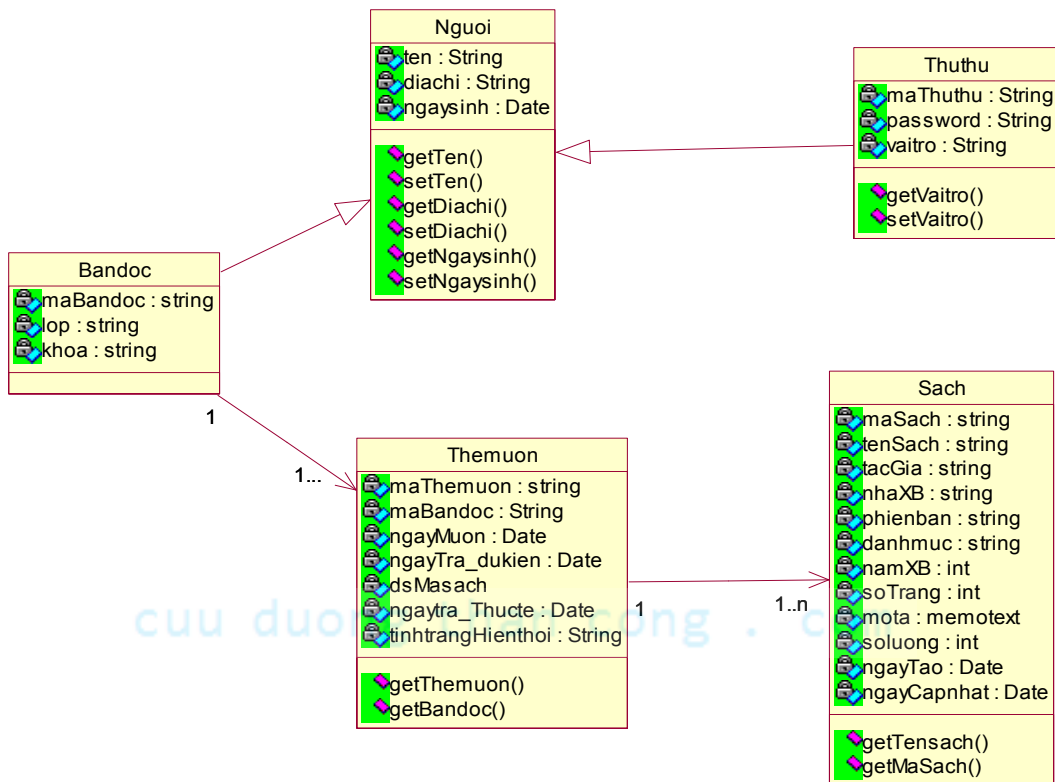
- Thống kê thông tin bạn đọc: cho biết danh sách các bạn đọc đang mượn sách, các bạn đọc quá hạn.
- Thống kê thông tin sách mượn: cho biết danh mục các cuốn sách đang được mượn, các cuốn sách lâu ngày không có ai mượn ...



Hình P.3: Phân rã use case Thống kê

2.2 Xây dựng biểu đồ lớp phân tích

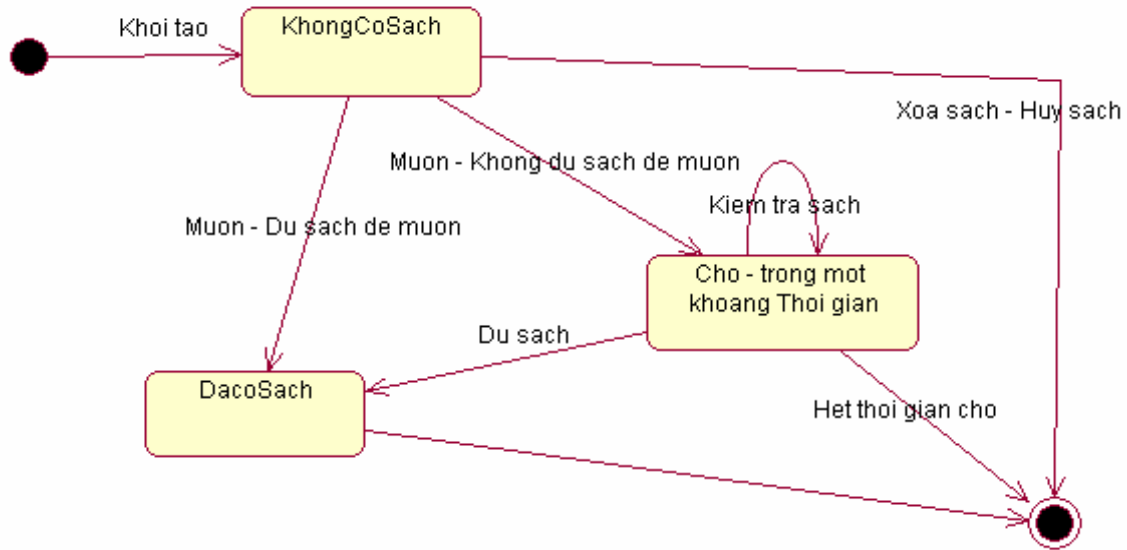
Biểu đồ lớp phân tích được xây dựng hoàn toàn tương tự như đã trình bày trong chương 3 của tài liệu này.



Hình P.4: Biểu đồ lớp phân tích của hệ thống

2.3 Biểu đồ trạng thái

Phần này trình bày hai biểu đồ trạng thái cho lớp Thẻ mượn (tương ứng với hai chức năng mượn sách và trả sách).



Hình P.5: Biểu đồ trạng thái lớp Thẻ mượn - chức năng mượn sách



Hình P.6: Biểu đồ trạng thái lớp Thẻ mượn - chức năng trả sách

3. PHA THIẾT KẾ

Trong phần này, tài liệu sẽ trình bày các biểu đồ UML được xây dựng trong pha thiết kế hệ thống Quản lý thư viện. Sau khi xây dựng các biểu đồ tương tác (dạng tuần tự), pha thiết kế sẽ đưa ra biểu đồ lớp thiết kế. Tuy nhiên trong biểu đồ lớp này chưa bổ sung các lớp giao diện và điều khiển. Phần 3.3 sẽ trình bày thiết kế

chi tiết theo từng chức năng trong đó mỗi chức năng có một (hoặc nhiều) lớp giao diện và một (hoặc nhiều) lớp điều khiển cùng với các lớp thực thể tương ứng.

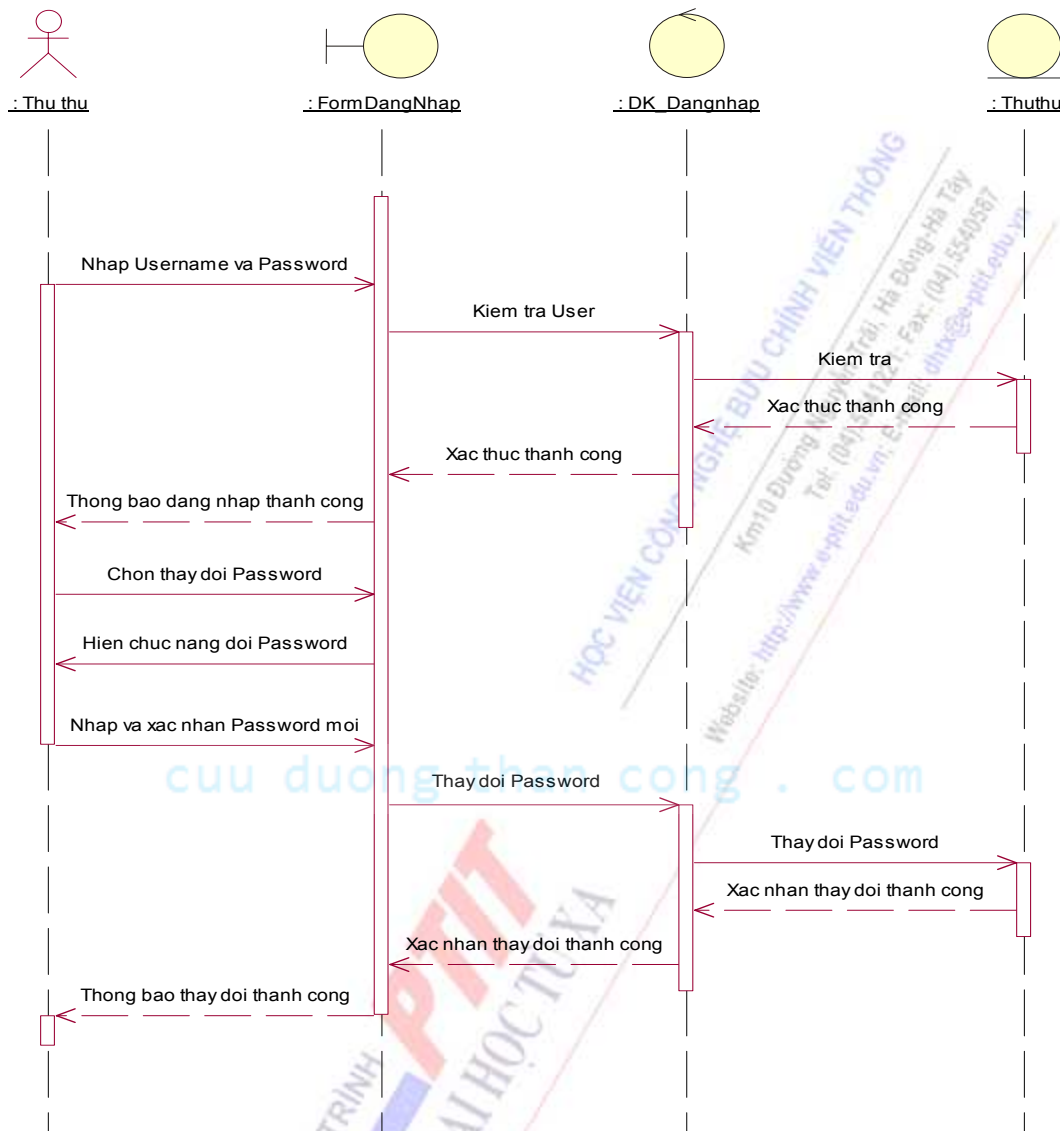
3.1 Các biểu đồ tuần tự

Trong hệ thống quản lý thư viện, chúng ta lựa chọn biểu đồ tương tác dạng tuần tự để biểu diễn các tương tác giữa các đối tượng. Để xác định rõ các thành phần cần bổ sung trong biểu đồ lớp, trong mỗi biểu đồ tuần tự của hệ thống quản lý thư viện sẽ thực hiện:

- Xác định rõ kiểu của đối tượng tham gia trong tương tác (ví dụ giao diện, điều khiển hay thực thể).
- Mỗi biểu đồ tuần tự có thể có ít nhất một lớp giao diện (Form) tương ứng với chức năng (use case) mà biểu đồ đó mô tả
- Mỗi biểu đồ tuần tự có thể liên quan đến một hoặc nhiều đối tượng thực thể. Các đối tượng thực thể chính là các đối tượng của các lớp đã được xây dựng trong biểu đồ thiết kế chi tiết.

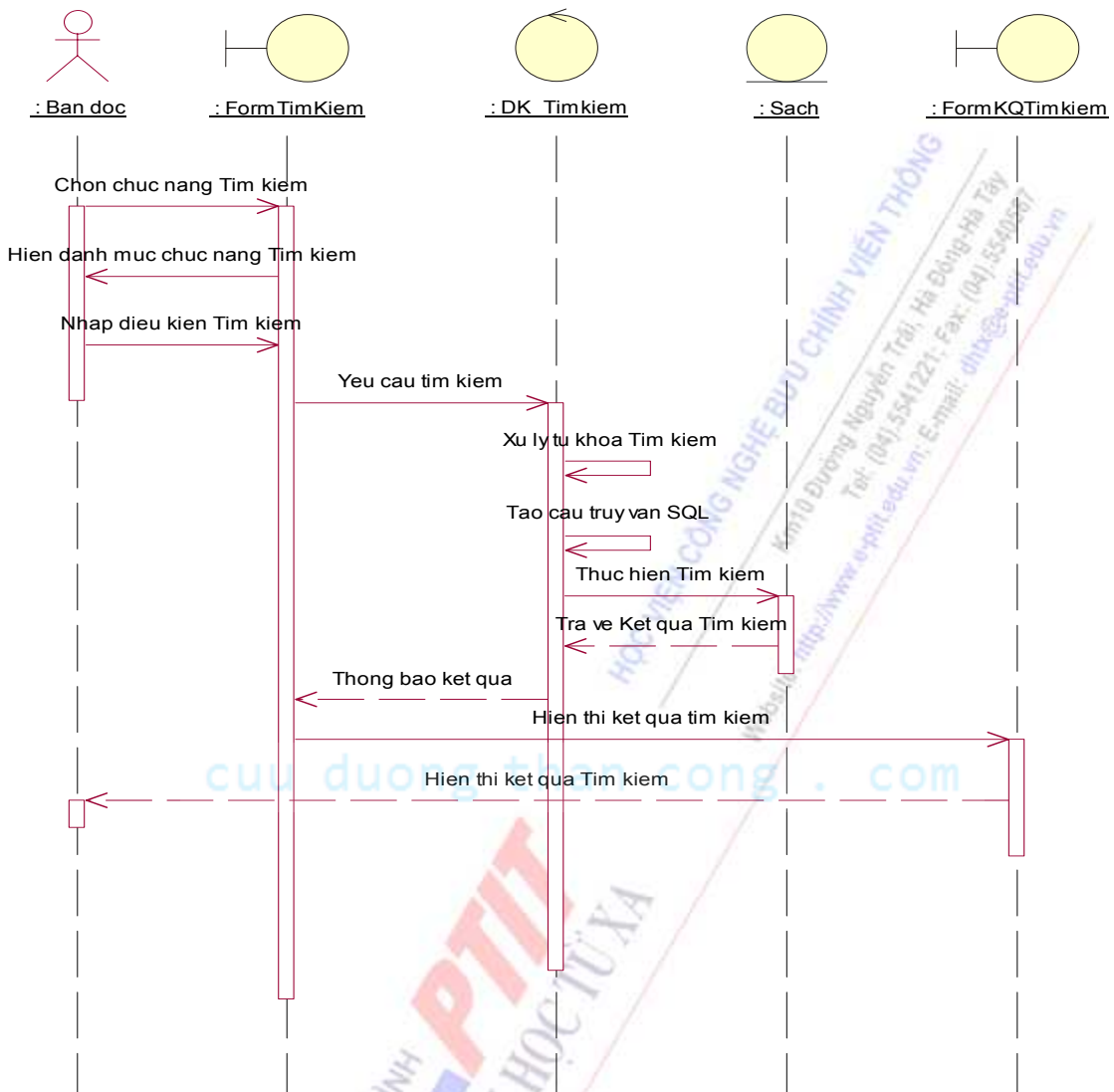
Dưới đây là một số biểu đồ tuần tự cho các chức năng của hệ thống:

a) Biểu đồ tuần tự cho chức năng Đăng nhập



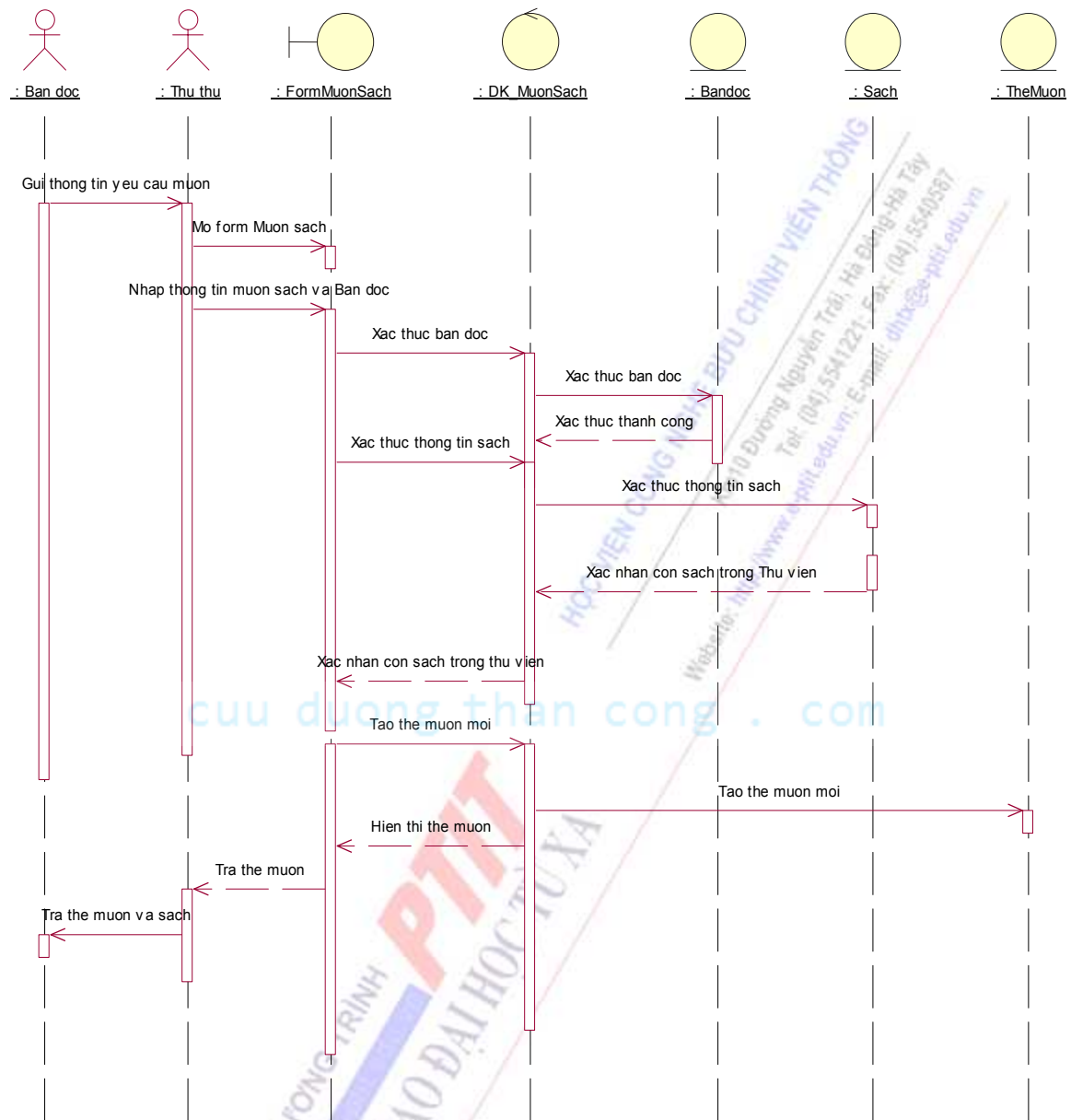
Hình P.7: Biểu đồ tuần tự cho chức năng Đăng nhập

b) Biểu đồ tuần tự cho chức năng Tìm kiếm



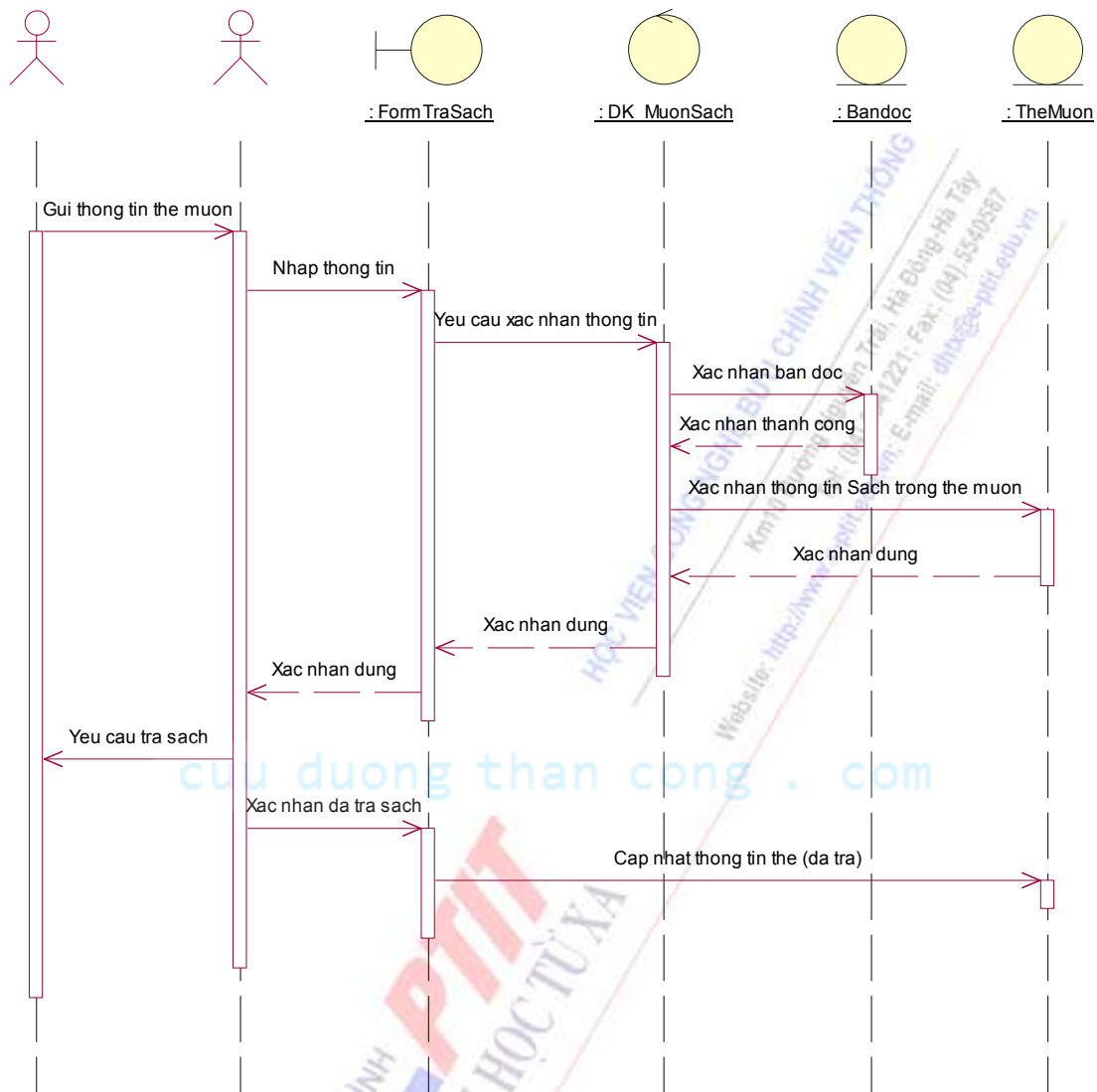
Hình P.8: Biểu đồ tuần tự cho chức năng Tìm kiếm

c) Biểu đồ tuần tự cho chức năng Quản lý mượn sách



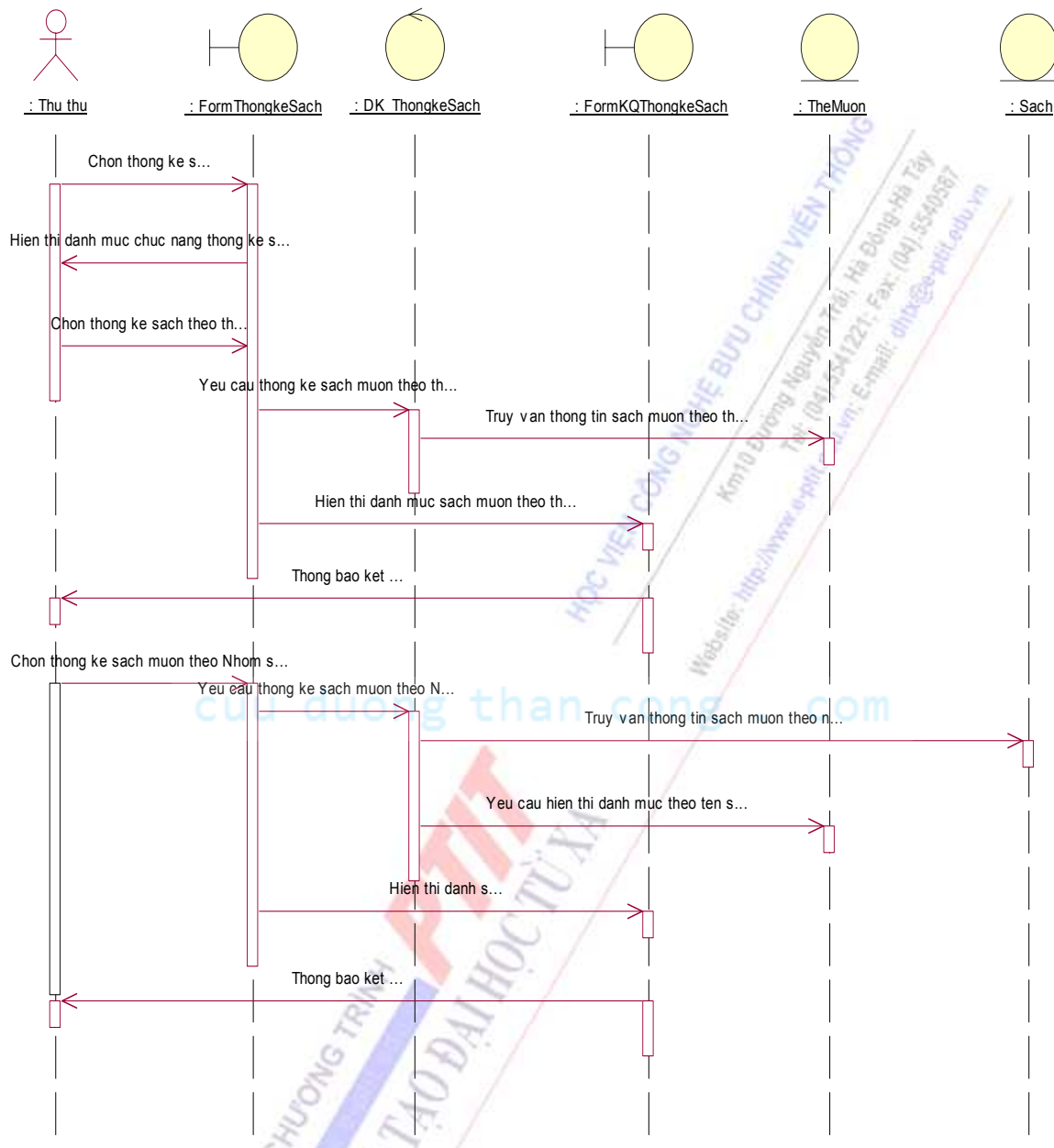
Hình P.9: Biểu đồ tuần tự cho chức năng Quản lý mượn sách

d) Biểu đồ tuần tự cho chức năng Quản lý trả sách



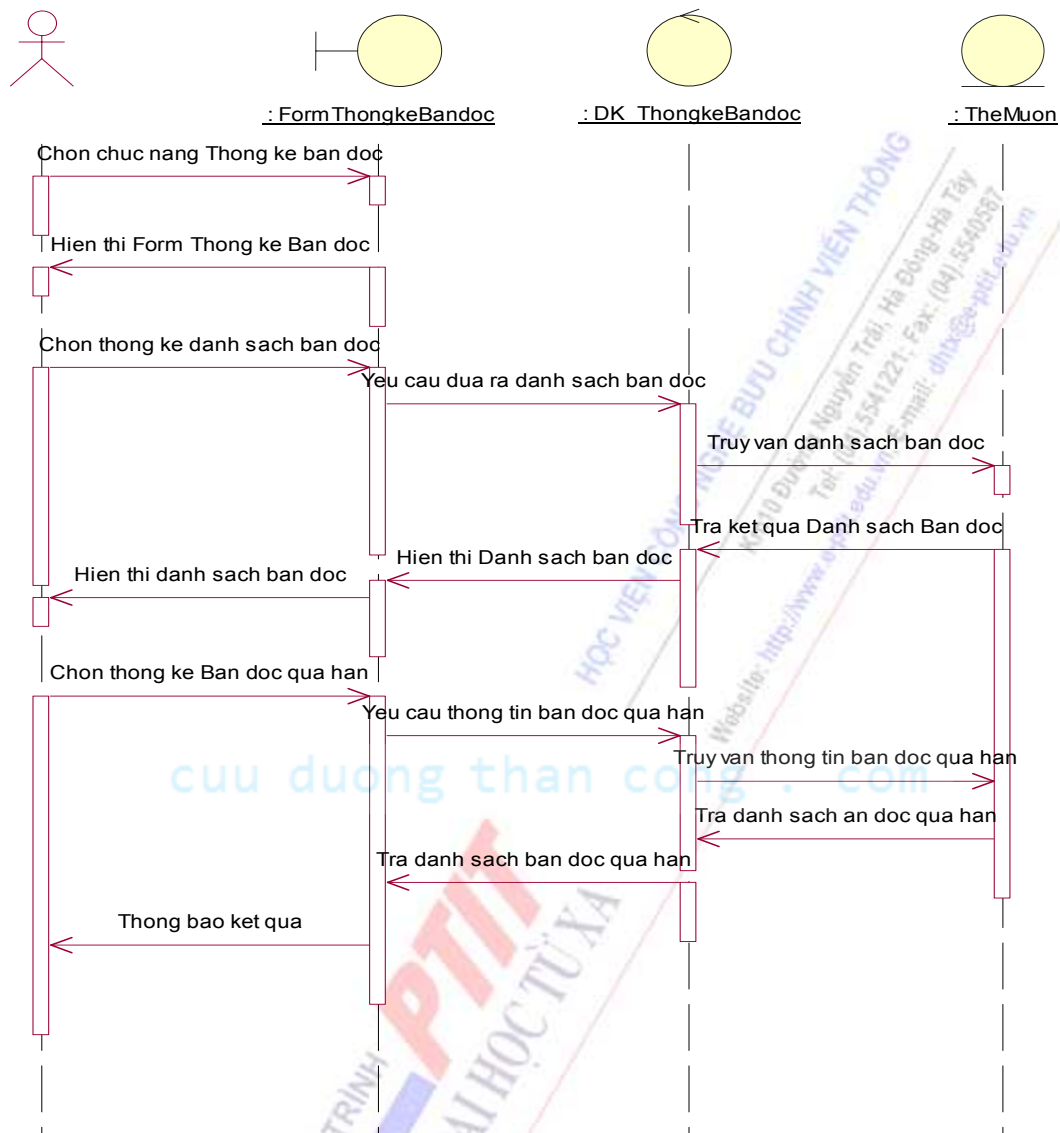
Hình P.10: Biểu đồ tuần tự cho chức năng Quản lý trả sách

e) Biểu đồ tuần tự cho chức năng Thống kê thông tin sách



Hình P.11: Biểu đồ tuần tự cho chức năng Thống kê thông tin sách

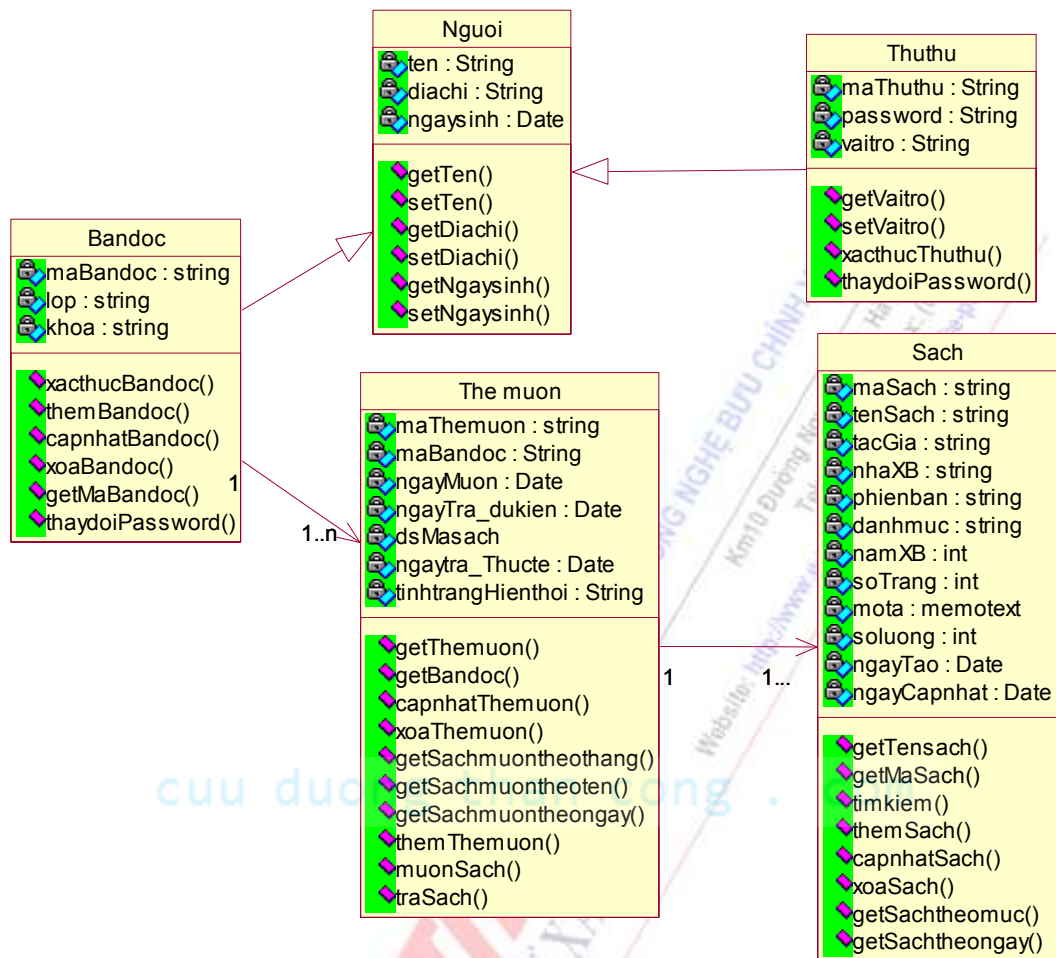
f) Biểu đồ tuần tự cho chức năng Thống kê thông tin bạn đọc



Hình P.12: Biểu đồ tuần tự cho chức năng Thống kê thông tin bạn đọc

3.2 Biểu đồ lớp chi tiết

Dựa trên biểu đồ lớp trong pha phân tích và các biểu đồ trạng thái, biểu đồ tuần tự, biểu đồ lớp thiết kế được xây dựng như trong Hình P.13. Biểu đồ lớp thiết kế bổ sung nhiều thuộc tính và phương thức so với biểu đồ lớp phân tích.



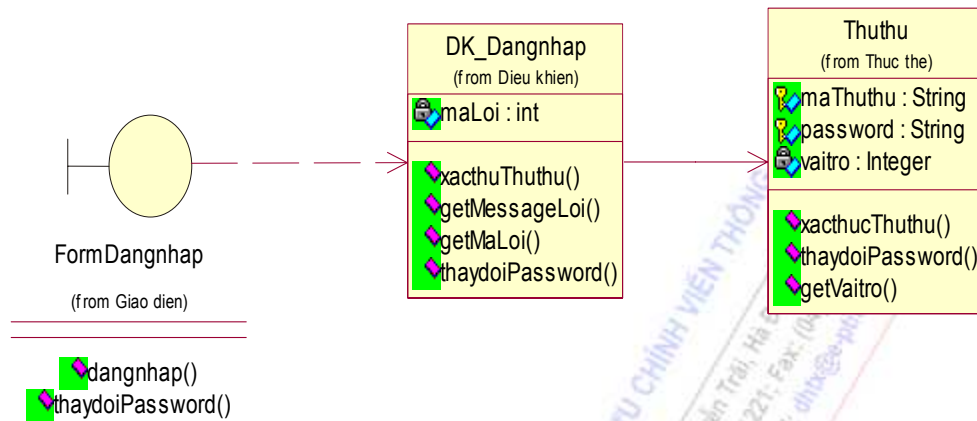
Hình P.13: Biểu đồ lớp thiết kế

3.3 Thiết kế riêng từng chức năng

Với mỗi chức năng, pha thiết kế sẽ xác định:

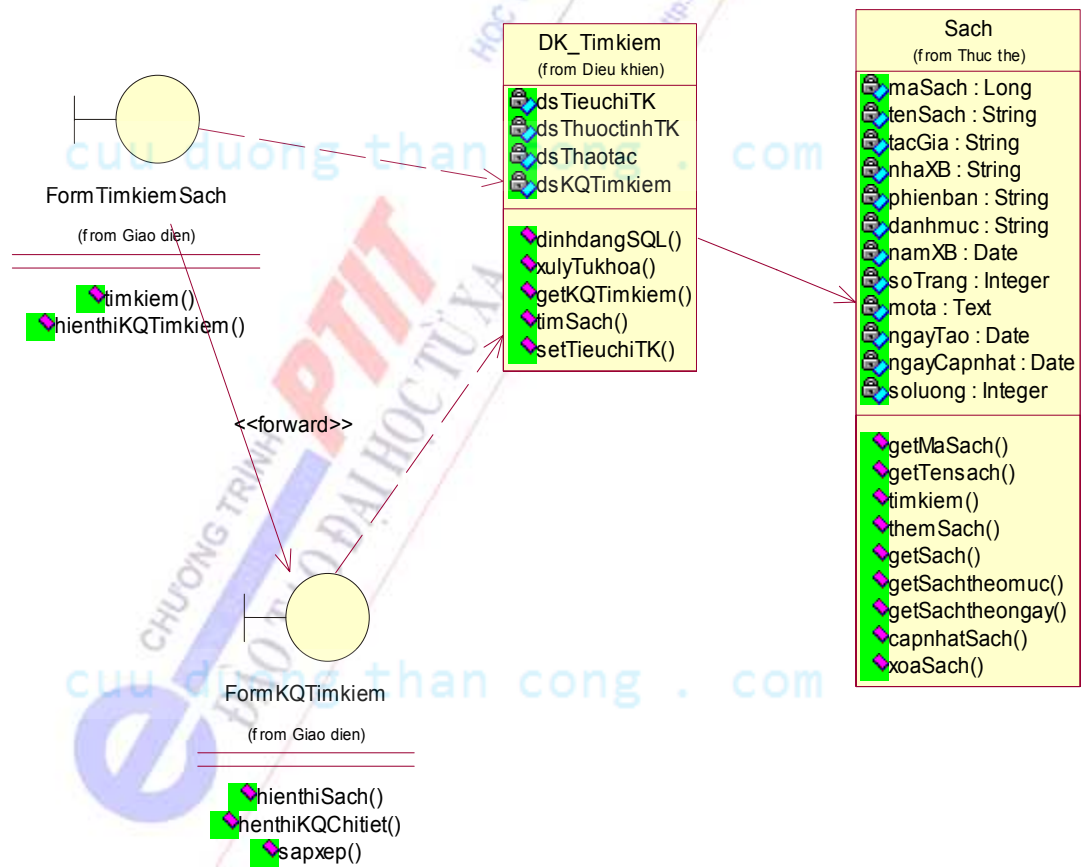
- Các lớp giao diện tương ứng
- Lớp điều khiển
- Lớp thực thể
- Các mối quan hệ giữa các lớp trên trong chức năng đó

a) Chức năng Đăng nhập



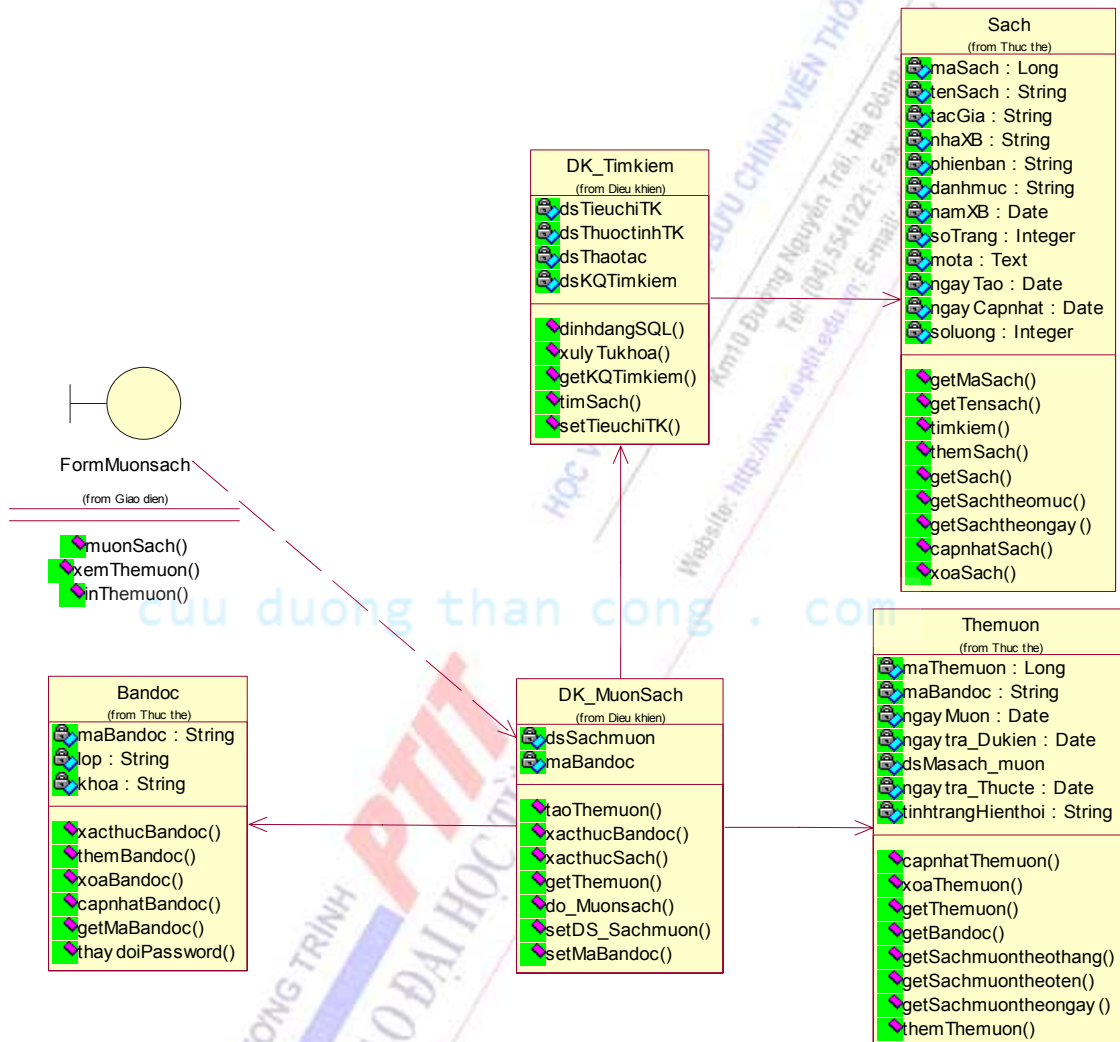
Hình P.14: Thiết kế lớp cho chức năng Đăng nhập

b) Chức năng Tìm kiếm



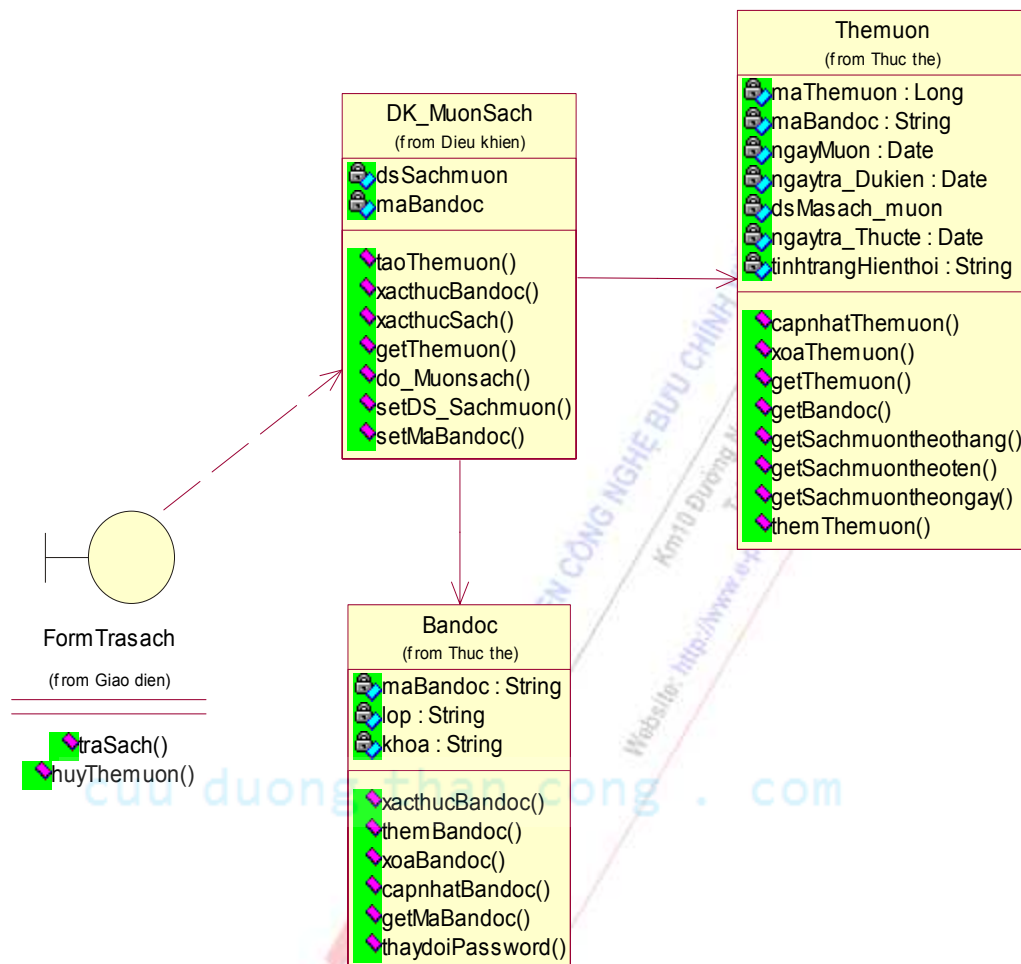
Hình P.15: Thiết kế lớp cho chức năng Tìm kiếm

c) Chức năng Quản lý mượn sách



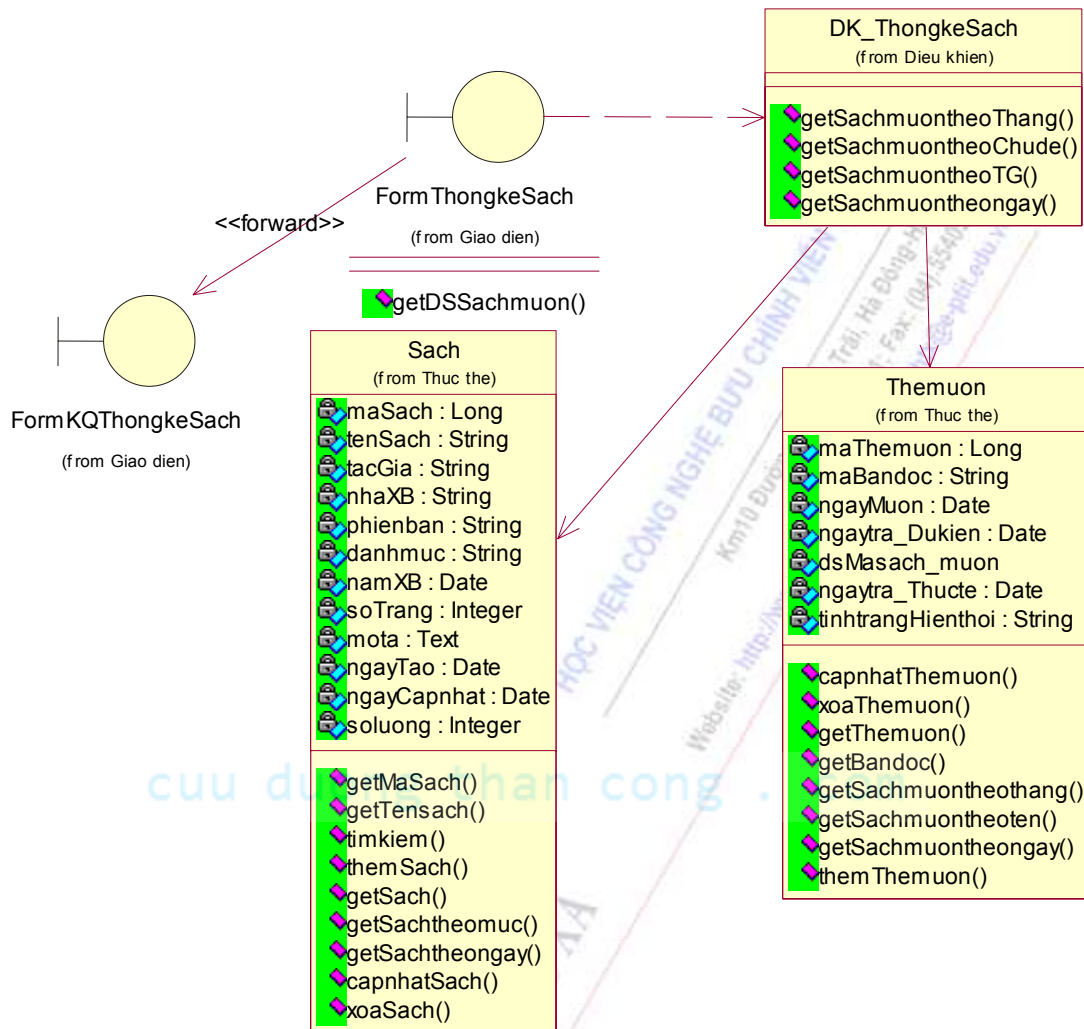
Hình P.16: Thiết kế lớp cho chức năng Quản lý mượn sách

d) Chức năng Quản lý trả sách



Hình P.17: Thiết kế lớp cho chức năng Quản lý trả sách

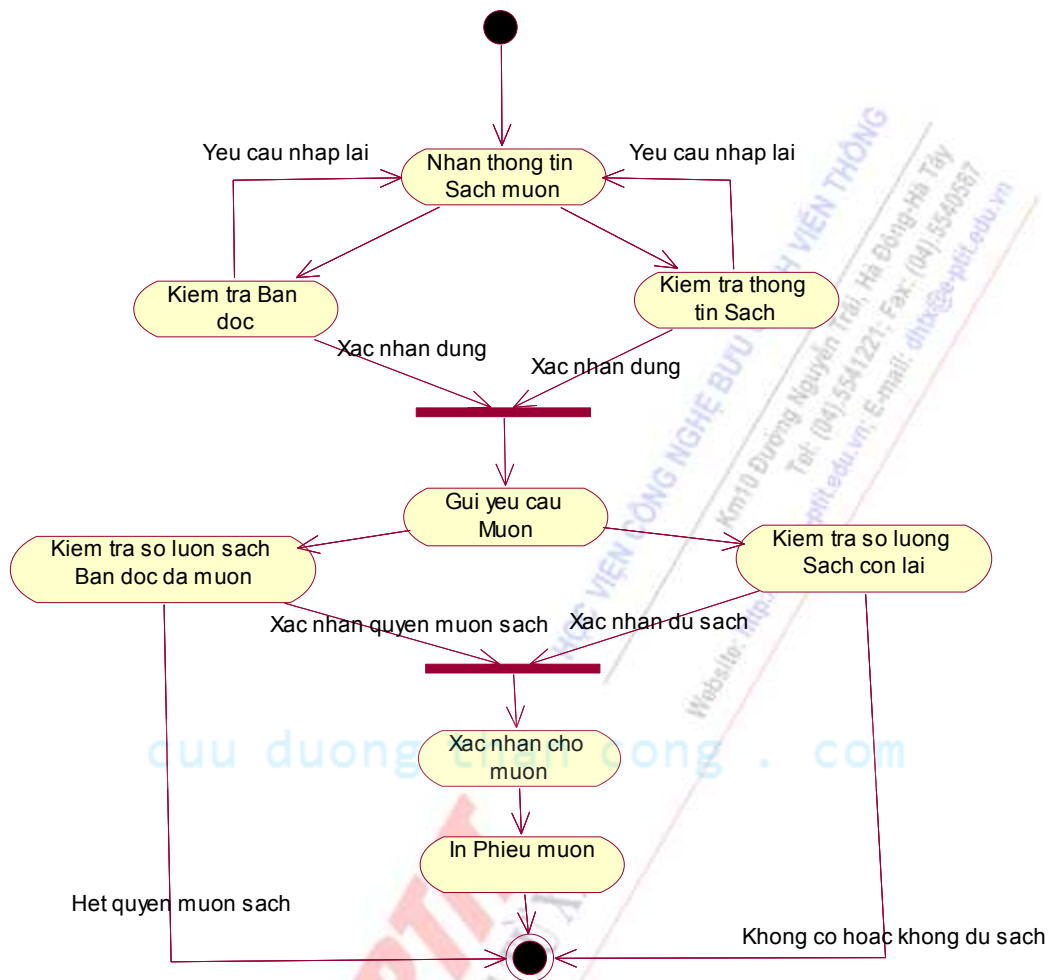
e) Chức năng Thống kê thông tin sách



Hình P.18: Thiết kế lớp cho chức năng Thống kê thông tin sách

3.4 Biểu đồ hoạt động

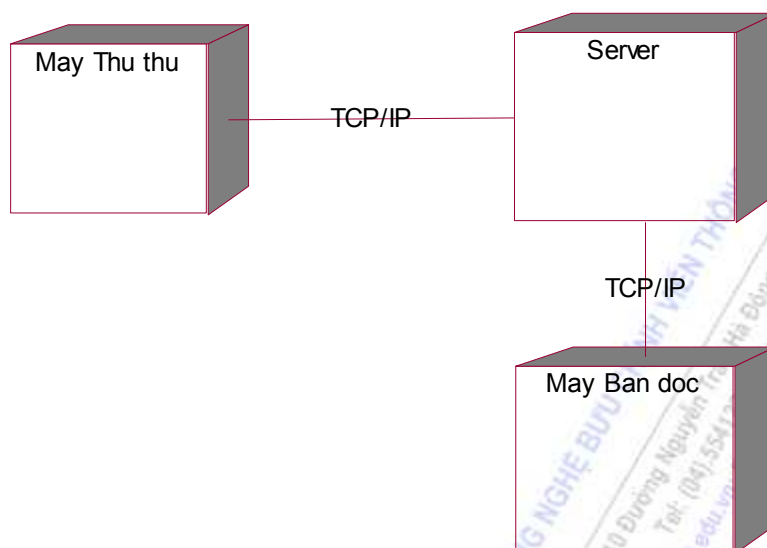
Vì đa số các chức năng (các hàm) thiết kế đều tương đối rõ ràng nên hệ thống quản lý thư viện không cần xây dựng nhiều biểu đồ hoạt động. Phần này chỉ trình bày ví dụ một biểu đồ hoạt động đơn giản của chức năng mượn sách (trong lớp Thẻ mượn).



Hình P.19: Biểu đồ hoạt động của chức năng mượn sách

3.5 Biểu đồ triển khai hệ thống

Hệ thống được triển khai dưới dạng Website và cài đặt khác nhau trên các máy Client cho thủ thư và cho sinh viên. Biểu đồ triển khai hệ thống được mô tả đơn giản như dưới đây:



Hình P.20: Biểu đồ triển khai hệ thống

GỢI Ý TRẢ LỜI CÁC BÀI TẬP

CHƯƠNG 3

Bài 1. B

Bài 2. Quan hệ cộng hợp (Composition) vì Address là một thành phần của lớp Employee.

Bài 3. Quan hệ phụ thuộc.

Bài 4. b

a->e->b

a->d

Bài 5. D

Bài 6. C

Bài 7. A

CHƯƠNG 4

Bài 1. B

Bài 2. C, D

Bài 3. D

Bài 4. B (Vì hàm print() không phải là phương thức của đối tượng b)

Bài 5. A

Bài 6.

```
class Nguoi{
```

```
    private:
```

```
        char ten[40];
```

```
        char gioi;
```

```
        int tuoi, chieucao, kannang;
```

```
        void setTen(char *s) {strcpy(ten,s);}
```

```
        void setTuoi(int a){tuoi=a;}
```

```
        void setGioi(char s){gioi=s;}
```

```

public:
    Nguoi(char *ten, char s, int a ){
        setTen(ten); setTuoi(a); setGioi(s);
    }
    ~Nguoi() {}
};

class Nhanvien: public Nguoi{
    private: int luong;
    public:
        Nhanvien(char *n, char s, int a, int l=0):Nguoi(n,s,a){
            luong =l;
        }
}

```

Bài 7.

```

class Nguoi{
    private:
        char ten[40];
        char gioi;
        int tuoi, chieucao, cannang;
        Nguoi* vochong;
        void setTen(char *s) {strcpy(ten,s);}
        void setTuoi(int a){tuoi=a;}
        void setGioi(char s){gioi=s;}
    public:
        Nguoi(char *ten, char s, int a ){
            setTen(ten); setTuoi(a); setGioi(s);
        }
        ~Nguoi(){};
}

```



```
    Ngươi* getVochong(){return vochong;}
    void setVochong(Ngươi *p){vochong = p; return;}
};
```

Bài 8.

```
class Ngươi{
    private:
        char ten[40];
        char gioi;
        int tuoi, chieucao, cannang;
        void setTen(char *s) {strcpy(ten,s);}
        void setTuoi(int a){tuoi=a;}
        void setGioi(char s){gioi=s;}
        Ngươi* Cha;
        Ngươi* Me;
        Ngươi* Con[];
        void setCha(Ngươi* p){Cha = p; return;}
        void setMe(Ngươi* p){Me = p; return;}
    public:
        Ngươi(char *ten, char s, int a ){
            setTen(ten); setTuoi(a); setGioi(s);
        }
        ~Ngươi(){};
        Ngươi* getCha(){return Cha;}
        Ngươi* getMe(){return Me}
        Ngươi* c[] getCon(){return Con}
};
```

Bài 9.

```
class Tay{
```

```
.....  
};  
class Chan{  
.....  
}  
  
class Nguoi{  
private:  
    char ten[40];  
    char gioi;  
    int tuoi, chieucao, cannang;  
    Tay taytrai;  
    Tay tayphai;  
    Chan chantrai;  
    Chan chanphai;  
    void setTen(char *s) {strcpy(ten,s);}  
    void setTuoi(int a){tuoi=a;}  
    void setGioi(char s){gioi=s;}  
public:  
    Nguoi(char *ten, char s, int a ){  
        setTen(ten); setTuoi(a); setGioi(s);  
    }  
    ~Nguoi(){}  
};
```

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Văn Ba, “Phát triển hệ thống hướng đối tượng với UML 2.0 và C++”, Nhà xuất bản Đại học Quốc gia Hà Nội, 2005.
- [2] Dương Anh Đức, “Bài giảng Phân tích thiết kế hướng đối tượng sử dụng UML”, Đại học KHTN - Đại học Quốc gia TP. HCM, 9-2000.
- [3] Đặng Văn Đức, “Phân tích thiết kế hướng đối tượng bằng UML”, Nhà xuất bản Giáo dục – 2001.
- [4] M. Fowler and K. Scott, “UML Distilled Second Edition – A Brief Guide to the Standard Object Modelling Language”, Addison Wesley Book, August 18, 1999.
- [5] L. Mathiassen, A. Munk-Madsen, P.A. Nielsen, J. Stage, “ObjectOriented Analysis&Design (OOA&D) – Concept, Principles & Methodology”, 2004.
- [6] R. LeMaster, D. Lebrknight, “Object-Oriented Programming & Design”, CSCI 4448, University of Colorado, 2002.
- [7] J. Jumbaugh, I. Jacobson, G. Booch, “The Unified Modelling Language Reference Manual”, 1999.
- [8] G. Sparks, “An Introduction to modelling software systems using the Unified Modelling Language”, <http://www.sparxsystems.com.au/>, 2000.
- [9] S. Sendall and A. Strhomeier, “Requirements Analysis with Use Case”, 2001
- [10] Sun Microsystems, “Object-Oriented Application Analysis and Design for Java Technology (UML) – Student Guide”, Revision B, March 2000.
- [11] The OMG Object Management Group, “OMG Unified Modeling Language Specification Version 1.5”, March 2003
- [12] “UML Notion Guide”,
online at <http://etna.int-evry.fr/COURS/UML/notation/index.html>.

PHÂN TÍCH THIẾT KẾ HỆ THỐNG THÔNG TIN

Mã số: 412PTH440

Chịu trách nhiệm bản thảo

TRUNG TÂM ĐÀO TẠO BƯU CHÍNH VIỄN THÔNG